# Utilities II

*Part No. 09845-10151*

Hewlett-Packard Desktop Computer Division
3404 East Harmony Road, Fort Collins, Colorado 80525

# Important

The tape cartridge or disc containing the programs is very reliable, but being a mechanical device, is subject to wear over a period of time. To avoid having to purchase a replacement medium, we recommend that you immediately duplicate the contents of the tape onto a permanent backup tape or disc. You should also keep backup copies of your important programs and data on a separate medium to minimize the risk of permanent loss.

# Table of Contents

**Introduction**

**Cross Reference Program**

**Listing Formatting Program**

**Search a File for Given Strings**

**Character Redefinition Program**

**Header Program**

**Banner Program**

**File Manager Program**

**9885 Flexible Disk Backup and Restoration Using Tapes**

**Duplicate a Tape**

**Duplicate a 9885 Flexible Disk**

**Bases Program**

**Convert Binary to Decimal**

**Convert Decimal to Binary**

**Convert Octal to Decimal**

**Convert Decimal to Octal**

**98036A Setup Subprogram - Prompter Version**

**98036A Setup - Parameter Version**

### Read Status of I/O Cards

### Read Usart Status Word for 98036A

### Set Time on 98035A Real Time Clock

### Display Time From 98035A Real Time Clock

### Menu Selection

# Introduction

The Utilities II software pack is a selection of relatively unrelated programs, subroutines, functions and subprograms that we hope will be of some use to you. They were written to satisfy some of your specific requirements without requiring you to do the actual coding. In addition to information about each program and how to use it, this manual includes a listing and cross reference for each one.

## Linking Files

Since some of the programs in this pack can be run by themselves, some are subprograms, and some are multiple-line functions, you should know how to link files together.

To link a subprogram or a multiple line function to your program you should have your program in memory. You should then type a command of the form:

a. Type: GET "FILE",5000

b. Press: EXECUTE

where "FILE" stands for the name of the file in which the subprogram or function is saved and 5000 will be altered by you to be some line number larger than the largest line number in your program. The function or subprogram will now be attached to the end of your program.

Once the subprogram is attached it may be used by your program by using a statement of the form:

```
1300 CALL Subprogram(P1,P2)
```

where Subprogram is the name of the subprogram in question and P1 and P2 are any parameters required by the subprogram.

In order to use a multiple line function you must use a statement of the form:

```
1300 X=FNFunction(P1,P2)
```

or

```
1300 X$=FNFunction$(P1,P2)
```

where X or X$ is the variable to which you are assigning the value of the function, FNFunction or FNFunction$ is the function in question and P1 and P2 are any parameters required by the function. A function may be included in an expression as well as being the entire right hand side of an assignment. However, a multiple line function may not be referened in a PRINT statement.

The subroutine can be attached in the same way that the subprogram and function were attached. It can be used by including a statement of the form:

```
1300 GOSUB Subroutine
```

The subroutine becomes an actual part of your program while the subprogram and the multiple line function are separate entities.

The programs in this pack are of two types. Some are SAVE as DATA files and some are STORE as PROG files. DATA files can be retrieved with these steps:

a. Type: GET "FILE"

b. Press: EXECUTE

while the PROG files can be retrieved with these steps:

a. Type: LOAD "FILE"

b. Press: EXECUTE

You can then begin the program by pressing RUN after the run light in the lower right hand corner of the CRT has disappeared and the tape cartridge has stopped.

## Entering Data

In the use of these utilities there will be times when you are asked to enter some data. When this is necessary a message of the form:

**ENTER: THE DATA REQUESTED**

will be displayed at the bottom of the CRT. The word ENTER will be in inverse video. You should enter the data requested, inspect it to make sure it is correct and then press CONT. The data will be entered.

There are times when information will be required to answer a Yes/No question. These questions will be of the form:

**WHAT IS THE RESPONSE? (YES/NO)**

To answer these questions you should type in "YES" (or "Y") or "NO" (or "N") and press CONT.

## Restrictions

This pack is designed for use with the 9845B. All programs, subprograms, functions and subroutines contained in this pack will run on the standard 9845B without alteration. In some instances the programs might be made more general by increasing the size of a data structure in the program if the 9845B has a larger amount of memory. If this is the case it will be mentioned in the program description.

The documentary programs will also run on the 9845A with 64K memory without alteration.

The Internal Thermal Printer Enhancements, of course, require an internal thermal printer. These programs will run on a 9845A with sufficient memory.

The Mass Storage Programs will work ONLY on the 9845B (NOT the 9845A).

The Base conversion routines will run on either the 9845B or the 9845A without alteration.

The I/O Card utilities and the Real Time Clock utilities require the I/O ROM as well as the applicable interface card. They will run on the 9845A as well as the 9845B.

The Menu subroutine will run ONLY on the 9845B (NOT the 9845A).

# Cross Reference Program

## Object of Program

This program produces cross reference listings for BASIC programs saved as DATA files on a mass storage medium. It alphabetically lists all variables used in the program and sequentially lists all line numbers of lines in which the variable is referenced. Variables which are included in a COM statement are listed separately, as are subprogram names referenced in a CALL statement and user defined functions invoked in the program. Also included is a sequential list of all lines which are referenced as jump targets (or are referenced as an IMAGE or DATA line) in the program. These lines are listed by line number but will also include a label for that line if one exists. Line numbers of lines referencing either the line number or the label are included in this case.

Since subprograms are independent of each other and of the main program they are treated as distinct entities by the cross referencer. The main program will be headed MAIN PROGRAM with subheadings COMMON VARIABLES, VARIABLES, USER DEFINED FUNCTIONS, SUB PROGRAMS and JUMP TARGETS. Subprograms will be headed "SUBPROGRAM Your—function".

## Technical Information

(May be skipped by the casual reader)

This program reads one line at a time from a specified DATA file on a mass storage device. The line is then analyzed for references to variables, subprogram names, user defined functions and jump targets by the subprogram ANALYZE. These references are stored and returned in the string array Token$(*) which is kept in COMMON. Names which are referenced for the first time are stored in the Name_table$(*) array. The position of the name in the table is determined by the first letter of the name referenced. In the case where the name is a line number the position is determined by its magnitude. The line number of the line being scanned is then stored in the reference table, Ref_table(*), for each name which was referenced in the line. In order to maintain a connection between the name table and the reference table a third table, Lk_up_tbl(*), the look up table, has been set up. This table is "parallel" to the name table. That is, for each entry in the name table there is a corresponding entry in the look up table (e.g. Lk_up_tbl(3,6) corresponds to Name table$(3,6)). The look up table holds pointers into the reference table which in turn holds a linked list of references to the corresponding name.

The name table and the look up table are divided into segments called blocks. Blocks 0 through 25 are used for variable names beginning with the letters A through Z. Block 26 is used to hold the names of user defined funtions. Subprogram names are kept in block 27. Labels are stored in block 28. Block 29 is reserved for variables found in a COM statement. Blocks 30 through 61 are used to store referenced line numbers, each block containing a multiple of 1000 (e.g. block 30 contains line numbers 1 through 999, block 32 contains line numbers 2000 through 2999). The zeroth element of the look up table block is used as a pointer to the first free element of that block. For example, if Lk_up_up_tble(1,0)=3 then the first position in block 1 without an entry is Name_table$(1,3).

Similarly the reference table is divided into "slots". Each slot holds references to a single name. Each time a new name is encountered a new slot is allocated to hold its references. The zeroth element of a slot holds a pointer to the first free element in the slot.

Blocks and slots are of finite size. In the program they are defined as follows

```
40 INTEGER Lk_up_up_tbl(150,5)
50 DIM Name_table$(150,5)[18]
60 INTEGER Ref_table(6:855,5)
```

Each slot and each block can hold five entries. They can become full. When this happens a new slot or block must be allocated to hold the overflow. In the case of a block the first free block (originally 62 since the first 61 are reserved) will be allocated. The zeroth element of the full block is set to point to the newly allocated block so that it now becomes an extension of the old block. (This pointer can be distinguished from the free element indicator originally there by the fact that the latter can only go up to 5 while the latter begins with 62). In this way an indefinite number of blocks can be strung together. For instance, if block 2 becomes full first block 62 would be allocated to hold the overflow and Lk_up_tbl(2,0) would be set to 62. The slots are handled in exactly the same manner. To distinguish element pointers from slot pointers, slots run from 6 to 855 instead of from 1 to 860.

For instance if "B_one", a variable name is encountered in the line

```
150 B_one=X*133
```

the following occur:

a) The first letter of "B_one" is "B", the second letter of the alphabet, so the "hashnumber" used to find the appropriate block is set to 1 (0 comes first, 1 second).

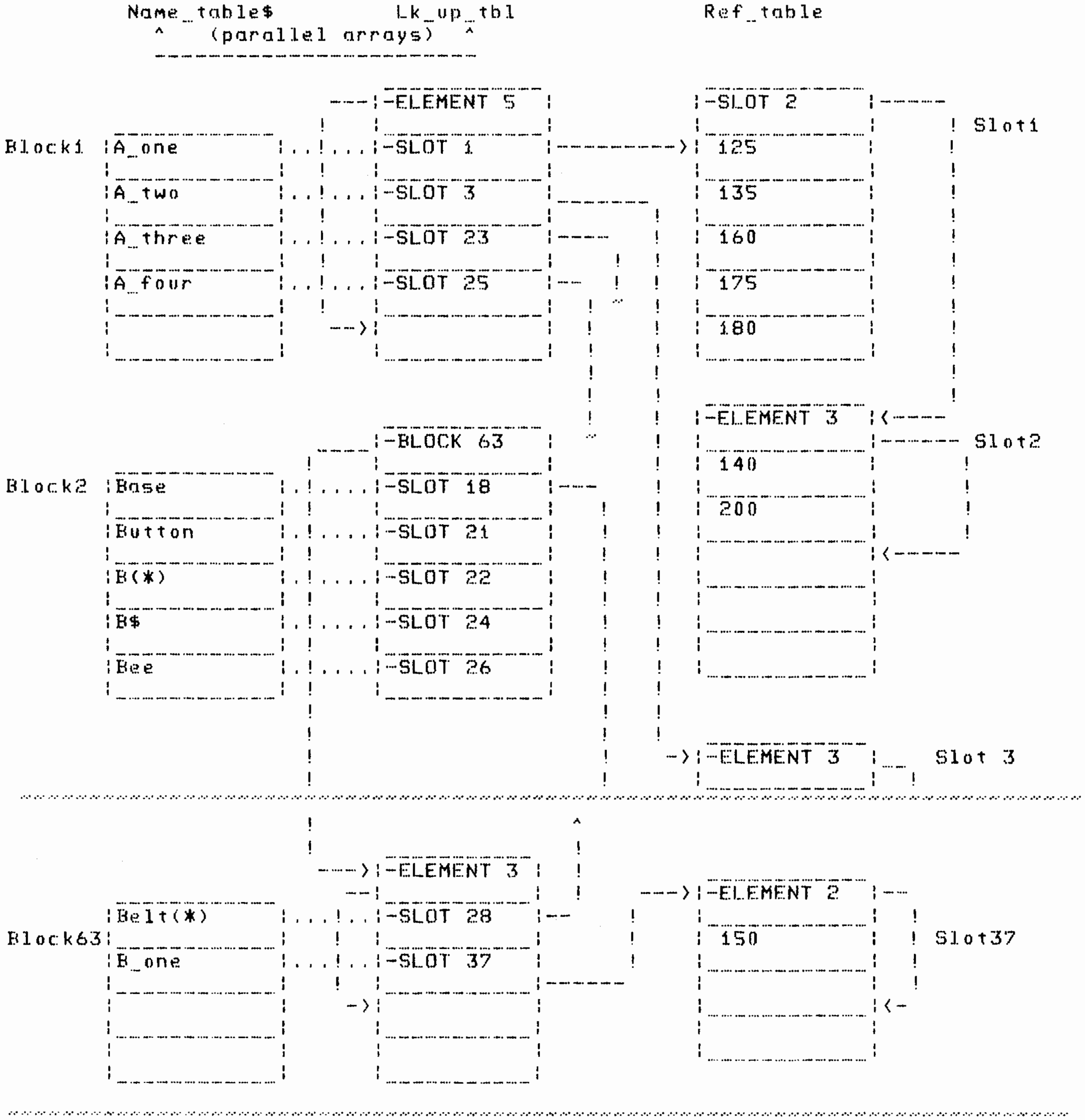
b) The name table is checked to see if "B_one" was previously encountered. Suppose Lk_up_tbl (1,0) is 63. This means that block 1 was filled and the overflow was sent to block 63. Therefore, if "B_one" is not found in block 1, block 63 must also be searched. If Lk_up_tbl(63,0) was greater than 5, then further blocks would have to be searched. If "B_one" is found then the line number of the line referencing it will be placed in the reference table.

c) If the name "B_one" has not been found then the new name is placed in the name table. Suppose Lk_up_tbl(63,0)=3. This means that Name_table$(63,3) is the first free element in the proper block. The name "B_one" is entered into Name_table$(63,3) and Lk_up_tbl(63,0) is increased to 4.

d) Since a new name has been entered, a new slot must be allocated to hold the references to it. Free_slot at present equals, say, 37. Slot 37 is allocated to hold references to "B_one". Lk_up_tbl(63,3) is set to 37 to point to this slot. Free slot is increased to 38.

e) The reference to "B_one" now has to be recorded in the reference table. Ref_table(37,0), the pointer element of slot 37, is initially set to 1 when allocated since Ref_table(37,1) is the first free element. Ref_table(37,1) is set to 150 (the line number of the current line) and Ref_table(37,0) is increased to 2.

A portion of the tables might appear as follows:



## Large Programs

The tables have been set up to fit the standard 9845B and so you may run out of space on an extremely large program. If the program you wish to cross reference is too large, you may wish to alter this program to accommodate a greater number of references. This can be done as follows:

If you have a configuration with more memory than the basic configuration you can increase the size of the storage arrays. You might double the size of the tables with these declarations:

```
40 INTEGER Lk_up_tbl(300,5)
50 DIM Name_table$(300,5)[18]
60 INTEGER Ref_table(6:1710,5)
```

Note that you MUST keep the dimensions of Name_table and Lk_up_tbl the same. The ratio of entries in these two tables and the Ref_table can vary quite a bit but the Ref_table should remain large in comparison to the other two. It is suggested that you do not change the size of the slots or blocks, the second dimensions of these arrays.

## User Instructions

The program CROSS REFERENCE is in DATA file "CROSS"

a. Type: GET "CROSS"

b. Press: EXECUTE

### Data to be entered

You will be asked to enter the select code of your hard copy printer (e.g. 0 for the internal thermal printer) and its bus code, if applicable. You will then be asked to enter the names of the files you wish to cross reference. These files should be on the same mass storage device and there should be no more than twenty of them at a time. All programs to be cross referenced should be on DATA files since the cross referencer uses the program in string form. All programs should also be legal HP BASIC programs since processing an illegal program may have unpredictable results.

Should you make an error in entering the file names simply continue with the next name. When all the file names have been entered you will be asked to make any corrections to those names that you wish. This is done by entering the number printed next to the file when they are listed on the CRT, a comma and the corrected file name. For instance, you wished to enter: DATE3. You entered instead: DATR3. On the CRT the files are listed:

```
FILE #1    DATE1
FILE #2    DATE2
FILE #3    DATR3
FILE #4    DATE4
```

When asked to make corrections you would simply enter: 3,DATE3. When you press CONT the corrected list of files will be listed. When asked to make further corrections, if there are none, just press CONT.

When the file name entry has been completed you will be asked to enter the mass storage device code (e.g. T15, for the primary tape drive, or F8, for a flexible disk drive). Finally, you will be requested to indicate whether or not you are using perforated paper.

Once the above information has been entered the program will begin to cross reference the files chosen. As the lines of the program being processed are read in they will be displayed as an indication to the user of how the cross referencing is progressing. When the end of a program is reached (or the beginning of a subprogram) then the word "OUTPUT" will be displayed. At this time, the cross reference will be printed on the hard copy printer. When the printing is complete the next file (or subprogram) will be processed, until all files have been completed. As each file is completed a message will be printed on the CRT indicating its completion. When all files are complete a message indicating that the program is done will be printed and the program will stop.

The reference listing will be printed out with the referenced name first followed by all line numbers of lines which refer to that name. For example:

```
Variable_name     140    1600    1610    1650    1880    2450    2900    2960
                         3010    4055    10210
```

Or a variable which is a string array:

```
String_array$(*)    1020   3025   3035
```

In the case of jump targets the format is slightly different. First, the line number of the referenced line will appear followed by the label (if any) and then the list of references (which may be to either the line number or the label). For example:

```
4430 Label:       220    1300    1543    1620    1630    1650    1710    2020
                         3100
```

Should the program contain a reference to a label which is not defined the cross reference will flag that label. All references to that label will be listed. This problem should be corrected before running your program. References to LINE NUMBERS which do not exist will NOT be flagged. In order to catch this error it would be necessary to either keep a list of ALL line numbers used in a program, which is too expensive in storage, or to make a second pass over the program, which is too expensive in time.

# CROSS

## PROGRAM TO CROSS REFERENCE BASIC PROGRAMS STORED AS DATA

```
Comment: Table declarations: these values must be changed to

         alter program for smaller or larger memory

40    INTEGER Lk_up_tbl(150,5)      !Comment: Look up table to find
         slot in Ref_table corresponding to name in Name_table$
50    DIM Name_table$(150,5)[18]    !Comment: Table to hold names
         encountered
60    INTEGER Ref_table(6:355,5)      !Comment: Table to hold line
         numbers of lines referencing names in Name_table$
70    INTEGER Block_size            !Comment: Size of a block; second
         dimension of Name_table$ and Lk_up_tbl
80    INTEGER Slot_size             !Comment: Size of slot; second
         dimension of Ref_table
90    Block_size=Slot_size=5

Comment: Declarations for accessing files

110   INTEGER Buscode1             !Comment: Bus code of printer (-1
         if none)
120   INTEGER File_counter         !Comment: Loop counter
130   INTEGER File_missing         !Comment: Flag for missing file
140   INTEGER File_pointer         !Comment: Index into File_name$(*)
150   INTEGER Line_count           !Comment: Number of lines printed
         on current page
160   INTEGER Lines_per_page       !Comment: number of lines printed
         on a page
170   INTEGER Max_files            !Comment: maximum number of files
         cross ref.
180   INTEGER Num_files            !Comment: Number of files to be
         cross ref.
190   INTEGER Page                 !Comment: Current page being
         printed
200   INTEGER Perforated           !Comment: =1 if paper perforated;
         else =0
210   INTEGER Printercode          !Comment: Select code of printer


230   DIM Answer$[20]              !Comment: Response to a question
240   DIM Device$[3]               !Comment: Mass storage device code
250   DIM File$[6]                 !Comment: Name of file currently
         in process
260   DIM File_name$(1:20)[6]      !Comment: Array to hold file names
         to be cross referenced

Comment: Initialize variables
```

```
CROSS


280   Max_files=20
290   Lines_per_page=50
300   Buscode1=-1

Comment: Program heading, set printer

320   PRINTER IS 16
330   PRINT PAGE,LIN(2),SPA(25);"CROSS REFERENCE",LIN(2)
340   INPUT "ENTER PRINTER SELECT CODE (0 for internal thermal line
         printer)",Printercode
350   IF (Printercode<>0) AND (Printercode<>16) THEN 370
360   GOTO 440
370   INPUT "ARE YOU USING HP-IB?  (YES/NO)",Answer$
380   IF Answer$[1,1]="Y" THEN 420
390   IF Answer$[1,1]="N" THEN 440
400   BEEP
410   GOTO 370
420   INPUT "ENTER PRINTER HP-IB BUS CODE NUMBER",Buscode1

Comment: Enter the names of files to be processed

440   INPUT "ENTER: number of files to be cross referenced (maximum
         20)",Num_files
450   IF (Num_files>0) AND (Num_files<=Max_files) THEN 480
460   BEEP
470   GOTO 440
480   PRINT PAGE;"The following files are to be cross referenced:",LIN(
         2)
490      FOR File_counter=1 TO Num_files
500      DISP "ENTER FILE NAME:";
510      LINPUT "",File_name$(File_counter)
520      PRINT SPA(10);"FILE #";File_counter;TAB(23);File_name$(
            File_counter)
530      NEXT File_counter

Comment: Make corrections if necessary

550   File_pointer=0
560   INPUT "ENTER A CORRECTION:(FILE #,CORRECTED NAME) OR (PRESS
         CONT IF NO CORRECTIONS)",File_pointer,File_name$(
         File_pointer)
570   IF File_pointer=0 THEN 650
580   PRINT PAGE;"The following files are to be cross referenced:",LIN(
         2)
590      FOR File_counter=1 TO Num_files
600      PRINT SPA(10);"FILE #";File_counter;TAB(23);File_name$(
            File_counter)
610      NEXT File_counter
620   File_pointer=0
630   GOTO 560
```

```
Comment: Enter mass storage device code

650  LINPUT "ENTER: MASS STORAGE DEVICE CODE  (eg. T14 or F8)",
        Device$
660  ON ERROR GOTO Mass_error
670  MASS STORAGE IS ":"&Device$
680  GOTO 730
690 Mass_error:     BEEP
700     DISP "MASS STORAGE DEVICE NOT FOUND; PLEASE REENTER"
710     WAIT 2000
720     GOTO 650
730  OFF ERROR

Comment: Set perforated/non-perforated paper flag

750  LINPUT "LISTING REFERENCES ON PERFORATED PAPER? (YES/NO)",
        Answer$
760  IF (Answer$[1,1]="N") OR (Answer$[1,1]="n") THEN 800
770  IF (Answer$[1,1]="Y") OR (Answer$[1,1]="y") THEN 820
780  BEEP
790  GOTO 730
800  Perforated=0
810  GOTO 840
820  Perforated=1

Comment: Procees each file

840   Line_count=69
850   GOSUB Set_printer
860      FOR File_pointer=1 TO Num_files
870      File$=File_name$(File_pointer)
880      DISP File$
890      GOSUB Header
900      GOSUB Xref
910      NEXT File_pointer
920   IF Perforated THEN PRINT PAGE
930   IF NOT Perforated THEN PRINT LIN(70-Line_count),RPT$("_",80)
940   PRINTER IS 16
950   PRINT LIN(1);"The programs have been cross referenced"
960   PRINT "                    DONE"
970   DISP ""
980   STOP

Comment: Print heading to cross reference output

1000 Header:  IF Perforated THEN PRINT PAGE
1010          IF NOT Perforated THEN PRINT LIN(70-Line_count),RPT$(
                 "_",80)
1020          IF NOT Perforated THEN PRINT LIN(2)
```

CROSS

```
1030          IF Printercode=0 THEN PRINT LIN(4),TAB(35);CHR$(27)&"&
                 k1S",File$&CHR$(27)&"&k0S",LIN(2)
1040          IF Printercode<>0 THEN PRINT LIN(4),TAB(35);File$,LIN(2)
1050          Line_count=10
1060          Page=1
1070          RETURN
```

Comment: Handle lines printed, pages

```
1090 Lines:IF Line_count>10+Lines_per_page THEN GOSUB New_page
1100       RETURN
1110 New_page:Page=Page+1
1120          IF Perforated THEN PRINT PAGE
1130          IF NOT Perforated THEN PRINT LIN(70-Line_count),RPT$(
                 "_",80),LIN(3)
1140          PRINT LIN(4),TAB(Page MOD 2*55+5);File$,LIN(2)
1150          Line_count=10
1160          RETURN
```

Comment: Set printer to hard copy printer

```
1180 Set_printer:          !
1190    IF Buscode1=-1 THEN PRINTER IS Printercode
1200    IF Buscode1>-1 THEN PRINTER IS Printercode,Buscode1
1210    RETURN
```

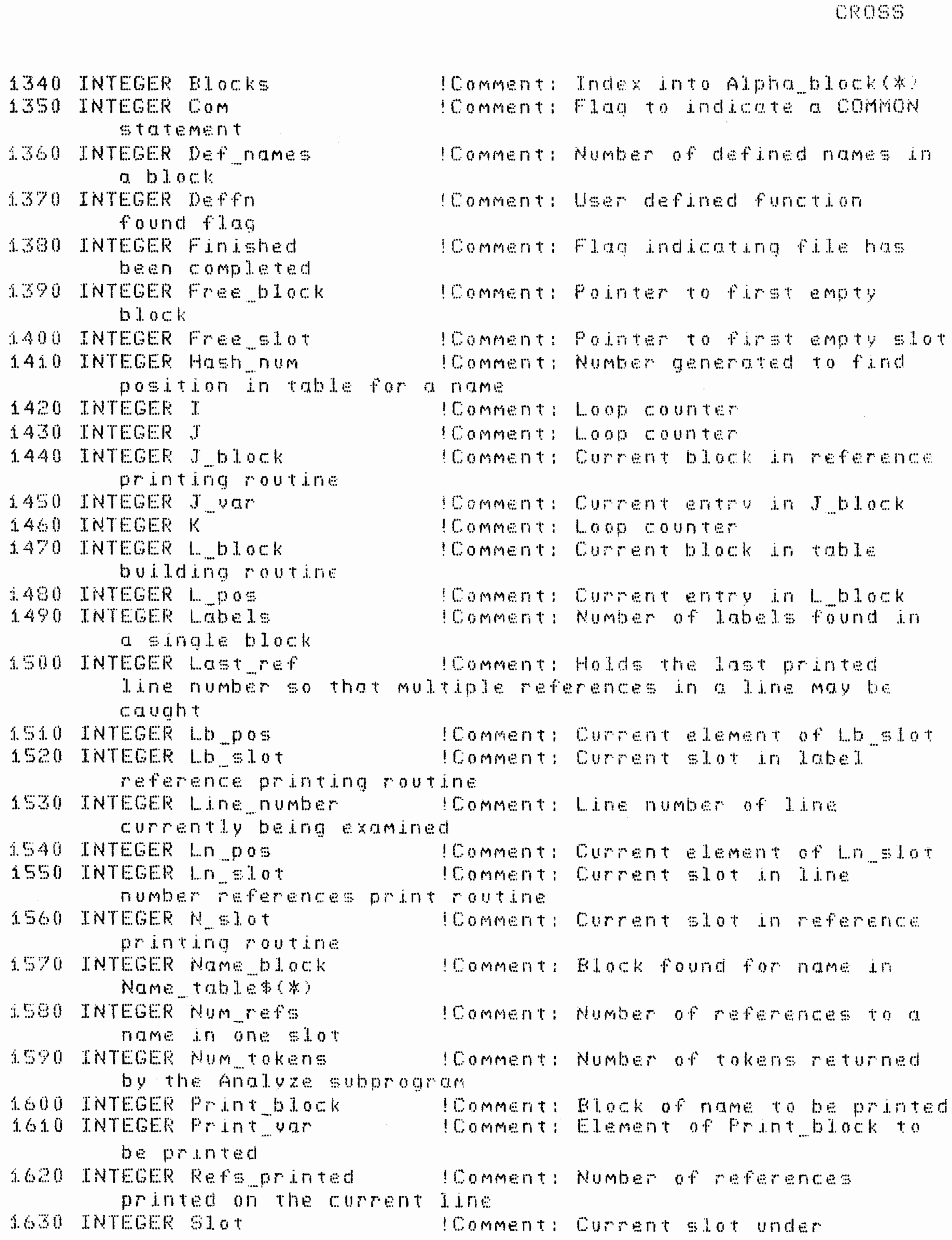
CROSS REFERENCE ONE FILE

```
1230 Xref:      !
1240 PRINTER IS 16
```

Comment: VARIABLE DEFINITIONS

```
1260 COM Token$(80)[20]
```

```
1280 DIM A$[165]                 !Comment: Line of code to be
        processed
1290 DIM Alpha_block(40)         !Comment: Temporary holder for
        blocks of one name
1300 INTEGER Alpha_sum           !Comment: Number of slots used to
        hold referencesfor a single name
1310 INTEGER Bl_block            !Comment: Current block in the bad
        label routine
1320 INTEGER Block               !Comment: Current block being
        examined
1330 INTEGER Block_counter       !Comment: Counter for blocks
```



```
1340 INTEGER Blocks              !Comment: Index into Alpha_block(*)
1350 INTEGER Com                 !Comment: Flag to indicate a COMMON
     statement
1360 INTEGER Def_names           !Comment: Number of defined names in
     a block
1370 INTEGER Deffn               !Comment: User defined function
     found flag
1380 INTEGER Finished            !Comment: Flag indicating file has
     been completed
1390 INTEGER Free_block          !Comment: Pointer to first empty
     block
1400 INTEGER Free_slot           !Comment: Pointer to first empty slot
1410 INTEGER Hash_num            !Comment: Number generated to find
     position in table for a name
1420 INTEGER I                   !Comment: Loop counter
1430 INTEGER J                   !Comment: Loop counter
1440 INTEGER J_block             !Comment: Current block in reference
     printing routine
1450 INTEGER J_var               !Comment: Current entrv in J_block
1460 INTEGER K                   !Comment: Loop counter
1470 INTEGER L_block             !Comment: Current block in table
     building routine
1480 INTEGER L_pos               !Comment: Current entry in L_block
1490 INTEGER Labels              !Comment: Number of labels found in
     a single block
1500 INTEGER Last_ref            !Comment: Holds the last printed
     line number so that multiple references in a line may be
     caught
1510 INTEGER Lb_pos              !Comment: Current element of Lb_slot
1520 INTEGER Lb_slot             !Comment: Current slot in label
     reference printing routine
1530 INTEGER Line_number         !Comment: Line number of line
     currently being examined
1540 INTEGER Ln_pos              !Comment: Current element of Ln_slot
1550 INTEGER Ln_slot             !Comment: Current slot in line
     number references print routine
1560 INTEGER N_slot              !Comment: Current slot in reference
     printing routine
1570 INTEGER Name_block          !Comment: Block found for name in
     Name_table$(*)
1580 INTEGER Num_refs            !Comment: Number of references to a
     name in one slot
1590 INTEGER Num_tokens          !Comment: Number of tokens returned
     by the Analyze subprogram
1600 INTEGER Print_block         !Comment: Block of name to be printed
1610 INTEGER Print_var           !Comment: Element of Print_block to

     be printed
1620 INTEGER Refs_printed        !Comment: Number of references
     printed on the current line
1630 INTEGER Slot                !Comment: Current slot under
```

CROSS

```
        examination
1640 INTEGER Subpro                   !Comment: SUB statement encountered
        flag
1650 INTEGER X_block                  !Comment: Current block for setting
        up Alpha_block(*)

Comment: Initialization

1670 GOSUB Setup
1680 GOTO 1770


Comment: Set up arrays

1700 Setup:MAT Ref_table=ZER
1710       MAT Lk_up_tbl=ZER
1720       Free_slot=6
1730       Free_block=62
1740       Finished=0
1750       RETURN


 Comment: Pick file to cross-reference

1770  ON END #1 GOTO File_done
1780  ON ERROR GOTO File_error
1790  ASSIGN #1 TO File$&":"&Device$,File_missing


1810  IF NOT File_missing THEN 1910
1820     PRINTER IS 16
1830     PRINT File$&":"&Device$;" NOT FOUND"
1840     GOSUB Set_printer
1850     GOTO 2790
1860 File_error:  BEEP
1870  PRINTER IS 16
1880  PRINT "ERROR ENCOUNTERED IN '";File$;"' - CONTINUING WITH NEXT
         FILE"
1890  GOSUB Set_printer
1900  GOTO 2790
1910  GOSUB Setup
1920  GOSUB Set_printer
1930  PRINT "     "&File$,LIN(2),"    MAIN PROGRAM"
1940  Line_count=Line_count+3
1950  GOSUB Lines
1960  PRINTER IS 16


 Comment: Read one line of program and analyze
```

```
1980  READ #1;A$
1990  IF LEN(A$)<81 THEN DISP A$
2000  IF LEN(A$)>80 THEN DISP A$[1,80]
2010     A$=A$&"@"
2020     CALL Analyze(A$,Line_number,Num_tokens)
2030     IF Line_number>=0 THEN 2090
2040     IF (Token$(0)="DEF") AND (A$[POS(A$,")")+1;1]="=") THEN 2060
2050     GOTO Out
2060     A$[POS(A$,"DEF");1]=" "
2070     CALL Analyze(A$,Line_number,Num_tokens)
```

Comment: Build Name, Look_up and Reference tables

```
2090 IF Num_tokens<1 THEN 1980
2100    FOR I=0 TO Num_tokens-1
2110    IF Token$(I)=" " THEN 2380
2120    Hash_num=NUM(Token$(I)[1,1])-NUM("A")
2130    IF Hash_num<0 THEN Hash_num=VAL(Token$(I)) DIV 1000+30
2140    IF POS(Token$(I),"FN")<>0 THEN Hash_num=26
2150    IF POS(Token$(I),"!")=0 THEN 2190
2160    Hash_num=29
2170    Token$(I)=Token$(I)[1,LEN(Token$(I))-1]
2180    GOTO 2360
2190    IF POS(Token$(I),"#")=0 THEN 2230
2200     Hash_num=27
2210     Token$(I)=Token$(I)[1,LEN(Token$(I))-1]
2220     GOTO 2360
2230    IF POS(Token$(I),":")=0 THEN 2360
2240     Hash_num=28
2250     IF POS(Token$(I),"::")=0 THEN 2360
2260     Token$(I)=Token$(I)[1,LEN(Token$(I))-1]
2270     GOSUB Find_slot
2280     L_block=Name_block
2290     L_pos=J
2300     Token$(I)=VAL$(Line_number)
2310     Hash_num=VAL(Token$(I)) DIV 1000+30
2320     GOSUB Find_slot
2330     Ref_table(Slot,1)=100*L_block+L_pos
2340     IF Ref_table(Slot,0)=0 THEN Ref_table(Slot,0)=1
2350     GOTO 2380
2360 GOSUB Find_slot
2370 GOSUB Put_ref
2380 NEXT I
2390 GOTO 1980
```

Comment: Output Cross-reference

```
2410 File_done: Finished=1
```

CROSS

```
2420 Out: DISP "OUTPUT"
2430 GOSUB Set_printer
2440 PRINT LIN(1),"    COMMON VARIABLES:"
2450 Line_count=Line_count+2
2460 GOSUB Lines
2470 Block=29
2480 Com=1
2490 GOSUB Output
2500 Com=0
2510 PRINT LIN(1),"    VARIABLES:"
2520 Line_count=Line_count+2
2530 GOSUB Lines
2540 FOR Block=0 TO 25
2550 GOSUB Output
2560 NEXT Block
2570 Block=26
2580 PRINT LIN(1),"    USER DEFINED FUNCTIONS:"
2590 Line_count=Line_count+2
2600 GOSUB Lines
2610 GOSUB Output
2620 Block=27
2630 PRINT LIN(1),"    SUB PROGRAMS:"
2640 Line_count=Line_count+2
2650 GOSUB Lines
2660 GOSUB Output
2670 PRINT LIN(1),"    JUMP TARGETS:"
2680 Line_count=Line_count+2
2690 GOSUB Lines
2700 FOR Block=30 TO 61
2710 GOSUB Output
2720 NEXT Block
2730 GOSUB Badlabels
2740 IF Finished THEN 2760
2750 GOTO Next_mod
2760 PRINTER IS 16
2770 PRINT LIN(1),File$;" HAS BEEN CROSS REFERENCED"
2780 GOSUB Set_printer
2790 RETURN


 Comment: Set up for the next module

2810 Next_mod:GOSUB Setup
2820 PRINT LIN(1)
2830 Line_count=Line_count+2
2840 GOSUB Lines
2850 IF Token$(0)="DEF" THEN Deffn=1
2860 IF Token$(0)="SUB" THEN Subpro=1
2870 A$[POS(A$,Token$(0));1]=" "
2880 CALL Analyze(A$,Line_number,Num_tokens)
```

```
2890 IF NOT Deffn THEN 2930
2900 IF POS(Token$(0),"(") THEN PRINT SPA(4);CHR$(132);Token$(0)[1,
        POS(Token$(0),"(")-1];CHR$(128)
2910 IF NOT POS(Token$(0),"(") THEN PRINT SPA(4);CHR$(132);Token$(0);
        CHR$(128)
2920 GOTO 2950
2930 IF POS(Token$(0),"(") THEN PRINT SPA(4);CHR$(132);"SUB PROGRAM ";
        Token$(0)[1,POS(Token$(0),"(")-1];CHR$(128)
2940 IF NOT POS(Token$(0),"(") THEN PRINT SPA(4);CHR$(132);"SUB
        PROGRAM ";Token$(0);CHR$(128)
2950 Line_count=Line_count+1
2960 GOSUB Lines
2970 Token$(0)=" "
2980 Deffn=Subpro=0
2990 PRINTER IS 16
3000 GOTO 2090
```

```
 Comment: Find the name in the name table
```

```
3020 Find_slot: !
3030 Name_block=Hash_num
3040 Def_names=Lk_up_tbl(Name_block,0)
3050 IF Def_names>Block_size THEN Def_names=Block_size
3060    FOR J=1 TO Def_names
3070    IF Name_table$(Name_block,J)=Token$(I) THEN Found_slot
3080    NEXT J
3090 IF Lk_up_tbl(Name_block,0)=Block_size THEN New_block
3100 IF Lk_up_tbl(Name_block,0)>Block_size THEN Next_block
3110 Lk_up_tbl(Name_block,0)=Lk_up_tbl(Name_block,0)+1
3120 Slot=Lk_up_tbl(Name_block,Lk_up_tbl(Name_block,0))=Free_slot
3130 Free_slot=Free_slot+1
3140 Name_table$(Name_block,Lk_up_tbl(Name_block,0))=Token$(I)
3150 GOTO 3270
3160 Next_block: Name_block=Lk_up_tbl(Name_block,0)
3170 GOTO 3040
3180 New_block: Name_block=Lk_up_tbl(Name_block,0)=Free_block
3190 Lk_up_tbl(Name_block,0)=1
3200 Free_block=Free_block+1
3210 Slot=Lk_up_tbl(Name_block,1)=Free_slot
3220 Free_slot=Free_slot+1
3230 Name_table$(Name_block,1)=Token$(I)
3240 J=1
3250 GOTO 3270
3260 Found_slot: Slot=Lk_up_tbl(Name_block,J)
3270 RETURN
```

```
 Comment: Put reference in reference table
```

```
3290 Put_ref: !
3300 IF Ref_table(Slot,0)>=Slot_size THEN Next_slot
3310 IF (Hash_num>29) AND (Ref_table(Slot,0)=0) THEN Ref_table(Slot,0)
        =1
3320 Ref_table(Slot,0)=Ref_table(Slot,0)+1
3330 Ref_table(Slot,Ref_table(Slot,0))=Line_number
3340 GOTO 3420
3350 Next_slot: !
3360 IF Ref_table(Slot,0)>Slot_size THEN 3400
3370 Slot=Ref_table(Slot,0)=Free_slot
3380 Free_slot=Free_slot+1
3390 GOTO 3320
3400 Slot=Ref_table(Slot,0)
3410 GOTO 3300
3420 RETURN


 Comment: Subroutine to print reference table

3440 Output: !
3450 X_block=Block
3460 Alpha_sum=0
3470 Blocks=1
3480 Alpha_block(1)=Block
3490 IF Lk_up_tbl(X_block,0)<=Block_size THEN Summed
3500 Alpha_sum=Alpha_sum+Block_size
3510 Blocks=Blocks+1
3520 X_block=Lk_up_tbl(X_block,0)
3530 Alpha_block(Blocks)=X_block
3540 GOTO 3490
3550 Summed:Alpha_sum=Alpha_sum+Lk_up_tbl(X_block,0)
3560 Block_counter=1
3570 FOR I=1 TO Alpha_sum
3580    J_block=1
3590    Print_block=Alpha_block(Block_counter)
3600    Print_var=1
3610       FOR J=1 TO Alpha_sum
3620       J_var=J MOD Block_size
3630       IF J_var=0 THEN J_var=Block_size
3640       IF Block>=30 THEN Num_order
3650       IF Name_table$(Print_block,Print_var)<=Name_table$(
             Alpha_block(J_block),J_var) THEN Nextj
3660       GOTO 3680
3670 Num_order: IF VAL(Name_table$(Print_block,Print_var))<=VAL(
        Name_table$(Alpha_block(J_block),J_var)) THEN Nextj
3680       Print_block=Alpha_block(J_block)
3690       Print_var=J_var
3700 Nextj:IF J MOD Block_size=0 THEN J_block=J_block+1
3710       NEXT J
3720    IF Name_table$(Print_block,Print_var)=CHR$(126) THEN RETURN
```

```
3730      PRINT SPA(4);Name_table$(Print_block,Print_var);
3740      IF I MOD Block_size=0 THEN Block_counter=Block_counter+1
3750      GOSUB List_ref
3760 NEXT I
3770 RETURN


  Comment: List reference lines

3790 List_ref:    IF NOT Com THEN 3900
3800      PRINT TAB(30);
3810      PRINT USING 4100;Ref_table(Lk_up_tbl(Print_block,Print_var),1)
3820      Refs_printed=1
3830      Last_ref=0
3840      Token$(I)=Name_table$(Print_block,Print_var)
3850      Hash_num=NUM(Token$(I)[1,1])-NUM("A")
3860      GOSUB Find_slot
3870      Name_table$(Name_block,J)=CHR$(126)
3880      N_slot=Slot
3890      GOTO 3930
3900      Refs_printed=0
3910      Last_ref=0
3920      N_slot=Lk_up_tbl(Print_block,Print_var)
3930      IF (Ref_table(N_slot,1)=0) OR (Block<30) THEN 3960
3940         GOSUB Print_label
3950         GOTO 4170
3960      IF Com THEN 3980
3970      PRINT TAB(30);
3980      Num_refs=Ref_table(N_slot,0)
3990      IF Num_refs>Slot_size THEN Num_refs=Slot_size
4000         FOR K=1 TO Num_refs
4010         IF Ref_table(N_slot,K)=Last_ref THEN 4130
4020         IF Ref_table(N_slot,K)=0 THEN 4130
4030         Refs_printed=Refs_printed+1
4040         IF Refs_printed<9 THEN 4110
4050         PRINT
4060         Line_count=Line_count+1
4070         GOSUB Lines
4080         PRINT TAB(36);
4090         Refs_printed=2
4100         IMAGE #,5D,X
4110         PRINT USING 4100;Ref_table(N_slot,K)
4120         Last_ref=Ref_table(N_slot,K)
4130         NEXT K
4140      IF Ref_table(N_slot,0)<=Slot_size THEN 4170
4150         N_slot=Ref_table(N_slot,0)
4160         GOTO 3980
4170      PRINT
4180      Line_count=Line_count+1
4190      GOSUB Lines
```

CROSS

```
4200    Name_table$(Print_block,Print_var)="~"
4210    IF Block>=30 THEN Name_table$(Print_block,Print_var)="32767"
4220 RETURN
4230 STOP
4240 Print_label: !
4250       GOSUB Find_label
4260       GOSUB Ord_lbl_refs
4270       RETURN
4280 Find_label:  !
4290     L_block=Ref_table(N_slot,1) DIV 100
4300     L_pos=Ref_table(N_slot,1) MOD 100
4310     IMAGE #,15A,X
4320     PRINT TAB(11);
4330     PRINT USING 4310;Name_table$(L_block,L_pos)
4340     Name_table$(L_block,L_pos)="~"
4350     PRINT TAB(30);
4360 RETURN
4370 Ord_lbl_refs:!
4380     Refs_printed=0
4390     Last_ref=0
4400     Ln_slot=N_slot
4410     Ln_pos=2
4420     Lb_slot=Lk_up_tbl(L_block,L_pos)
4430     Lb_pos=1
4440     IF Ref_table(Lb_slot,Lb_pos)=0 THEN Run_ln
4450     IF Ref_table(Ln_slot,Ln_pos)=0 THEN Run_lb
4460     IF Last_ref=MIN(Ref_table(Ln_slot,Ln_pos),Ref_table(Lb_slot,
            Lb_pos)) THEN 4560
4470     IF Refs_printed<8 THEN 4530
4480     Refs_printed=1
4490     PRINT
4500     Line_count=Line_count+1
4510     GOSUB Lines
4520     PRINT TAB(36);
4530     PRINT USING 4100;MIN(Ref_table(Ln_slot,Ln_pos),Ref_table(
            Lb_slot,Lb_pos))
4540     Last_ref=MIN(Ref_table(Ln_slot,Ln_pos),Ref_table(Lb_slot,
            Lb_pos))
4550     Refs_printed=Refs_printed+1
4560     IF Ref_table(Ln_slot,Ln_pos)>Ref_table(Lb_slot,Lb_pos) THEN
            4620
4570     Ln_pos=Ln_pos+1
4580     IF Ln_pos<=Slot_size THEN 4660
4590     Ln_pos=1
4600     Ln_slot=Ref_table(Ln_slot,0)
4610     GOTO 4660
4620     Lb_pos=Lb_pos+1
4630     IF Lb_pos<=Slot_size THEN 4660
4640     Lb_pos=1
4650     Lb_slot=Ref_table(Lb_slot,0)
```

```
4660        GOTO 4440
4670 Run_ln:       !
4680        IF Ref_table(Ln_slot,Ln_pos)=0 THEN Back
4690        IF Last_ref=Ref_table(Ln_slot,Ln_pos) THEN 4790
4700        IF Refs_printed<8 THEN 4760
4710        Refs_printed=1
4720        PRINT
4730        Line_count=Line_count+1
4740        GOSUB Lines
4750        PRINT TAB(36)
4760        PRINT USING 4100;Ref_table(Ln_slot,Ln_pos)
4770        Last_ref=Ref_table(Ln_slot,Ln_pos)
4780        Refs_printed=Refs_printed+1
4790        Ln_pos=Ln_pos+1
4800        IF Ln_pos<=Slot_size THEN 4680
4810        IF Ref_table(Ln_slot,0)=Slot_size THEN Back
4820        Ln_slot=Ref_table(Ln_slot,0)
4830        Ln_pos=1
4840        GOTO 4680
4850 Run_lb:       !
4860        IF Ref_table(Lb_slot,Lb_pos)=0 THEN Back
4870        IF Last_ref=Ref_table(Lb_slot,Lb_pos) THEN 4970
4880        IF Refs_printed<8 THEN 4940
4890        PRINT
4900        Line_count=Line_count+1
4910        GOSUB Lines
4920        PRINT TAB(36);
4930        Refs_printed=1
4940        PRINT USING 4100;Ref_table(Lb_slot,Lb_pos)
4950        Last_ref=Ref_table(Lb_slot,Lb_pos)
4960        Refs_printed=Refs_printed+1
4970        Lb_pos=Lb_pos+1
4980        IF Lb_pos<=Slot_size THEN 4860
4990        IF Ref_table(Lb_slot,0)=Slot_size THEN Back
5000        Lb_slot=Ref_table(Lb_slot,0)
5010        Lb_pos=1
5020        GOTO 4860
5030 Back:       RETURN
```

Comment: Check for references to nonexistent labels

```
5050 Badlabels: !
5060      Bl_block=28
5070      Labels=Lk_up_tbl(Bl_block,0)
5080      IF Lk_up_tbl(Bl_block,0)>Block_size THEN Labels=Block_size
5090         FOR M=1 TO Labels
5100         IF Name_table$(Bl_block,M)<>"~" THEN GOSUB Label_err
5110         NEXT M
5120      IF Lk_up_tbl(Bl_block,0)<=Block_size THEN GOTO 5150
5130      Bl_block=Lk_up_tbl(Bl_block,0)
```

```
5140     GOTO 5070
5150     RETURN
5160 Label_err:      !
5170     Block=28
5180     PRINT "***  Reference to undefined label ***"
5190     Line_count=Line_count+1
5200     GOSUB Lines
5210     Print_block=Bl_block
5220     Print_var=M
5230     PRINT SPA(4);"--->  ";Name_table$(Bl_block,M);
5240     GOSUB List_ref
5250     RETURN
```

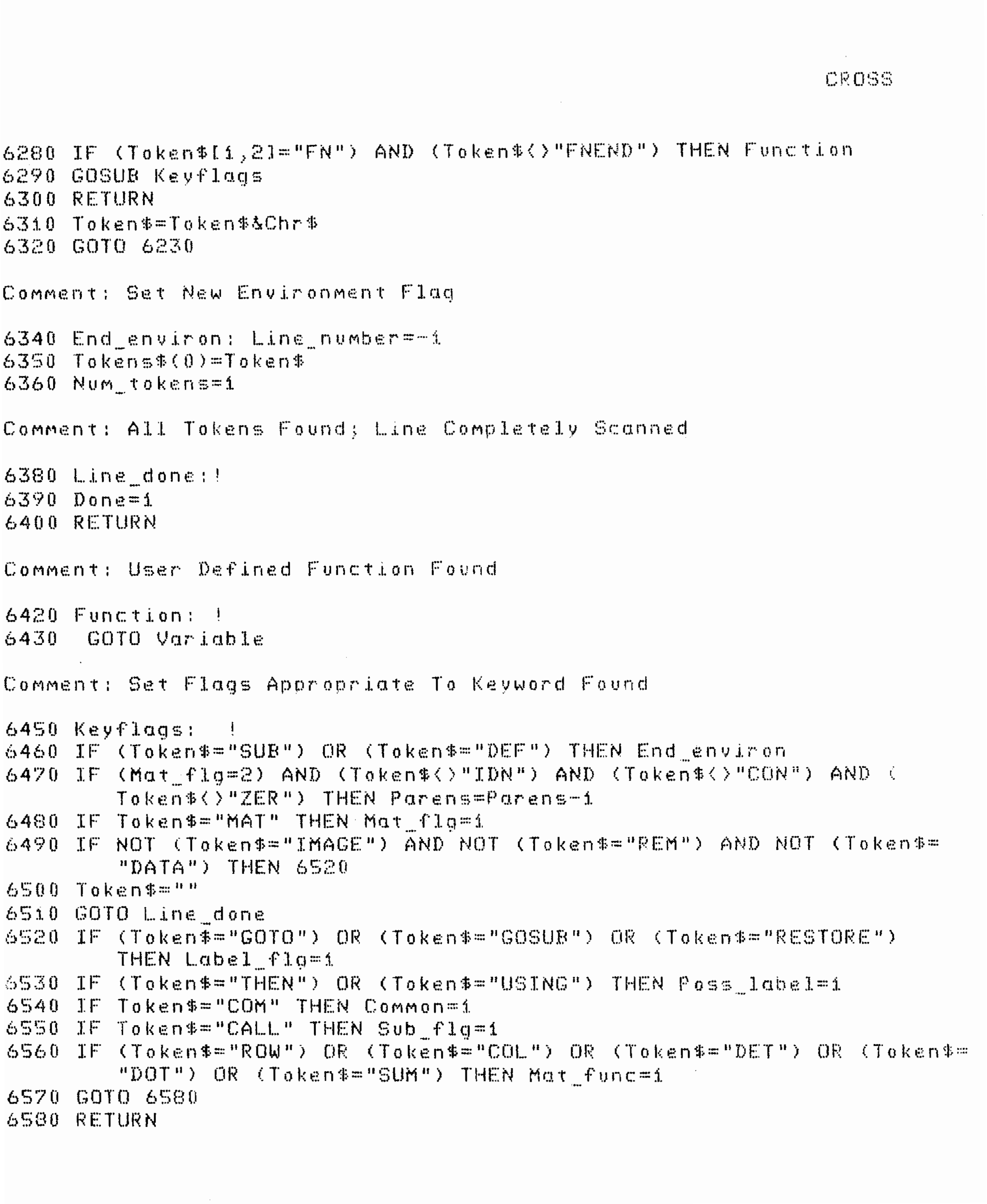
## ANALYZE ONE STATEMENT

```
5270 SUB Analyze(Line$,INTEGER Line_number,INTEGER Num_tokens)


5290 GOTO 5420
5300 COM Tokens$(80)[20]                  !Comment: Holds referenced
         objects

Comment: Declarations

5320 DIM Chr$[1]                          !Comment: Character being
         scanned
5330 INTEGER Common                       !Comment: Common statement
         flag
5340 INTEGER Done                         !Comment: Completion flag
5350 INTEGER Label_flg                    !Comment: Label flag
5360 INTEGER Mat_flg                      !Comment: MAT statement flag
5370 INTEGER Mat_func                     !Comment: Function with
         matrix argument flag
5380 INTEGER Parens                       !Comment: Parentheses
         nesting level
5390 INTEGER Pos                          !Comment: Position in line,
         of scan
5400 INTEGER Poss_label                   !Comment: Possible label
         flag
5410 INTEGER Sub_flg                      !Comment: SUB statement flag
5420 Num_tokens=0

Comment: Get Line Number

5440 Line_number=VAL(Line$[1,POS(Line$," ")-1])
5450 Pos=POS(Line$," ")

Comment: Get Tokens
```

```
5470 Pos=1
5480 GOSUB Grab_chr
5490 GOSUB Token
5500 Token$=""
5510 IF Done THEN SUBEXIT
5520 GOTO 5490

Comment: Find One Token

5540 Token: !
5550 GOTO 5590
5560 DEF FNVar(X$)=(X$)="a") AND (X$<="z") OR (X$)="0") AND (X$<="9")
        OR (X$="_") OR (X$="$") OR (X$=":")
5570 DEF FNCaps(X$)=(X$)="A") AND (X$<="Z")
5580 DEF FNDigits(X$)=(X$)="0") AND (X$<="9")
5590 IF (Line$[Pos;1]="!") OR (Pos>LEN(Line$)) THEN Line_done
5600 IF Done OR (Chr$="!") THEN Line_done
5610 IF FNCaps(Chr$) AND NOT FNDigits((Line$[Pos-1;1])) THEN
        Key_or_var
5620 IF Chr$=CHR$(34) THEN Run_quote
5630 IF Chr$=";" THEN Poss_label=0
5640 IF (Label_flg OR Poss_label) AND FNDigits(Chr$) THEN Num_label
5650 IF Chr$="(" THEN Parens=Parens+1
5660 IF Chr$=")" THEN Parens=Parens-1
5670 IF Chr$=")" THEN Mat_func=0
5680 IF (Chr$="=") AND (Mat_flg=1) THEN Mat_flg=2
5690 GOSUB Grab_chr
5700 GOTO 5600

Comment: Get One Character

5720 Grab_chr: Pos=Pos+1
5730 IF Pos>LEN(Line$) THEN Fin
5740 Chr$=Line$[Pos;1]
5750 GOTO 5770
5760 Fin: Done=1
5770 RETURN

Comment: Ignore Quoted Strings

5790 Run_quote: GOSUB Grab_chr
5800 IF Done THEN Line_done
5810 IF Chr$<>CHR$(34) THEN GOTO 5790
5820 GOSUB Grab_chr
5830 GOTO 5600

Comment: Find A LineNumber Used As A Jump Target

5850 Num_label: !
5860 Token$=Token$&Chr$
```

CROSS

```
5870 GOSUB Grab_chr
5880 IF Done THEN Line_done
5890 IF FNDigits(Chr$) THEN GOTO 5860
5900 Tokens$(Num_tokens)=Token$
5910 Num_tokens=Num_tokens+1
5920 RETURN

Comment: Keywords And Variables

5940 Key_or_var: !
5950 Token$=Chr$
5960 GOSUB Grab_chr
5970 IF Done THEN Line_done
5980 IF FNCaps(Chr$) THEN Key_word

Comment: Find A Variable

6000 Variable:!
6010 IF FNVar(Chr$) THEN 6150
6020 IF (Chr$="(") AND (Token$[1,2]<>"FN") AND NOT Sub_flg THEN
        Token$=Token$&"(*)"
6030 IF POS(Token$,":") THEN Token$=Token$&":"
6040 IF Common THEN Token$=Token$&"!"
6050 IF Sub_flg THEN Token$=Token$&"#"
6060 Sub_flg=0
6070 IF Mat_flg AND (Parens=0) AND (Token$[LEN(Token$);1]<>")") THEN
        Token$=Token$&"(*)"
6080 IF Mat_func THEN Token$=Token$&"(*)"
6090 IF Poss_label AND ((Chr$=";") OR (Chr$="@") OR (Chr$=" ") OR (
        Chr$="!")) THEN Token$=Token$&":"
6100 Poss_label=0
6110 IF Label_flg THEN Token$=Token$&":"
6120 Tokens$(Num_tokens)=Token$
6130 Num_tokens=Num_tokens+1
6140 RETURN
6150 Token$=Token$&Chr$
6160 GOSUB Grab_chr
6170 IF Done THEN Line_done
6180 GOTO 6010

Comment: Find One Keyword

6200 Key_word: !
6210 Poss_label=0
6220 Token$=Token$&Chr$
6230 GOSUB Grab_chr
6240 IF NOT Done THEN 6270
6250   Token$=""
6260   GOTO Line_done
6270 IF FNCaps(Chr$) THEN 6310
```



```
6280 IF (Token$[1,2]="FN") AND (Token$<>"FNEND") THEN Function
6290 GOSUB Keyflags
6300 RETURN
6310 Token$=Token$&Chr$
6320 GOTO 6230
```

Comment: Set New Environment Flag

```
6340 End_environ: Line_number=-1
6350 Tokens$(0)=Token$
6360 Num_tokens=1
```

Comment: All Tokens Found; Line Completely Scanned

```
6380 Line_done:!
6390 Done=1
6400 RETURN
```

Comment: User Defined Function Found

```
6420 Function: !
6430  GOTO Variable
```

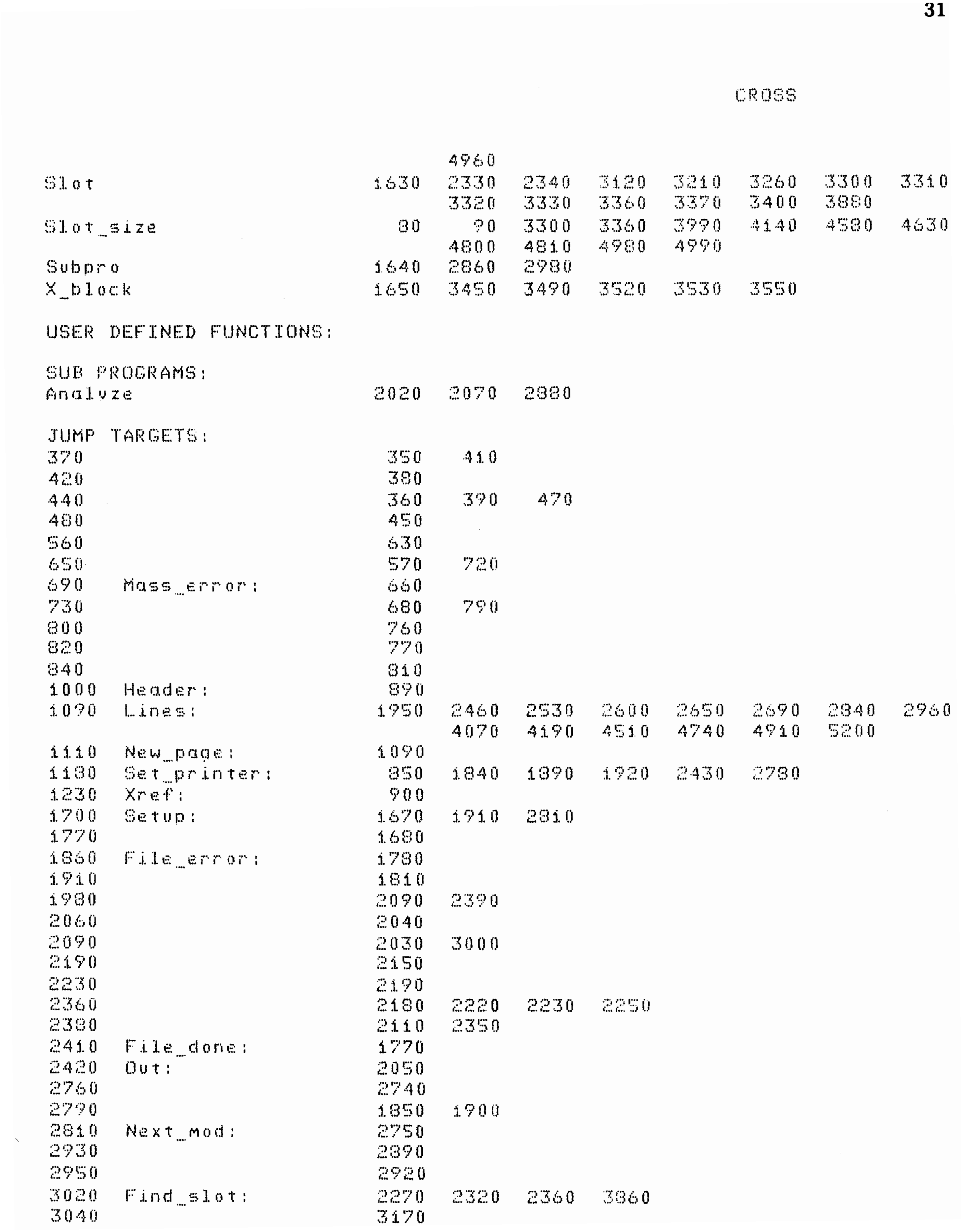
Comment: Set Flags Appropriate To Keyword Found

```
6450 Keyflags:  !
6460 IF (Token$="SUB") OR (Token$="DEF") THEN End_environ
6470 IF (Mat_flg=2) AND (Token$<>"IDN") AND (Token$<>"CON") AND (
        Token$<>"ZER") THEN Parens=Parens-1
6480 IF Token$="MAT" THEN Mat_flg=1
6490 IF NOT (Token$="IMAGE") AND NOT (Token$="REM") AND NOT (Token$=
        "DATA") THEN 6520
6500 Token$=""
6510 GOTO Line_done
6520 IF (Token$="GOTO") OR (Token$="GOSUB") OR (Token$="RESTORE")
        THEN Label_flg=1
6530 IF (Token$="THEN") OR (Token$="USING") THEN Poss_label=1
6540 IF Token$="COM" THEN Common=1
6550 IF Token$="CALL" THEN Sub_flg=1
6560 IF (Token$="ROW") OR (Token$="COL") OR (Token$="DET") OR (Token$=
        "DOT") OR (Token$="SUM") THEN Mat_func=1
6570 GOTO 6580
6580 RETURN
```

# CROSS

```
CROSS

MAIN PROGRAM

COMMON VARIABLES:
Token$(*)                 1260  2040  2110  2120  2130  2140  2150  2170
                                2190  2210  2230  2250  2260  2300  2310
                                2350  2360  2370  2900  2910  2930  2940
                                2970  3070  3140  3230  3840  3850

VARIABLES:
A$                        1280  1980  1990  2000  2010  2020  2040  2060
                                2070  2870  2880
Alpha_block(*)            1290  3480  3530  3590  3650  3670  3680
Alpha_sum                 1300  3460  3500  3550  3570  3610
Answer$                    230   370   380   390   750   760   770
Bl_block                  1310  5060  5070  5080  5100  5120  5130  5210
                                5230
Block                     1320  2470  2540  2560  2570  2620  2700  2720
                                3450  3480  3640  3930  4210  5170
Block_counter             1330  3560  3590  3740
Block_size                  70    90  3050  3090  3100  3490  3500  3620
                                3630  3700  3740  5080  5120
Blocks                    1340  3470  3510  3530
Buscodei                   110   300   420  1190  1200
Com                       1350  2480  2500  3790  3960
Def_names                 1360  3040  3050  3060
Deffn                     1370  2850  2890  2980
Device$                    240   650   670  1790  1830
File$                      250   870   880  1030  1040  1140  1790  1830
                                1880  1930  2770
File_counter               120   490   510   520   530   590   600   610
File_missing               130  1790  1810
File_name$(*)              260   510   520   560   600   870
File_pointer               140   550   560   570   620   860   870   910
Finished                  1380  1740  2410  2740
Free_block                1390  1730  3180  3200
Free_slot                 1400  1720  3120  3130  3210  3220  3370  3380
Hash_num                  1410  2120  2130  2140  2160  2200  2240  2310
                                3030  3310  3850
I                         1420  2100  2110  2120  2130  2140  2150  2170
                                2190  2210  2230  2250  2260  2300  2310
                                2380  3070  3140  3230  3570  3740  3760
                                3840  3850
J                         1430  2290  3060  3070  3080  3240  3260  3610
                                3620  3700  3710  3870
J_block                   1440  3580  3650  3670  3680  3700
J_var                     1450  3620  3630  3650  3670  3690
K                         1460  4000  4010  4020  4110  4120  4130
L_block                   1470  2280  2330  4290  4330  4340  4420
L_pos                     1480  2290  2330  4300  4330  4340  442
```

```
CROSS


Labels                   1490   5070   5080   5090
Last_ref                 1500   3830   3910   4010   4120   4390   4460   4540
                                4690   4770   4870   4950
Lb_pos                   1510   4430   4440   4460   4530   4540   4560   4620
                                4630   4640   4860   4870   4940   4950   4970
                                4980   5010
Lb_slot                  1520   4420   4440   4460   4530   4540   4560   4650
                                4860   4870   4940   4950   4990   5000
Line_count                150    840    930   1010   1050   1090   1130   1150
                                1940   2450   2520   2590   2640   2680   2830
                                2950   4060   4180   4500   4730   4900   5190
Line_number              1530   2020   2030   2070   2300   2880   3330
Lines_per_page            160    290   1090
Lk_up_tbl(*)               40   1710   3040   3090   3100   3110   3120   3140
                                3160   3180   3190   3210   3260   3490   3520
                                3550   3810   3920   4420   5070   5080   5120
                                5130
Ln_pos                   1540   4410   4450   4460   4530   4540   4560   4570
                                4580   4590   4680   4690   4760   4770   4790
                                4800   4830
Ln_slot                  1550   4400   4450   4460   4530   4540   4560   4600
                                4680   4690   4760   4770   4810   4820
M                        5090   5100   5110   5220   5230
Max_files                 170    280    450
N_slot                   1560   3880   3920   3930   3980   4010   4020   4110
                                4120   4140   4150   4290   4300   4400
Name_block               1570   2280   3030   3040   3070   3090   3100   3110
                                3120   3140   3160   3180   3190   3210   3230
                                3260   3870
Name_table$(*)             50   3070   3140   3230   3650   3670   3720   3730
                                3840   3870   4200   4210   4330   4340   5100
                                5230
Num_files                 180    440    450    490    590    860
Num_refs                 1580   3980   3990   4000
Num_tokens               1590   2020   2070   2090   2100   2880
Page                      190   1060   1110   1140
Perforated                200    800    820    920    930   1000   1010   1020
                                1120   1130
Print_block              1600   3590   3650   3670   3680   3720   3730   3810
                                3840   3920   4200   4210   5210
Print_var                1610   3600   3650   3670   3690   3720   3730   3810
                                3840   3920   4200   4210   5220
Printercode               210    340    350   1030   1040   1190   1200
Ref_table(*)               60   1700   2330   2340   3300   3310   3320   3330
                                3360   3370   3400   3810   3930   3980   4010
                                4020   4110   4120   4140   4150   4290   4300
                                4440   4450   4460   4530   4540   4560   4600
                                4650   4680   4690   4760   4770   4810   4820
                                4860   4870   4940   4950   4990   5000
Refs_printed             1620   3820   3900   4030   4040   4090   4380   4470
                                4480   4550   4700   4710   4780   4880   4930
```



```
                                                         CROSS


                                 4960
Slot                       1630  2330  2340  3120  3210  3260  3300  3310
                                 3320  3330  3360  3370  3400  3880
Slot_size                    80    90  3300  3360  3990  4140  4580  4630
                                 4800  4810  4980  4990
Subpro                     1640  2860  2980
X_block                    1650  3450  3490  3520  3530  3550

USER DEFINED FUNCTIONS:

SUB PROGRAMS:
Analyze                    2020  2070  2880

JUMP TARGETS:
370                         350   410
420                         380
440                         360   390   470
480                         450
560                         630
650                         570   720
690   Mass_error:           660
730                         680   790
800                         760
820                         770
840                         810
1000  Header:               890
1090  Lines:               1950  2460  2530  2600  2650  2690  2840  2960
                                 4070  4190  4510  4740  4910  5200
1110  New_page:            1090
1130  Set_printer:          850  1840  1890  1920  2430  2780
1230  Xref:                 900
1700  Setup:               1670  1910  2810
1770                       1680
1860  File_error:          1780
1910                       1810
1980                       2090  2390
2060                       2040
2090                       2030  3000
2190                       2150
2230                       2190
2360                       2180  2220  2230  2250
2380                       2110  2350
2410  File_done:           1770
2420  Out:                 2050
2760                       2740
2790                       1850  1900
2810  Next_mod:            2750
2930                       2890
2950                       2920
3020  Find_slot:           2270  2320  2360  3860
3040                       3170
```

```
CROSS


3160  Next_block:       3100
3180  New_block:        3090
3260  Found_slot:       3070
3270                    3150  3250
3290  Put_ref:          2370
3300                    3410
3320                    3390
3350  Next_slot:        3300
3400                    3360
3420                    3340
3440  Output:           2490  2550  2610  2660  2710
3490                    3540
3550  Summed:           3490
3670  Num_order:        3640
3680                    3660
3700  Nextj:            3650  3670
3790  List_ref:         3750  5240
3900                    3790
3930                    3890
3960                    3930
3980                    3960  4160
4100                    3810  4110  4530  4760  4940
4110                    4040
4130                    4010  4020
4170                    3950  4140
4240  Print_label:      3940
4280  Find_label:       4250
4310                    4330
4370  Ord_lbl_refs:     4260
4440                    4660
4530                    4470
4560                    4460
4620                    4560
4660                    4580  4610  4630
4670  Run_ln:           4440
4600                    4800  4840
4760                    4700
4790                    4690
4850  Run_lb:           4450
4860                    4980  5020
4940                    4880
4970                    4870
5030  Back:             4680  4810  4860  4990
5050  Badlabels:        2730
5070                    5140
5150                    5120
5160  Label_err:        5100


SUB PROGRAM Analyze

COMMON VARIABLES:
```

```
                                                CROSS

Tokens$(*)                  5300  5900  6120  6350

VARIABLES:
Chr$                        5320  5600  5610  5620  5630  5640  5650  5660
                                  5670  5680  5740  5810  5860  5890  5950
                                  5980  6010  6020  6090  6150  6220  6270
                                  6310
Common                      5330  6040  6540
Done                        5340  5510  5600  5760  5800  5880  5970  6170
                                  6240  6390
Label_flg                   5350  5640  6110  6520
Line$                       5270  5440  5450  5590  5610  5730  5740
Line_number                 5270  5440  6340
Mat_flg                     5360  5680  6070  6470  6480
Mat_func                    5370  5670  6080  6560
Num_tokens                  5270  5420  5900  5910  6120  6130  6360
Parens                      5380  5650  5660  6070  6470
Pos                         5390  5450  5470  5590  5610  5720  5730  5740
Poss_label                  5400  5630  5640  6090  6100  6210  6530
Sub_flg                     5410  6020  6050  6060  6550
Token$                      5500  5860  5900  5950  6020  6030  6040  6050
                                  6070  6080  6090  6110  6120  6150  6220
                                  6250  6280  6310  6350  6460  6470  6480
                                  6490  6500  6520  6530  6540  6550  6560
X$                          5560  5570  5580

USER DEFINED FUNCTIONS:
FNCaps                      5570  5610  5980  6270
FNDigits                    5580  5610  5640  5890
FNVar                       5560  6010

SUB PROGRAMS:

JUMP TARGETS:
5420                        5290
5490                        5520
5540  Token:                5490
5590                        5550
5600                        5700  5830
5720  Grab_chr:             5480  5690  5790  5820  5870  5960  6160  6230
5760  Fin:                  5730
5770                        5750
5790  Run_quote:            5620  5810
5850  Num_label:            5640
5860                        5890
5940  Key_or_var:           5610
6000  Variable:             6430
6010                        6180
6150                        6010
6200  Key_word:             5980
6230                        6320
```

```
CROSS


6270                        6240
6310                        6270
6340  End_environ:          6460
6380  Line_done:            5590  5600  5800  5880  5970  6170  6260  6510
6420  Function:             6280
6450  Keyflags:             6290
6520                        6490
6580                        6570
```

# Listing Formatting Program

## Object of Program

This program allows you to format the listing of programs. Normally, a program listed using the LIST command will print out without break on unperforated paper or in predetermined page sizes on perforated paper. Using this program, the listing can be split into pages on unperforated paper. Also, you can set the number of lines per page and the width of each line.

This program reads the program to be listed from a DATA file, reading one line at a time. The line, if longer than the specified line width, is then split to fit within the line width. This split is made, if at all possible, to avoid splitting a word or number. The program is kept as readable as possible. The continuation line will be indented to further aid readability.

On perforated paper the program will be split as it normally would be except that you may choose the number of lines per page and maximum number of characters per line. On unperforated paper, pages will be separated by a printed line that approximates the page delimiters of perforated paper.

Any line feed or backspace, which might have been embedded in comments to format a listing, will have its normal meaning and need not be altered in order to use this program.

## User Instructions

The LIST FORMATTING program is in DATA file "LISTER"

a. Type: GET "LISTER"
b. Press: EXECUTE

A hard copy printer is required for this program.

### Data to be entered

You will be asked to enter the following information:

1) The number of files you wish to list.

2) The number of lines you want printed per page (from 35 to 58)

3) The maximum number of characters you want per line (from 40 to 80)

4) Information concerning your printer:

a. The select code of your printer (e.g. 0 for internal thermal printer)
b. Whether or not you are using an HP-IB (if applicable)
c. The bus address of your HP-IB, if applicable (e.g. 5)

5) The names of the files you want listed:

a. These files should be on a single mass storage device.

b. If you make a mistake, go on entering names—when all names have been entered you may make corrections by entering the file number and correct file name. For example:

| | | |
|---|---|---|
| You entered: | Tset | |
| Which is listed as: | FILE #3 | Tset |
| What you wanted was: | Test | |
| When asked to make corrections,enter: | 3,Test | |
| and press CONT | | |
| Now displayed: | FILE #3 | Test |

When all corrections are made as indicated press CONT.

6) You will be asked whether or not you are using perforated paper.

When this information has been input the program will begin listing your programs in the format you have requested. If any of the files listed cannot be found on the mass storage medium you will be so notified and the program will continue with the next file.

# LISTER

```
10    ! PROGRAM TO FORMAT PROGRAM LISTINGS


20    !  Variables:
30       INTEGER N           ! Number of files to be listed


40    PRINTER IS 16
50    PRINT PAGE,LIN(2),SPA(25)," LISTER PROGRAM ",LIN(2)
60    INPUT "NUMBER OF FILES TO BE LISTED?",N
70    IF N<=0 THEN 60
80    CALL Setup(N)
90    END



110   SUB Setup(INTEGER N)
120      !     This subprogram permits dynamic dimensioning of the file
           name array

130   ! Constants:
140     INTEGER True,False,Lines_per_page,Line_width,Printercode,
           Buscode
150     Buscode=-1
160     True=(1=1)
170     False=(1=2)
180     INPUT "ENTER the number of lines to be printed on each
           page (35 to 58)",Lines_per_page
190       IF (Lines_per_page<59) AND (Lines_per_page>34) THEN 220
200       BEEP
210       GOTO 180

220     INPUT "ENTER the number of characters to be printed on a
           line (40 to 80)",Line_width
230       IF (Line_width<81) AND (Line_width>39) THEN 260
240       BEEP
250       GOTO 220

260     INPUT "ENTER printer select code (0 for internal thermal
           printer)",Printercode
270       IF (Printercode<>0) AND (Printercode<>16) THEN 290
280       GOTO 350

290     INPUT "Are you using HP-IB? (YES/NO)",Answer$
300       IF Answer$[1,1]="Y" THEN 340
310       IF Answer$[1,1]="N" THEN 350
320       BEEP
330       GOTO 290

340     INPUT "ENTER printer HP-IB bus code number",Buscode
```

LISTER

```
350  ! Variables:
360      INTEGER Begin_line          ! True if the current line segment
            is the beginning of a program statement
370      INTEGER Identation          ! Amount the program line is
            idented
380      INTEGER Line_count          ! # of lines printed of current
            page
390      INTEGER Page                ! Used to count # of pages
400      INTEGER Split               ! Used as a pointer for splitting
            a program line
410      INTEGER Linefeed,Last_linefeed  ! Position of a linefeed in
            the print string, used to count linefeeds
420      DIM Line$[160]              ! Program line
430      DIM Line_segment$[160]      ! Portion of line delimited by a
            formfeed
440      DIM Print$[160]             ! String to be printed
450      DIM File_name$(1:N)[6]      ! File names to be listed
460      INTEGER X,Y                 ! Used to find position to divide
            a line
470      INTEGER I,J                 ! Iteration counters


480  GOSUB Get_files

490  LINPUT "LISTING ON PERFORATED PAPER?  (Y/N)",Answer$
500  IF POS(UPC$(Answer$),"Y") THEN Perforated=True
510  IF POS(UPC$(Answer$),"N") THEN Perforated=False
520  IF NOT (POS(UPC$(Answer$),"Y") OR POS(UPC$(Answer$),"N")) THEN
        490

530  Line_count=69               !Initialization for first heading
540  GOSUB Setprinter

550  FOR I=1 TO N
560     ASSIGN #1 TO File_name$(I)&":"&Device$,File_missing
570     IF NOT File_missing THEN GOTO 620
580        PRINTER IS 16
590        PRINT File_name$(I)&":"&Device$&" NOT FOUND"
600        GOSUB Setprinter
610        GOTO 730
620     ON END #1 GOTO 730
630     GOSUB Heading
640     GOSUB Get_line
650     IF LEN(Line_segment$)+NOT Begin_line*Identation>Line_width
           THEN GOTO 690
660     Print$=Line_segment$
670     GOSUB Print_line
680     GOTO 640
690     GOSUB Split_line
700     GOSUB Print_line
```

```
710     IF LEN(Line_segment$) THEN 650
720     GOTO 640
730  NEXT I

740  IF NOT Perforated THEN PRINT LIN(70-Line_count),RPT$("_",80)
750  IF Perforated THEN PRINT PAGE
760  SUBEXIT


770 Setprinter:  !
780     IF Buscode=-1 THEN PRINTER IS Printercode
790     IF Buscode>-1 THEN PRINTER IS Printercode,Buscode
800     RETURN

810 Get_files: !
820     PRINT PAGE,"The following files are to be listed:",LIN(1)
830     FOR J=1 TO N
840        DISP "ENTER FILE NAME:";
850        LINPUT "",File_name$(J)
860        PRINT SPA(10);"FILE #";J;TAB(23);File_name$(J)
870     NEXT J

880     J=0
890     INPUT "ENTER A CORRECTION: (FILE #,CORRECTED NAME) OR (
           PRESS CONT IF NO CORRECTIONS)",J,File_name$(J)
900     IF J=0 THEN 960
910     PRINT PAGE,"The following files are to be listed:",LIN(1)
920     FOR J=1 TO N
930        PRINT SPA(10);"FILE #";J;TAB(23);File_name$(J)
940     NEXT J
950     GOTO 880

960     LINPUT "ENTER MASS STORAGE UNIT SPECIFIER: (eg T15 or F8)",
           Device$
970     RETURN


980 Heading:   !
990     IF Perforated THEN PRINT PAGE
1000    IF NOT Perforated THEN PRINT LIN(70-Line_count),RPT$("_",80)
1010    IF NOT Perforated THEN PRINT LIN(2)
1020    IF Printercode=0 THEN PRINT LIN(4),TAB(35);CHR$(27)&"&k1S"&
           File_name$(I)&CHR$(27)&"&k0S",LIN(2)
1025    IF Printercode<>0 THEN PRINT LIN(4),TAB(35);File_name$(I),LIN(
           2)
1030    Line_count=10
1040    Page=1
1050    RETURN
```

LISTER

```
1060 Get_line: !
1070     IF LEN(Line$) THEN 1180
1080        READ #1;Line$
1090        Backspace=POS(Line$,CHR$(8))  !Strip out backspaces
1100        IF Backspace<>0 THEN Line$=Line$[1,(Backspace-2)*(
               Backspace<>1)]&Line$[Backspace+1]
1110        IF Backspace<>0 THEN GOTO 1090
1120        IF NOT LEN(Line$) THEN Line$=" "
1130        Identation=POS(Line$," ")  ! Compute amount to ident any
               continuation lines
1140        Identation=Identation+1
1150        IF Line$[Identation,Identation]=" " THEN GOTO 1140
1160        Identation=Identation+2
1170        Begin_line=True
1180     Split=POS(Line$,CHR$(12))  ! A formfeed delimites the line
            segment
1190     IF Split=0 THEN Line_segment$=Line$
1200     IF Split=0 THEN Line$=""
1210     IF Split=1 THEN Line_segment$=Line$[1,1]
1220     IF Split=1 THEN Line$=Line$[2]
1230     IF Split>1 THEN Line_segment$=Line$[1,Split-1]
1240     IF Split>1 THEN Line$=Line$[Split]
1250     RETURN


1260 Split_line:  !
1270     X=Line_width-NOT Begin_line*Identation
1280     Y=POS(" ,;+-*/^<>=&()[]",Line_segment$[X,X])
1290     IF Y THEN 1330
1300     X=X-1
1310     IF X<>0 THEN GOTO 1280
1320     X=Line_width-NOT Begin_line*Identation
1330     Print$=Line_segment$[1,X]
1340 Delete_spaces: IF Line_segment$[X+1,X+1]<>" " THEN 1370
1350     X=X+1
1360     GOTO Delete_spaces
1370     Line_segment$=Line_segment$[X+1]
1380     RETURN


1390 Print_line:  !
1410     IF Print$<>CHR$(12) THEN 1440
1420        GOSUB New_page
1430        RETURN
1440     IF Line_count>=10+Lines_per_page THEN GOSUB New_page
1450     IF Line_width<71 THEN PRINT SPA(4+NOT Begin_line*Identation);
1455     IF Line_width>=71 THEN PRINT SPA(NOT Begin_line*Identation);
1460     IF Line_count<10+Lines_per_page THEN 1490
1470        PRINT LIN(1);
1471        Line_count=Line_count+1
```

```
1480           GOSUB New_page
1490        Split=POS(Print$,CHR$(10))
1500        IF Split=0 THEN PRINT Print$;
1510        IF Split=1 THEN PRINT Print$[1,1];
1520        IF Split=1 THEN Print$=Print$[2]
1530        IF Split>1 THEN PRINT Print$[1,Split];
1540        IF Split>1 THEN Print$=Print$[Split+1]
1550        IF Split THEN Line_count=Line_count+1
1560        IF Split AND LEN(Print$) THEN 1460
1561        Line_count=Line_count+1
1580        PRINT LIN(1);
1650        Begin_line=False
1660        RETURN


1670  New_page:  !
1680        Page=Page+1
1690        IF Perforated THEN PRINT PAGE
1700        IF NOT Perforated THEN PRINT LIN(70-Line_count),RPT$("_",80),
               LIN(3)
1710        PRINT LIN(4),TAB(Page MOD 2*60+5);File_name$(I),LIN(2)
1720        Line_count=10
1730        RETURN
```

# LISTER

```
LISTER

MAIN PROGRAM

COMMON VARIABLES:

VARIABLES:
N                           30    60    70    80

USER DEFINED FUNCTIONS:

SUB PROGRAMS:
Setup                       80

JUMP TARGETS:
60                          70


SUB PROGRAM Setup

COMMON VARIABLES:

VARIABLES:
Answer$                    290   300   310   490   500   510   520
Backspace                 1090  1100  1110
Begin_line                 360   650  1170  1270  1320  1450  1650
Buscode                    140   150   340   780   790
Device$                    560   590   960
False                      140   170   510  1650
File_missing               560   570
File_name$(*)              450   560   590   850   860   890   930  1020
                                1710
I                          470   550   560   590   730  1020  1710
Identation                 370   650  1130  1140  1150  1160  1270  1320
                                1450
J                          470   830   850   860   870   880   890   900
                                 920   930   940
Last_linefeed              410
Line$                      420  1070  1080  1090  1100  1120  1130  1150
                                1180  1190  1200  1210  1220  1230  1240
Line_count                 380   530   740  1000  1030  1440  1460  1471
                                1550  1561  1700  1720
Line_segment$              430   650   660   710  1190  1210  1230  1280
                                1330  1340  1370
Line_width                 140   220   230   650  1270  1320
Linefeed                   410
Lines_per_page             140   180   190  1440  1460
N                          110   450   550   830   920
Page                       390  1040  1680  1710
Perforated                 500   510   740   750   990  1000  1010  1690
                                1700
```

```
LISTER


Print$                     440   660  1330  1410  1490  1500  1510  1520
                                1530  1540  1560
Printercode                140   260   270   780   790
Split                      400  1180  1190  1200  1210  1220  1230  1240
                                1490  1500  1510  1520  1530  1540  1550
                                1560
True                       140   160   500  1170
X                          460  1270  1280  1300  1310  1320  1330  1340
                                1350  1370
Y                          460  1280  1290

USER DEFINED FUNCTIONS:

SUB PROGRAMS:

JUMP TARGETS:
180                        210
220                        190   250
260                        230
290                        270   330
340                        300
350                        280   310
490                        520
620                        570
640                        680   720
650                        710
690                        650
730                        610   620
770   Setprinter:          540   600
810   Get_files:           480
880                        950
960                        900
980   Heading:             630
1060  Get_line:            640
1090                      1110
1140                      1150
1180                      1070
1260  Split_line:          690
1280                      1310
1330                      1290
1340  Delete_spaces:      1360
1370                      1340
1390  Print_line:          670   700
1440                      1410
1460                      1560
1490                      1460
1670  New_page:           1420  1440  1480
```

# Search A File For Given Strings

## Object of Program

This program will allow you to search a program, stored on a DATA file, for up to twenty strings. The program will let you input the strings which are to be searched for. As each one is entered it will be assigned a number (#1, #2, etc.). The data file containing the specified program will then be read one line at a time. As it is read the line will be scanned for an occurrence of the strings. If any of the strings do occur in the line, then the numbers of the strings found will be printed, followed by the line.

For instance, if you were searching for the strings "Count", "FOR", and "PRINTER" in your program. The strings would be assigned the numbers:

#1 : Count

#2 : FOR

#3 : PRINTER

If the line being scanned happened to be:

```
150 IF Count=Max THEN PRINTER IS 0
```

The output would be:

```
#1, #3:

150 IF Count = Max THEN PRINTER IS 0
```

If no occurrences of any of the strings are found in the line then there will be no output for that line.

## User Instructions

The program SEARCH is in DATA file "SEARCH".

a. Type: GET "SEARCH"

b. Press: EXECUTE

### Data to be entered

You will be asked to enter information concerning your printer including the select code (16 for the CRT, 0 for the internal thermal printer), whether or not you are using an HP–IB (if applicable) and what the bus address is (if applicable).

You will then be requested to enter the file name of the file which you wished searched. You may include the mass storage unit specifier in the file name if you wish (for example, FILE:T14). If you do not, then you will be asked to enter the mass storage unit specifier (for example, T14).

You will then be requested to enter the number (maximum 20) of strings. The strings will then be entered, one by one, and assigned numbers. These numbered strings will be output both to the CRT and your printer (if it is not the CRT).

When this information has been entered the specified file will be searched and occurrences of the strings will be output. When the last line of the file has been read the message ''ALL OCCURRENCES OF THE STRINGS FOUND'' will be printed on the CRT.

# SEARCH

```
PROGRAM TO SEARCH A PROGRAM FOR OCCURENCES OF STRINGS

20    PRINTER IS 16
30    DIM Answer$[5]                   !Comment: response to a question
40    INTEGER Bus                      !Comment: bus code (if applicable)
41    INTEGER Count                    !Comment: loop counter for strings
50    DIM Device$[5]                   !Comment: mass storage device
60    DIM File$[12]                    !Comment: file to be searched
61    INTEGER Found                    !Comment: flag that string has been
         found
70    DIM Line$[160]                   !Comment: the current program line
80    INTEGER Number                   !Comment: number of strings to be
         searched for
90    INTEGER Printer                  !Comment: hard copy printer
100   DIM Strings$(20)[160]            !Comment: strings to be searched for

Comment: Enter file name, printer

110   PRINT PAGE;LIN(3),TAB(20),"SEARCH FILE FOR STRINGS"
120   INPUT "ENTER THE SELECT CODE FOR YOUR PRINTER (eg. 0 for
         internal thermal printer)",Printer
130      IF (Printer=0) OR (Printer=16) THEN 210
140   INPUT "ARE YOU USING AN HP-IB? (YES/NO)",Answer$
150      IF Answer$[1,1]="N" THEN 210
160      IF Answer$[1,1]="Y" THEN 190
170        BEEP
180        GOTO 140
190   INPUT "ENTER HP-IB BUS CODE",Bus
200      Printer=Printer*100+Bus
210      IF Printer=16 THEN 270
220      ON ERROR GOTO Printer_err
230      PRINTER IS Printer
240      PRINT LIN(1)
250      OFF ERROR
260      PRINTER IS 16
270   INPUT "ENTER THE FILE NAME OF THE FILE YOU WISH TO SEARCH:",
         File$
280   IF POS(File$,":")<>0 THEN 320

Comment: Enter mass storage device code

290   INPUT "ENTER MASS STORAGE DEVICE CODE WHERE FILE CAN BE FOUND (
         eg. T14 or F8)",Device$
300   File$=File$&":"&Device$
310   ON ERROR GOTO File_err
320   ASSIGN #1 TO File$
330   OFF ERROR
340   PRINT PAGE;LIN(3);TAB(20);"SEARCHING ";File$,LIN(3)
350   IF Printer=16 THEN 380
360      PRINTER IS Printer
```

SEARCH

```
370      PRINT LIN(3);TAB(20);"SEARCHING ";File$,LIN(3)
```

Comment: Enter number of strings

```
380  INPUT "ENTER THE NUMBER OF STRINGS YOU WISH TO SEARCH FOR (
        maximum 20)",Number
390     IF (Number>0) AND (Number<21) THEN 520
400     BEEP
410     GOTO 380
```

Comment: ERROR IN FINDING FILE

```
430 File_err:    !
440    BEEP
450    PRINT "ERROR: FILE NOT FOUND; PLEASE REENTER INFORMATION"
460    GOTO 210
```

Comment: PRINTER ERROR

```
480 Printer_err:   !
490    BEEP
500    PRINT "ERROR: PRINTER ERROR; PLEASE REENTER INFORMATION"
510    GOTO 120
```

Comment: ENTER STRINGS

```
530    PRINTER IS 16
540      FOR Count=1 TO Number
550      DISP "ENTER STRING #";Count;":";
560      LINPUT "",Strings$(Count)
570      PRINT "#";Count;":  ";Strings$(Count)
580      IF Printer=16 THEN 620
590      PRINTER IS Printer
600      PRINT "#";Count;":  ";Strings$(Count)
610      PRINTER IS 16
620      NEXT Count
```

Comment: SEARCH FILE FOR STRINGS

```
640  PRINTER IS Printer
650     PRINT LIN(2);"----------------------------------------",LIN(2)
660 Read:  !
670  ON END #1 GOTO Done
680  ON ERROR GOTO Read_err
690  READ #1;Line$
700   Found=0
710   FOR Count=1 TO Number
720   IF POS(Line$,Strings$(Count))=0 THEN 760
730      IF Found=1 THEN PRINT ", ";
740      Found=1
```

```
750      PRINT "#";Count;
760   NEXT Count
770   IF Found THEN PRINT ":"
780   IF Found THEN PRINT "     ";Line$
790   GOTO Read

Comment: ENTIRE FILE SEARCHED

810 Done:  !
820   DISP "ALL OCCURENCES OF THE STRINGS FOUND"
830   STOP

Comment: READ ERROR

850 Read_err: !
860   BEEP
870   PRINT "ERROR: WRONG DATA TYPE ENCOUNTERED; PROGRAM STOPPED"
880   STOP
890   END
```

# SEARCH

```
SEARCH

MAIN PROGRAM

COMMON VARIABLES:

VARIABLES:
Answer$                     30   140   150   160
Bus                         40   190   200
Count                       41   540   550   560   570   600   620   710
                                 720   750   760
Device$                     50   290   300
File$                       60   270   280   300   320   340   370
Found                       61   700   730   740   770   780
Line$                       70   690   720   780
Number                      80   380   390   540   710
Printer                     90   120   130   200   210   230   350   360
                                 580   590   640
Strings$(*)                100   560   570   600   720

USER DEFINED FUNCTIONS:

SUB PROGRAMS:

JUMP TARGETS:
120                        510
140                        180
190                        160
210                        130   150   460
270                        210
320                        280
380                        350   410
430   File_err:            310
480   Printer_err:         220
520                        390
620                        580
660   Read:                790
760                        720
810   Done:                670
850   Read_err:            680
```

# Character Redefinition Program

## Object of Program

This program allows you to redefine up to nine characters printed by the internal thermal printer. Each new character will replace a specified character usually output by the printer.

The characters which are produced by the thermal printer are actually a set of dots within a 7X12 matrix. For the most part characters are usually restricted to a smaller 5X7 matrix within the larger printing zone. The columns on either side of the inner matrix are usually blank to separate characters. Similarly the top row of dots is usually blank to separate printed lines. The bottom row of dots is used primarily for an underline. If the character has "descenders", such as the tails on "y" or "g", they will fall in the third and second rows from the bottom. "Ascenders", such as an umlaut, would go in the second from top row of dots. Some examples are shown below:

```
     1 2 3 4 5 6 7          1 2 3 4 5 6 7          1 2 3 4 5 6 7
  1  . . . . . . .       1  . . . . . . .       1  . . . . . . .
  2  . . . . . . .       2  . . @ . @ . .       2  . . . . . . .
  3  . . @ @ @ . .       3  . . . . . . .       3  . . . . . . .
  4  . @ . . . @ .       4  . . @ @ @ . .       4  . . . . . . .
  5  . @ . . . @ .       5  . @ . . . @ .       5  . @ . @ @ . .
  6  . @ @ @ @ @ .       6  . @ . . . @ .       6  . @ @ . . @ .
  7  . @ . . . @ .       7  . @ . . . @ .       7  . @ . . . @ .
  8  . @ . . . @ .       8  . @ . . . @ .       8  . @ @ . . @ .
  9  . @ . . . @ .       9  . . @ @ @ . .       9  . @ . @ @ . .
 10  . . . . . . .      10  . . . . . . .      10  . @ . . . . .
 11  . . . . . . .      11  . . . . . . .      11  . @ . . . . .
 12  . . . . . . .      12  . . . . . . .      12  . . . . . . .
```

Dot matrices representing "A, "O" with an umlaut and "p" where "@" is a printed dot and "." is a dot which is not printed.

## Restrictions

Due to the manner in which character redefinition for the internal thermal printer is implemented (see the System 45B Operating and Programming Manual, Appendix B, The Internal Printer, New Characters) there are certain restrictions on the new characters which you can define. These restrictions are summarized below:

1) Rows 5 through 9 may be defined without restriction. Any pattern of dots may be used in these rows.
2) If any combination of the rows 1, 2 and 12 are defined (i.e., have dots to be printed) they must be identical to one another. For example, if you have defined a dot pattern for row 2 and you wish to define row 12, it will have to have the SAME dot pattern as row 2. Thus, you could not have an umlaut (two dots) defined in row 2 and an underline in row 12.

3) Row 3 and row 10 are tied together in a specific way:

   a. If row 3 has not been defined row 10 may be defined as desired.

   b. If row 10 has not been defined, row 3 may be defined as desired.

   c. If row 3 has been defined and rows 1,2 and 12 are NOT defined, then row 10 may be defined as desired.

   d. If row 3 has been defined and any combination of rows 1,2 and 12 have been defined then row 10 must be the same as the rows 1,2 and 12 which are defined.

   e. If row 10 has been defined and any combination of rows 1,2 and 12 have been defined to be different from row 10 then row 3 may NOT be defined without first making row 10 blank.

4) Rows 4 and 11 have a relationship exactly as described for 3 and 10 above. Rules for their defintion can be obtained by substituting 4 for 3 and 11 for 10 in restricton 3) above.

These restrictions may seem rather complicated, but the CHARACTER REDEFINITION program has been written to simplify the process of redefinition as much as possible. The program will monitor which rows have been defined and which ones may be redefined. When there is a restriction involved only the legal options will be made available to you. The restrictions should be kept in mind when you are designing characters, however, since some combinations of dots cannot be achieved.

## User Instructions

The program CHARACTER REDEFINITION is in DATA file "CHRDEF"

   a. Type: GET "CHRDEF"

   b. Press: EXECUTE

The internal thermal line printer is necessary for this program.

### Data to be entered

A brief description of how a character is defined is included as a prologue to this program. Also included are a list of restrictions for the redefinition process.

Initially an empty 7X12 matrix, representing the character you wish to define, will be displayed. Dots within the matrix which will not be printed are represented as periods (".")

You will be asked to enter the number of the row of dots which you wish to change. Choose a row, enter its number and press CONT. The row you have chosen will then be displayed for you to edit. If you wish to have a dot printed then put an "X" in place of the period at that position. If there is a dot already there then you can remove it by replacing it with a period. Note that there is a blank between each two dot positions which should not be altered. When you have altered the displayed row to your satisfation press CONT. The matrix will then be changed to include the new row. Dots will replace the X's which you entered.

Should you accidently enter a character in place of one of the blanks separating the dot positions, or should you enter a character other than "." or "X", then you will be notified that you have entered an illegal row. The original row will then be redisplayed for you to change. If you enter a wrong row number then press CONT before making any changes in the row.

A row may be altered any number of times.

Should you encounter one of the restrictions involved in character redefinition, then a message telling you what the conflict is will be displayed. You will then be offered numbered options describing what actions are available to you. You should select one of these options, enter its number and press CONT. The action you have chosen will be carried out and you may then enter a new row number to alter.

When you are satisfied with the character you have defined then enter "0" as the row you wish to change. You will then be asked to enter the character which you wish to redefine. You may enter this character either as a single character (e.g. %) or in the form of a CHR$ expression (e.g. CHR$(37)). Should the character be one of the control characters (i.e., NUM of the character is less than 32) then, even though the character is redefined, it will not be printed unless the control characters are "disabled" (see Appendix B, Disabling Control Codes in the System 45B Operating and Programming Manual).

**NOTE**

You might accidentally enter "0" for the row number before you are ready to redefine a character. You can continue changing the character by entering "NO" when asked which character you wish to define. You will then be asked to enter the row which you wish to change. At this point you can continue as before.

You will then be asked if you wish to redefine the character at this time. If you answer "NO", then the string of characters which you need to output to the internal thermal printer to redefine the desired character will be printed out on the thermal printer. If you wish to redefine the character at some future time you need only print out this series to the thermal printer.

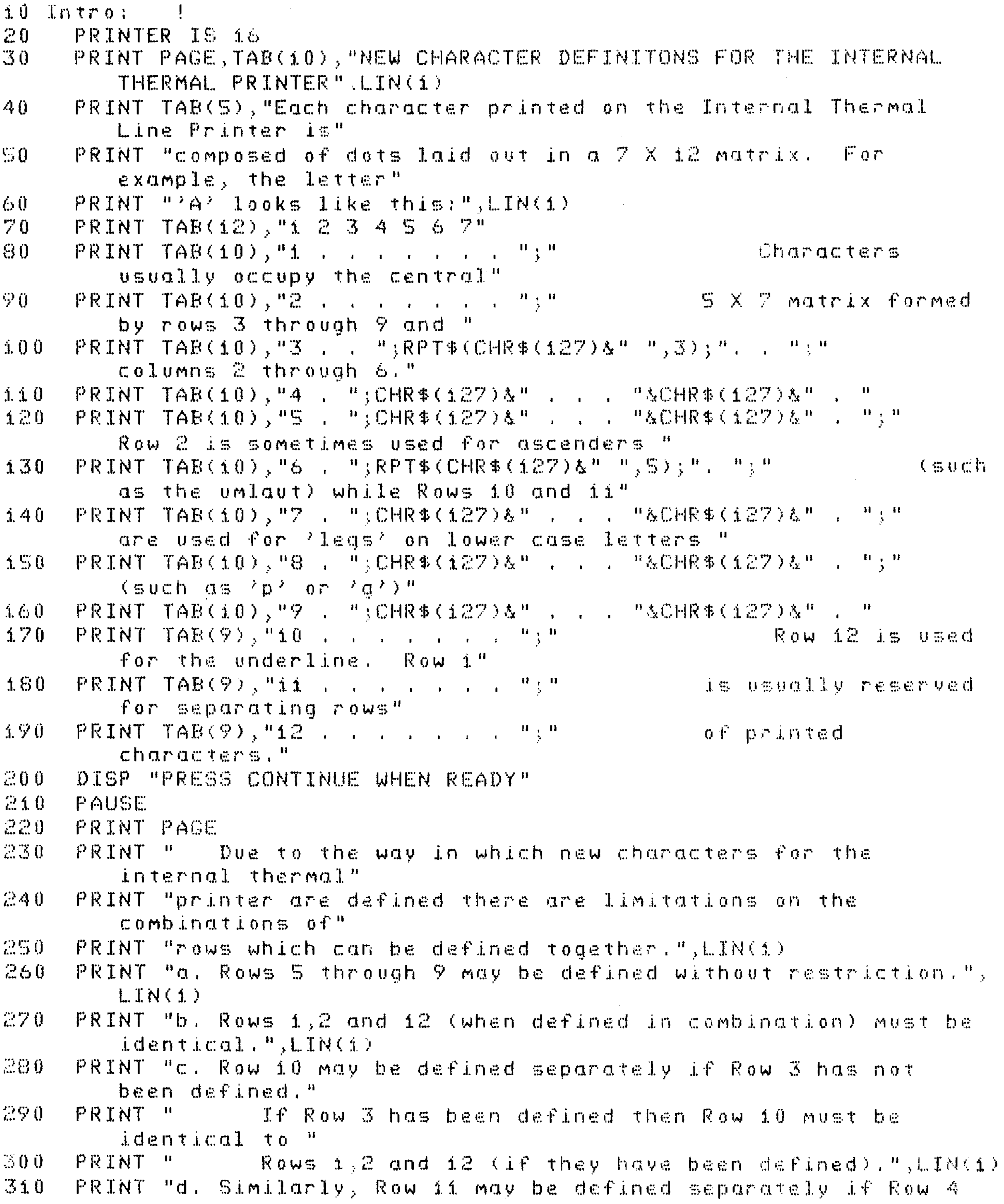
If you DO wish to redefine the character at this time then, in addition to the string of characters mentioned above, the new character will be printed out. Until the thermal printer has been reset (CONTROL/STOP), or turned off, any time that the redefined character is printed on the thermal printer the NEW character will be printed.

Another way that the definitions may be nullified is to use the following commands:

```
PRINTER IS 0
PRINT CHR$(27)&"E"
```

Up to nine characters may be redefined at one time. Should you redefine a tenth character then the last (ninth) definition will be replaced and will no longer hold. If you redefine a previously redefined character the EARLIEST redefinition will hold (unless it is the ninth redefined character). Once you have redefined a character there is no way to change it again without resetting all the characters to their original definitions.

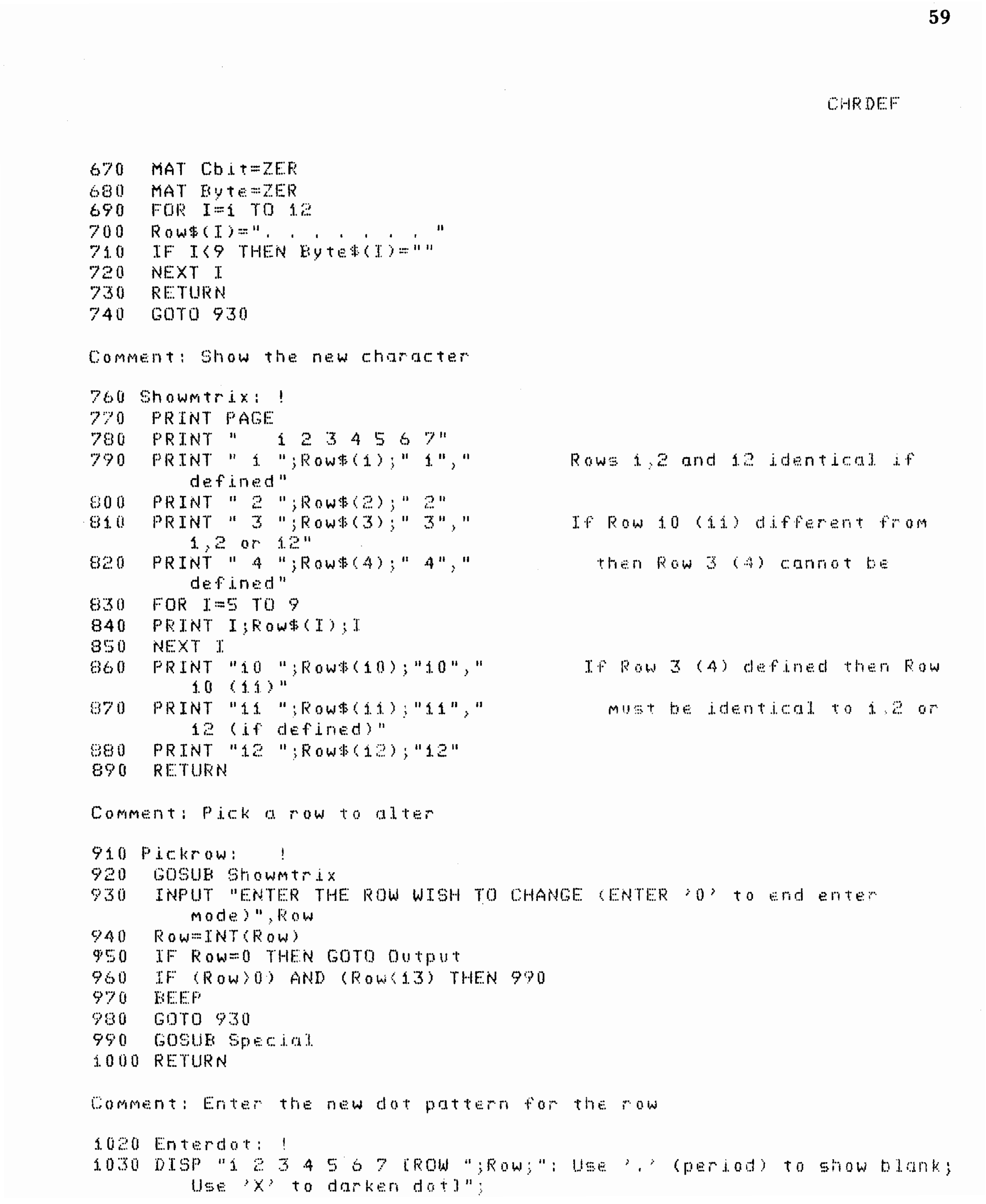
# CHRDEF



```
10 Intro:    !
20    PRINTER IS 16
30    PRINT PAGE,TAB(10),"NEW CHARACTER DEFINITONS FOR THE INTERNAL
         THERMAL PRINTER".LIN(1)
40    PRINT TAB(5),"Each character printed on the Internal Thermal
         Line Printer is"
50    PRINT "composed of dots laid out in a 7 X 12 matrix.  For
         example, the letter"
60    PRINT "'A' looks like this:",LIN(1)
70    PRINT TAB(12),"1 2 3 4 5 6 7"
80    PRINT TAB(10),"1 . . . . . . . ";"                 Characters
         usually occupy the central"
90    PRINT TAB(10),"2 . . . . . . . ";"            5 X 7 matrix formed
         by rows 3 through 9 and "
100   PRINT TAB(10),"3 . . ";RPT$(CHR$(127)&" ",3);". . ";"
         columns 2 through 6."
110   PRINT TAB(10),"4 . ";CHR$(127)&" . . . "&CHR$(127)&" . "
120   PRINT TAB(10),"5 . ";CHR$(127)&" . . . "&CHR$(127)&" . ";"
         Row 2 is sometimes used for ascenders "
130   PRINT TAB(10),"6 . ";RPT$(CHR$(127)&" ",5);". ";"           (such
         as the umlaut) while Rows 10 and 11"
140   PRINT TAB(10),"7 . ";CHR$(127)&" . . . "&CHR$(127)&" . ";"
         are used for 'legs' on lower case letters "
150   PRINT TAB(10),"8 . ";CHR$(127)&" . . . "&CHR$(127)&" . ";"
         (such as 'p' or 'g')"
160   PRINT TAB(10),"9 . ";CHR$(127)&" . . . "&CHR$(127)&" . "
170   PRINT TAB(9),"10 . . . . . . . ";"                 Row 12 is used
         for the underline.  Row 1"
180   PRINT TAB(9),"11 . . . . . . . ";"            is usually reserved
         for separating rows"
190   PRINT TAB(9),"12 . . . . . . . ";"            of printed
         characters."
200   DISP "PRESS CONTINUE WHEN READY"
210   PAUSE
220   PRINT PAGE
230   PRINT "   Due to the way in which new characters for the
         internal thermal"
240   PRINT "printer are defined there are limitations on the
         combinations of"
250   PRINT "rows which can be defined together.",LIN(1)
260   PRINT "a. Rows 5 through 9 may be defined without restriction.",
         LIN(1)
270   PRINT "b. Rows 1,2 and 12 (when defined in combination) must be
         identical.",LIN(1)
280   PRINT "c. Row 10 may be defined separately if Row 3 has not
         been defined."
290   PRINT "       If Row 3 has been defined then Row 10 must be
         identical to "
300   PRINT "       Rows 1,2 and 12 (if they have been defined).",LIN(1)
310   PRINT "d. Similarly, Row 11 may be defined separately if Row 4
```

CHRDEF

```
        has not been"
320  PRINT "      defined.  If Row 4 has been defined then Row 11
        must be identical "
330  PRINT "       to Rows 1,2 and 12 (if they have been defined)"
340  PAUSE

Comment: Main program

360 Begin:  PRINTER IS 16
370  GOSUB Setup
380  GOSUB Pickro
390  GOTO 380
400  GOSUB Output

Comment: Declarations

420  DIM A$[7]                     !Comment: Response to question
430  DIM Binrow(12)                !Comment: Binary equivalent of dot
        row
440  INTEGER Bit                   !Comment: Bit specifier
450  DIM Byte$(8)[3]               !Comment: Octal representation of
        matrix byte
460  INTEGER Byte(8)                   !Comment: representation of
        matrix byte
470  DIM C$[10]                     !Comment: Character to be redefined
480  DIM Cbit(7)                   !Comment: Bits set to vary row
        configuration
490  INTEGER Chr                   !Comment: Numeric value of
        redefined character
500  INTEGER Confbit               !Comment: Number of conflicting bit
510  INTEGER Confrow               !Comment: Number of conflicting row
520  DIM Crow$[20]                 !Comment: Temporary row holder
530  DIM Def1$[12]                 !Comment: First part of character
        definition
540  DIM Def2$[40]                 !Comment: Second part of character
        definition
550  INTEGER Defchr                !Comment: Character to be redefined
560  INTEGER I                     !Comment: Loop counter
570  INTEGER Legal                 !Comment: Row legality flag
580  DIM Lrow$[20]                 !Comment: Temporary row holder
590  DIM Newchr$[10]                !Comment: Character to be redefined
600  INTEGER Prntchr               !Comment: Actual redefinition flag
610  INTEGER Row                   !Comment: Row specifier
620  DIM Row$(12)[21]              !Comment: Row definitions

Comment: Set up for a new character

640 Setup:   !
650  PRINT PAGE
660  MAT Binrow=ZER
```



CHRDEF

```
670  MAT Cbit=ZER
680  MAT Byte=ZER
690  FOR I=1 TO 12
700  Row$(I)=". . . . . . . "
710  IF I<9 THEN Byte$(I)=""
720  NEXT I
730  RETURN
740  GOTO 930

Comment: Show the new character

760 Showmtrix: !
770  PRINT PAGE
780  PRINT "   1 2 3 4 5 6 7"
790  PRINT " 1 ";Row$(1);" 1","        Rows 1,2 and 12 identical if
        defined"
800  PRINT " 2 ";Row$(2);" 2"
810  PRINT " 3 ";Row$(3);" 3","        If Row 10 (11) different from
        1,2 or 12"
820  PRINT " 4 ";Row$(4);" 4","          then Row 3 (4) cannot be
        defined"
830  FOR I=5 TO 9
840  PRINT I;Row$(I);I
850  NEXT I
860  PRINT "10 ";Row$(10);"10","         If Row 3 (4) defined then Row
        10 (11)"
870  PRINT "11 ";Row$(11);"11","           must be identical to 1,2 or
        12 (if defined)"
880  PRINT "12 ";Row$(12);"12"
890  RETURN

Comment: Pick a row to alter

910 Pickrow:    !
920  GOSUB Showmtrix
930  INPUT "ENTER THE ROW WISH TO CHANGE (ENTER '0' to end enter
        mode)",Row
940  Row=INT(Row)
950  IF Row=0 THEN GOTO Output
960  IF (Row>0) AND (Row<13) THEN 990
970  BEEP
980  GOTO 930
990  GOSUB Special
1000 RETURN

Comment: Enter the new dot pattern for the row

1020 Enterdot: !
1030 DISP "1 2 3 4 5 6 7 [ROW ";Row;": Use '.' (period) to show blank;
        Use 'X' to darken dot]";
```

CHRDEF

```
1040 Crow$=Row$(Row)
1050 EDIT " ",Crow$
1060 GOSUB Legal
1070 IF Legal THEN 1130
1080 BEEP
1090 DISP "INCORRECT CONFIGURATION: TRY AGAIN"
1100 WAIT 1000
1110 Crow$=Row$(Row)
1120 GOTO 1030
1130 Row$(Row)=Crow$
1140 RETURN
```

Comment: Check for a legal string of dots and blanks

```
1160 Legal:!
1170 Binrow(Row)=0
1180 Lrow$=Crow$
1190 Crow$=""
1200 FOR I=1 TO 7
1210 IF (Lrow$[1,2]="X ") OR (Lrow$[1,2]=CHR$(127)&" ") THEN 1250
1220 IF Lrow$[1,2]=". " THEN 1280
1230 Legal=0
1240 GOTO 1320
1250 Crow$[2*I-1,2*I]=CHR$(127)&" "
1260 Binrow(Row)=Binrow(Row)+2^(8-I)
1270 GOTO 1290
1280 Crow$[2*I-1,2*I]=". "
1290 Lrow$=Lrow$[3,LEN(Lrow$)]
1300 NEXT I
1310 Legal=1
1320 RETURN
```

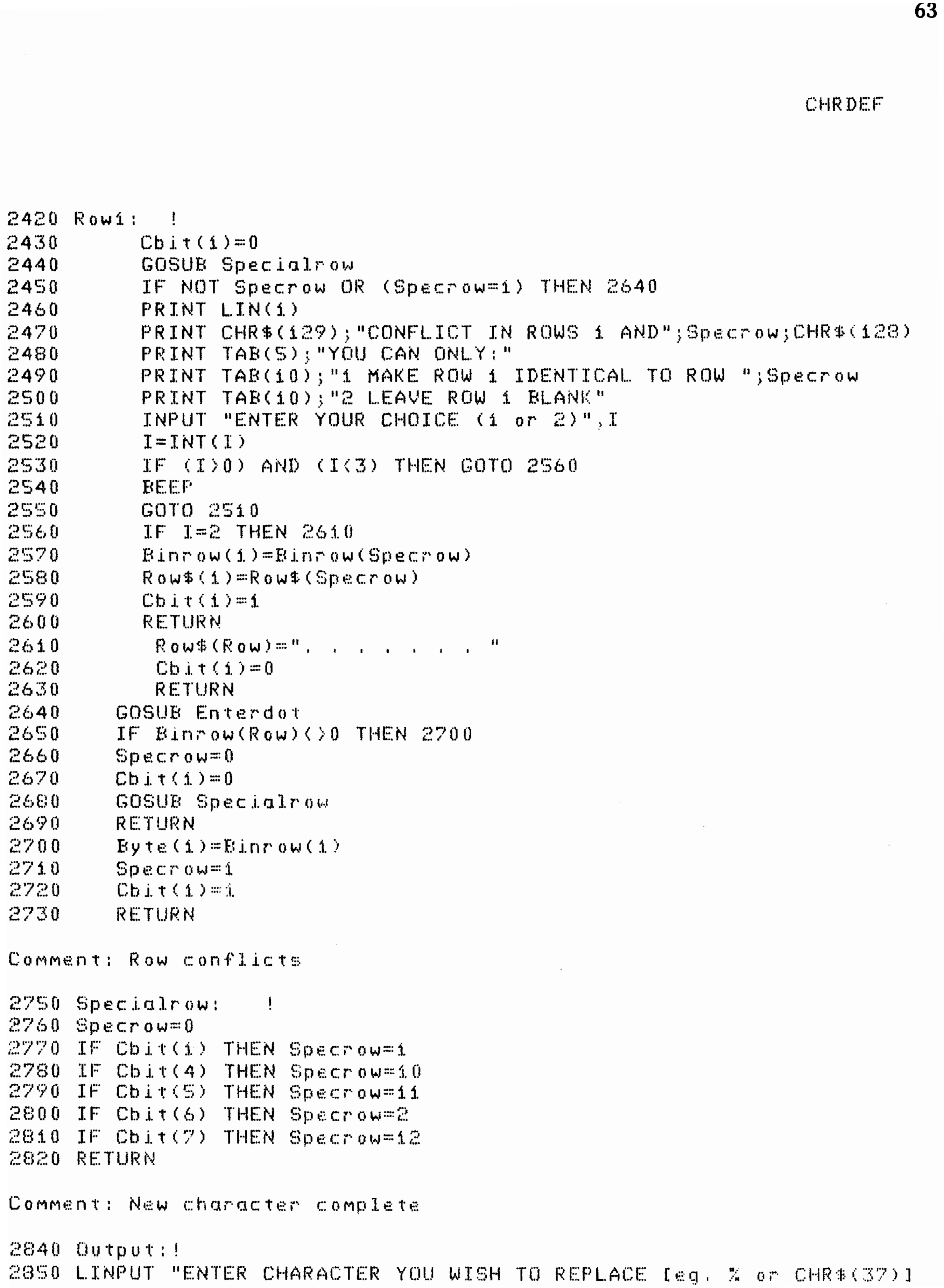
Comment: Check for row conflicts

```
1340 Special:!


1360 Row56789: IF (Row<5) OR (Row>9) THEN Row3
1370      GOSUB Enterdot
1380      Byte(Row-1)=Binrow(Row)
1390      RETURN


1410 Row3: IF (Row<>3) AND (Row<>4) THEN Row10
1420 IF Row=3 THEN Confrow=10
1430 IF Row=4 THEN Confrow=11
1440 IF Row=3 THEN Confbit=2
1450 IF Row=4 THEN Confbit=3
1460       IF NOT Cbit(Confbit) THEN 1520
1470       BEEP
```

```
1480       PRINT LIN(1),"CONFLICT IN ROWS ";CHR$(129);Row;" AND ";
              Confrow;CHR$(128),LIN(1)
1490       PRINT TAB(5),"YOU CAN NOT SET THIS ROW UNLESS ROW ";
              Confrow;" IS SET TO BLANK"
1500       INPUT "PRESS CONTINUE TO CONTINUE",Q
1510       RETURN
1520       GOSUB Enterdot
1530       Byte(Row-1)=Binrow(Row)
1540       RETURN


1560 Row10: IF (Row<>10) AND (Row<>11) THEN Row12
1570        IF Row=10 THEN Confrow=3
1580        IF Row=10 THEN Confbit=2
1590        IF Row=11 THEN Confrow=4
1600        IF Row=11 THEN Confbit=3
1610        IF Row=10 THEN Bit=4
1620        IF Row=11 THEN Bit=5
1630        IF NOT Specrow OR (Specrow=Row) THEN 1820
1640        IF NOT Binrow(Confrow) THEN 1960
1650        PRINT LIN(1)
1660        PRINT "CONFLICT IN ROWS ";
1670       PRINT CHR$(129);Specrow;", ";Row;" AND ";Confrow;CHR$(128)
1680       PRINT TAB(5),"YOU CAN ONLY:"
1690       PRINT TAB(10),"1 MAKE ROW ";Row;" IDENTICAL TO ROW";
              Specrow
1700       PRINT TAB(10),"2 LEAVE ROW ";Row;" BLANK"
1710       INPUT "ENTER YOUR CHOICE (1 or 2)",I
1720       IF (INT(I)>0) AND (INT(I)<3) THEN 1750
1730       BEEP
1740       GOTO 1710
1750       IF INT(I)=2 THEN Row$(Row)=".  .  .  .  .  .  .  ."
1760       IF INT(I)=2 THEN Cbit(Bit)=0
1770       IF INT(I)=2 THEN RETURN
1780       Binrow(Row)=Binrow(Specrow)
1790       Row$(Row)=Row$(Specrow)
1800       Cbit(Bit)=1
1810       RETURN
1820      IF NOT Specrow THEN 1870
1830      Cbit(Bit)=0
1840      GOSUB Specialrow
1850      IF NOT Specrow THEN 1870
1860      GOTO 1640
1870      GOSUB Enterdot
1880      IF Binrow(Row)<>0 THEN 1920
1890      Specrow=Cbit(Bit)=0
1900      GOSUB Specialrow
1910      RETURN
1920      Byte(1)=Binrow(Row)
1930      Cbit(Bit)=1
```

CHRDEF

```
1940       Specrow=Row
1950       RETURN
1960        GOSUB Enterdot
1970        IF Binrow(Row)<>0 THEN 2020
1980        Byte(Confbit)=0
1990        Cbit(Confbit)=0
2000        Cbit(Bit)=0
2010        RETURN
2020        Byte(Confbit)=Binrow(Row)
2030        Cbit(Confbit)=1
2040        RETURN


2060 Row12: IF (Row<>12) AND (Row<>2) THEN 2420
2070         IF Row=2 THEN Bit=6
2080         IF Row=12 THEN Bit=7
2090         IF NOT Specrow THEN 2320
2100         IF Specrow<>Row THEN 2140
2110         Cbit(Bit)=0
2120         GOSUB Specialrow
2130         IF NOT Specrow THEN 2320
2140         PRINT LIN(1)
2150         PRINT "CONFLICT IN ROWS ";CHR$(129);Specrow;CHR$(128);
                " AND ";CHR$(129);Row;CHR$(128)
2160         PRINT TAB(5);"YOU CAN ONLY:"
2170         PRINT TAB(10);"1 MAKE ROW ";Row;" IDENTICAL TO ROW ";
                Specrow
2180         PRINT TAB(10);"2 LEAVE ROW ";Row;" BLANK"
2190         INPUT "ENTER YOU CHOICE (1 or 2)",I
2200         I=INT(I)
2210         IF (I>0) AND (I<3) THEN 2240
2220         BEEP
2230         GOTO 2190
2240         IF I=2 THEN 2290
2250         Binrow(Row)=Binrow(Specrow)
2260         Row$(Row)=Row$(Specrow)
2270         Cbit(Bit)=1
2280         RETURN
2290        Row$(Row)=". . . . . . . . "
2300        Cbit(Bit)=0
2310        RETURN
2320         Specrow=Row
2330         GOSUB Enterdot
2340         IF Binrow(Row)<>0 THEN 2380
2350         Cbit(Bit)=0
2360         GOSUB Specialrow
2370         RETURN
2380         Byte(1)=Binrow(Row)
2390         Cbit(Bit)=1
2400         RETURN
```



```
2420 Row1:  !
2430         Cbit(1)=0
2440         GOSUB Specialrow
2450         IF NOT Specrow OR (Specrow=1) THEN 2640
2460         PRINT LIN(1)
2470         PRINT CHR$(129);"CONFLICT IN ROWS 1 AND";Specrow;CHR$(128)
2480         PRINT TAB(5);"YOU CAN ONLY:"
2490         PRINT TAB(10);"1 MAKE ROW 1 IDENTICAL TO ROW ";Specrow
2500         PRINT TAB(10);"2 LEAVE ROW 1 BLANK"
2510         INPUT "ENTER YOUR CHOICE (1 or 2)",I
2520         I=INT(I)
2530         IF (I>0) AND (I<3) THEN GOTO 2560
2540         BEEP
2550         GOTO 2510
2560         IF I=2 THEN 2610
2570         Binrow(1)=Binrow(Specrow)
2580         Row$(1)=Row$(Specrow)
2590         Cbit(1)=1
2600         RETURN
2610          Row$(Row)=".  .  .  .  .  .  .  .  "
2620          Cbit(1)=0
2630          RETURN
2640      GOSUB Enterdot
2650      IF Binrow(Row)<>0 THEN 2700
2660      Specrow=0
2670      Cbit(1)=0
2680      GOSUB Specialrow
2690      RETURN
2700      Byte(1)=Binrow(1)
2710      Specrow=1
2720      Cbit(1)=1
2730      RETURN

Comment: Row conflicts

2750 Specialrow:   !
2760 Specrow=0
2770 IF Cbit(1) THEN Specrow=1
2780 IF Cbit(4) THEN Specrow=10
2790 IF Cbit(5) THEN Specrow=11
2800 IF Cbit(6) THEN Specrow=2
2810 IF Cbit(7) THEN Specrow=12
2820 RETURN

Comment: New character complete

2840 Output:!
2850 LINPUT "ENTER CHARACTER YOU WISH TO REPLACE [eg. % or CHR$(37)]
```

```
         ",C$
2860 Newchr$=C$
2870 IF LEN(C$)=1 THEN GOTO 2970
2880 IF (POS(C$,"(")<>0) AND (POS(C$,")")<>0) AND (POS(C$,"(")<POS(C$,
         ")")) THEN 2940
2890 BEEP
2900 DISP C$;" IS NOT AN ACCEPTABLE CHARACTER; TRY AGAIN"
2910 WAIT 2000
2920 GOTO 2850

Comment: Define new character string for thermal printer

2940 C$=C$[POS(C$,"(")+1,POS(C$,")")-1]
2950 Defchr=Chr=VAL(C$)
2960 GOTO 2990
2970 ON ERROR GOTO 2900
2980 Defchr=Chr=NUM(C$)
2990 Chr$=FNOct$(Chr)
3000 Byte$(1)=FNOct$(Byte(1)+Cbit(1))
3010 Byte$(2)=FNOct$(Byte(2)+Cbit(2))
3020 Byte$(3)=FNOct$(Byte(3)+Cbit(3))
3030 Byte$(4)=FNOct$(Byte(4)+Cbit(6))
3040 Byte$(5)=FNOct$(Byte(5))
3050 Byte$(6)=FNOct$(Byte(6)+Cbit(4))
3060 Byte$(7)=FNOct$(Byte(7)+Cbit(5))
3070 Byte$(8)=FNOct$(Byte(8)+Cbit(7))

Comment: Output to printer if desired

3090 Print: !
3100 INPUT "DO YOU WISH TO REDEFINE THE CHARACTER AT THIS TIME? (YES/
         NO)",A$
3110 IF A$[1,1]="Y" THEN Prntchr=1
3120 IF A$[1,1]="N" THEN Prntchr=0
3130 IF (A$[1,1]="Y") OR (A$[1,1]="N") THEN 3160
3140 BEEP
3150 GOTO 3100
3160 Def1$=CHR$(27)&"&n"&Chr$&"c"
3170 Def2$=Byte$(1)&"p"&Byte$(2)&"q"&Byte$(3)&"r"&Byte$(4)&"s"&Byte$(
         5)&"t"&Byte$(6)&"u"&Byte$(7)&"v"&Byte$(8)&"W"

Comment: Output to thermal printer

3190 PRINTER IS 0
3200 PRINT "CHARACTER BEING REDEFINED: ";Newchr$,LIN(1)
3210 PRINT "STRING TO DEFINE NEW CHARACTER: ";
3220 PRINT CHR$(27)&"Y";
3230 PRINT Def1$&Def2$;
3240 PRINT ""
3250 IF NOT Prntchr THEN 3330
```

```
3260 PRINT Def1$&Def2$
3270 PRINT "NEWLY DEFINED CHARACTER: ";
3280 IF Defchr<32 THEN PRINT CHR$(27)&"Y";
3290 PRINT CHR$(Defchr);"    ";RPT$(CHR$(Defchr),5);
3300 IF Defchr<32 THEN PRINT CHR$(27)&"
3310 IF Defchr<32 THEN PRINT "REDEFINED CONTROL CODES MUST BE ENABLED
        TO BE PRINTED"
3320 PRINT LIN(2)
3330 PRINTER IS 16
3340 Def2$=""
3350 Def1$=""
```

Comment: Another character?

```
3370 Again: !
3380 INPUT "DO YOU WISH TO REDEFINE ANOTHER CHARACTER? (YES/NO)",A$
3390 IF A$[1,1]="Y" THEN Begin
3400 IF A$[1,1]="N" THEN 3430
3410 BEEP
3420 GOTO 3380
3430 STOP
3440 END
3450 DEF FNOct$(INTEGER X)
3460 DIM X$[4]
3470 FOR I=2 TO 1 STEP -1
3480 X$[3-I,3-I]=CHR$(INT(X DIV 8^I)+48)
3490 X=X MOD 8^I
3500 NEXT I
3510     RETURN X$&CHR$(X+48)
3520     FNEND
```

# CHRDEF

```
CHRDEF

MAIN PROGRAM

COMMON VARIABLES:

VARIABLES:
A$                        420  3100  3110  3120  3130  3380  3390  3400
Binrow(*)                 430   660  1170  1260  1380  1530  1640  1780
                               1880  1920  1970  2020  2250  2340  2380
                               2570  2650  2700
Bit                       440  1610  1620  1760  1800  1830  1890  1930
                               2000  2070  2080  2110  2270  2300  2350
                               2390
Byte$(*)                  450   710  3000  3010  3020  3030  3040  3050
                               3060  3070  3170
Byte(*)                   460   680  1380  1530  1920  1980  2020  2380
                               2700  3000  3010  3020  3030  3040  3050
                               3060  3070
C$                        470  2850  2860  2870  2880  2900  2940  2950
                               2980
Cbit(*)                   480   670  1460  1760  1800  1830  1890  1930
                               1990  2000  2030  2110  2270  2300  2350
                               2390  2430  2590  2620  2670  2720  2770
                               2780  2790  2800  2810  3000  3010  3020
                               3030  3050  3060  3070
Chr                       490  2950  2980  2990
Chr$                     2990  3160
Confbit                   500  1440  1450  1460  1580  1600  1980  1990
                               2020  2030
Confrow                   510  1420  1430  1480  1490  1570  1590  1640
                               1670
Crow$                     520  1040  1050  1110  1130  1180  1190  1250
                               1280
Def1$                     530  3160  3230  3260  3350
Def2$                     540  3170  3230  3260  3340
Defchr                    550  2950  2980  3280  3290  3300  3310
I                         560   690   700   710   720   830   840   850
                               1200  1250  1260  1280  1300  1710  1720
                               1750  1760  1770  2190  2200  2210  2240
                               2510  2520  2530  2560
Legal                     570  1070  1230  1310
Lrow$                     580  1180  1210  1220  1290
Newchr$                   590  2860  3200
Prntchr                   600  3110  3120  3250
Q                        1500
Row                       610   930   940   950   960  1030  1040  1110
                               1130  1170  1260  1360  1380  1410  1420
                               1430  1440  1450  1480  1530  1560  1570
                               1580  1590  1600  1610  1620  1630  1670
                               1690  1700  1750  1780  1790  1880  1920
```

```
CHRDEF

                                   1940  1970  2020  2060  2070  2080  2100
                                   2150  2170  2180  2250  2260  2290  2320
                                   2340  2380  2610  2650
Row$(*)                      620    700   790   800   810   820   840   860
                                    870   880  1040  1110  1130  1750  1790
                                   2260  2290  2580  2610
Specrow                     1630   1670  1690  1780  1790  1820  1850  1890
                                   1940  2090  2100  2130  2150  2170  2250
                                   2260  2320  2450  2470  2490  2570  2580
                                   2660  2710  2760  2770  2780  2790  2800
                                   2810

USER DEFINED FUNCTIONS:
FNOct$                      2990   3000  3010  3020  3030  3040  3050  3060
                                   3070

SUB PROGRAMS:

JUMP TARGETS:
10      Intro:
360     Begin:              3390
380                          390
640     Setup:               370
760     Showmtrix:           920
910     Pickrow:             380
930                          740    980
990                          960
1020    Enterdot:           1370   1520  1870  1960  2330  2640
1030                        1120
1130                        1070
1160    Legal:              1060
1250                        1210
1280                        1220
1290                        1270
1320                        1240
1340    Special:             990
1360    Row56789:
1410    Row3:               1360
1520                        1460
1560    Row10:              1410
1640                        1860
1710                        1740
1750                        1720
1820                        1630
1870                        1820   1850
1920                        1880
1960                        1640
2020                        1970
2060    Row12:              1560
2140                        2100
2190                        2230
```

```
                                                                        CHRDEF


2240                          2210
2290                          2240
2320                          2090   2130
2380                          2340
2420   Row1:                  2060
2510                          2550
2560                          2530
2610                          2560
2640                          2450
2700                          2650
2750   Specialrow:            1840   1900   2120   2360   2440   2680
2840   Output:                 400    950
2850                          2920
2900                          2970
2940                          2880
2970                          2870
2990                          2960
3090   Print:
3100                          3150
3160                          3130
3330                          3250
3370   Again:
3380                          3420
3430                          3400


FNOct$

COMMON VARIABLES:

VARIABLES:
I                             3470   3480
X                             3450   3480   3490   3510
X$                            3460   3480   3510

USER DEFINED FUNCTIONS:

SUB PROGRAMS:

JUMP TARGETS:
```

# Header Program

## Object of Program

This program prints characters approximately eight times the size of the normal characters printed by the internal thermal printer. This is accomplished by redefining five characters on the thermal printer (see CHARACTER DEFINITION) to be solid blocks or triangles. These new characters are then used in 5X8 groups to produce a character set equivalent to that of the internal thermal printer but much larger.

With this program you can simply print out a message in large characters or you can produce a subprogram which, when called, will print out a message in header characters on the thermal printer. The printing facility of this program might be used to make small signs or to put header titles before listings of programs. The subprogram facility might be used to print a specific message in a particular program

## Character Definition

The characters redefined by this program are:

1) \ is redefined to be a solid block (7X8 dots)
2) } is redefined to be a solid lower right triangle
3) | is redefined to be a solid lower left triangle.
4) ` is redefined to be a solid upper right triangle
5) { is redefined to be a solid upper left triangle

These five shapes and the blank are used to build the header characters. Each character is a 5X8 matrix. The eighth row, at the bottom, is considered to be "below the line" and is used for tails for the Q and small letters, punctuation and the underline character. In the program characters are defined by flattening the 5X8 matrix into a 40 character string. Thus, the "O" is defined by the matrix:

```
  12345

1 }\\\|
2 \   \
3 \   \
4 \   \
5 \   \
6 \   \
7 `\\\{
8
```

The flattened matrix looks like this:

```
row:     1         2        3        4        5        6          7         8
       "}\\\|      \        \\       \\       \        \\        \`\\\{          "
```

The matrices for the characters to be printed are assigned to the dummy variable X$ and then processed to form a complete line of up to 13 characters. Since the SHAPES described above are defined in blocks (and not the full 7X12 matrix for single characters) the internal thermal printer has been reset to print only 8 rows of dots per line instead of the default 12 (see the System 45B Operating and Programming Manual, Appendix B, The Internal Printer, Rows per line). Therefore, should this program be stopped without following the steps outlined in the User Instructions section, the internal thermal printer may continue to act strangely. Should this occur then the commands:

```
PRINTER IS 0
PRINT CHR$(27);"E"
PRINTER IS 16
```

should clear up the problem (CONTROL/STOP will also eliminate the problem but may also destroy things you are working on at the time.)

When the subprogram facility of the HEADER program is used, a subprogram is constructed by this program in a DATA file. This subprogram can print up to 8 lines of 13 characters each. The subprogram itself can be attached to a main program using a "GET" statement. It can then be invoked by using a "CALL Header" statement in the appropriate place in the main program.

## User Instructions

The program HEADER is in the DATA file "HEADER"

a. Type: GET "HEADER"
b. Press: EXECUTE

The internal thermal line printer is necessary for this program.

### Data to be entered

You will be asked if you wish to create a subprogram. If you don't want to create one then answer "NO". You may then enter as many lines to be printed in HEADER characters as you like. Each line should be no more than 13 characters as this is the maximum number of characters that will fit on one line. After each line is entered at the keyboard and CONT is pressed, that line will be printed on the internal printer. To stop just answer "NO" when asked if you wish to enter more lines.

If you wish to create a subprogram then you will be asked to enter the mass storage unit specifier(e.g. T15 for the main tape drive, or F8 for a flexible disk drive). You will then be asked to enter a file name. This should be a new name not previously used on the storage medium indicated. A DATA file 28 records long (256 bytes per record) will then be created on your mass storage medium.

You now will be requested to enter up to 8 lines of characters (again, no more than 13 characters per line). After each of these lines is entered it will be printed by the internal printer for you to review. If the line is satisfactory then when asked if you wish to store the line answer "YES". If not, answer "NO" and reenter the line if you wish. Until you have stored 8 lines or have indicated you are done you will be prompted to enter another line if desired.

When a line of characters is stored, it is stored in the form of PRINT statements. The subprogram header and the necessary manipulations of the internal printer will be automatically generated for the subprogram. When you have entered all the lines desired and so indicated, the program will finish creating the subprogram and print directions as to how to access it.

For example, if your file was named ''HFILE'' and the last line of code in your main program was 4600, you could attach the subprogram by using the statement.

**GET "HFILE",5000**

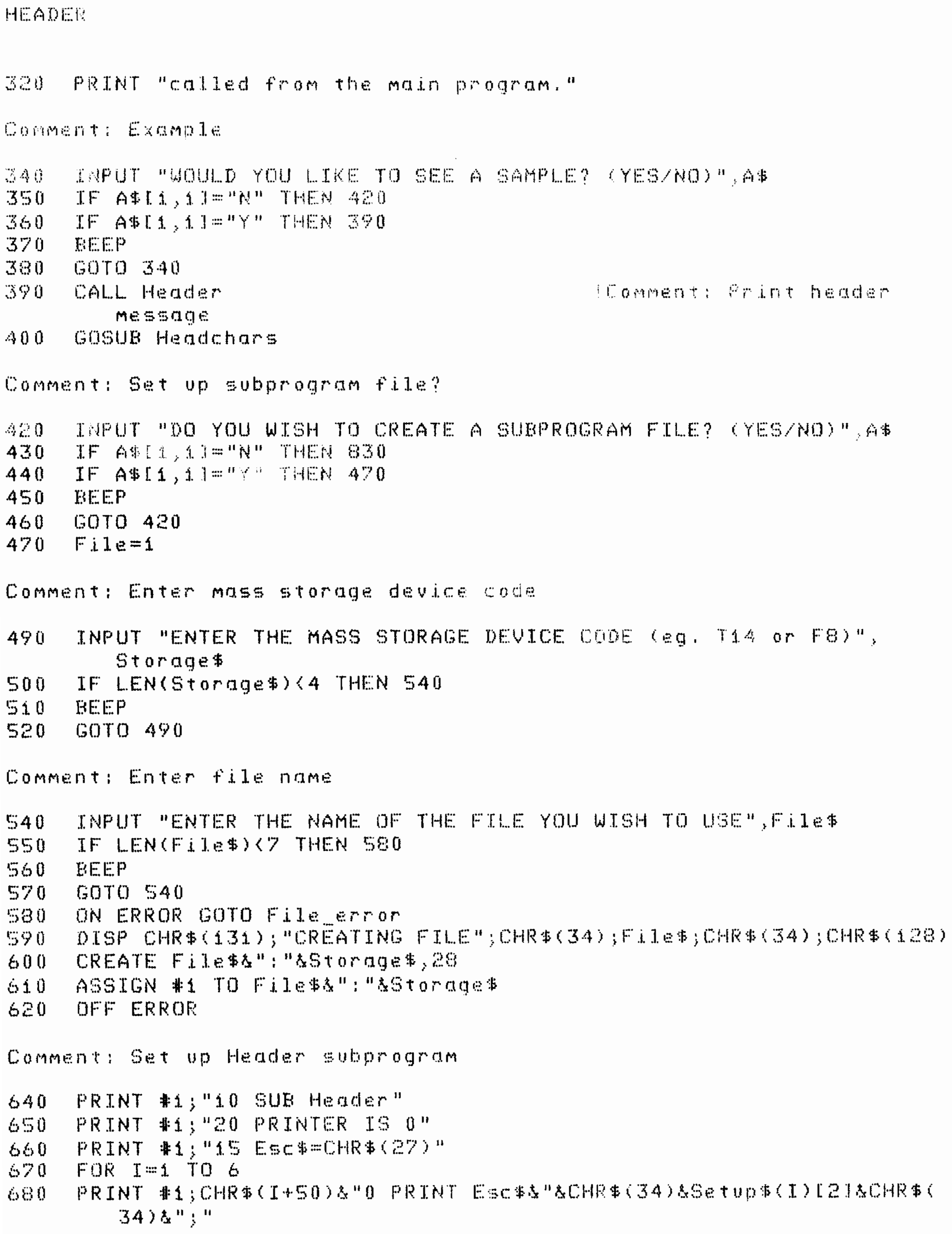
Once attached the message encoded in the subprogram will be printed on the internal printer each time a statement of the form:

**100 CALL Header**

is encountered. ''Header'' is the name of the subprogram that is synthesized by the HEADER program.

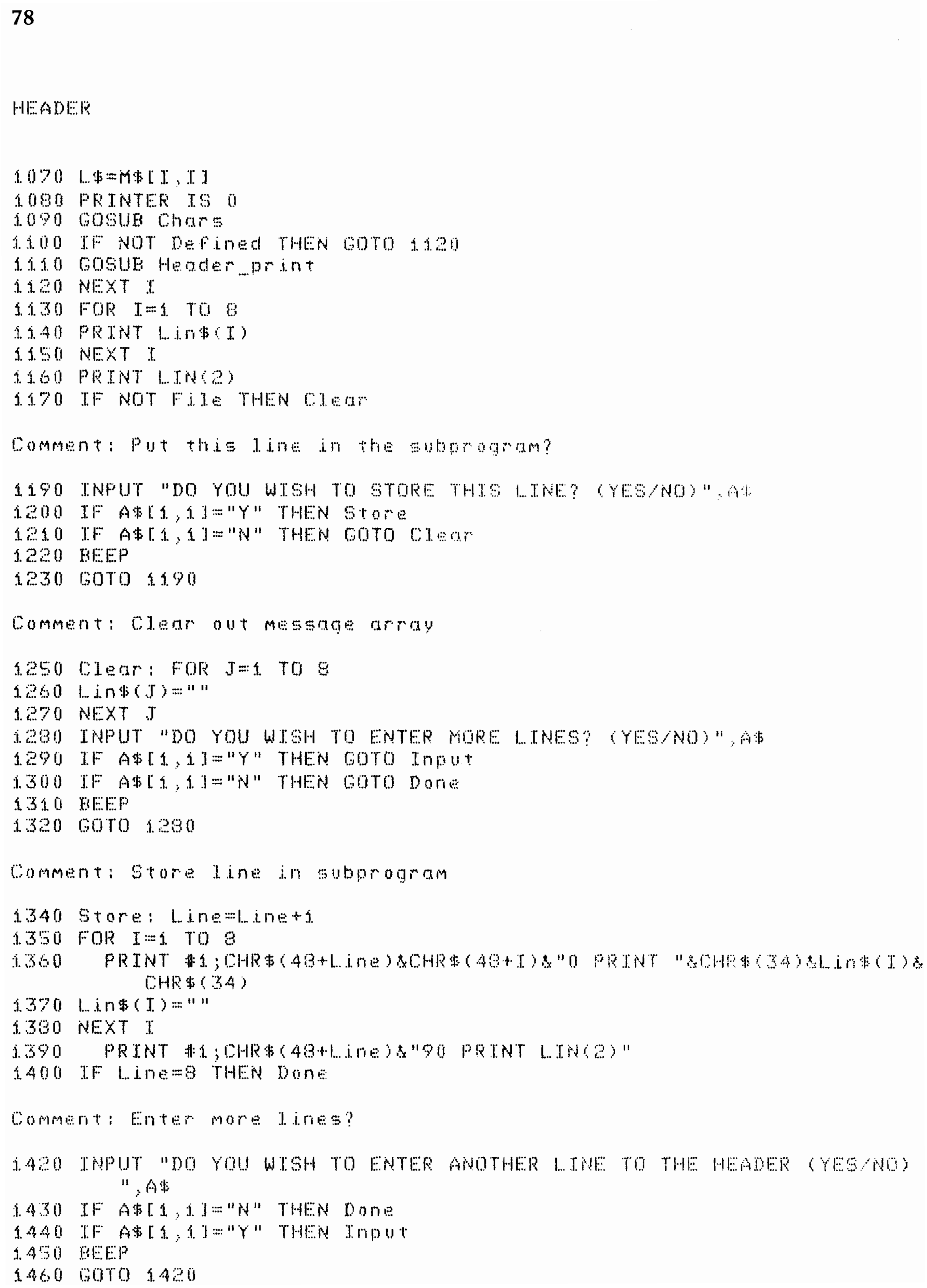
# HEADER

```
              HEADER PROGRAM

Comment: Declarations

30    DIM A$[3]                        !Comment: Response string
40    INTEGER Defined                  !Comment: Flag for undefined
         characters
50    DIM Esc$[1]                      !Comment: Esacpe character-CHR$(
         27)
60    INTEGER File                     !Comment: Flag for subprogram
         uasage
70    DIM File$[12]                    !Comment: File name
80    INTEGER I                        !Comment: Loop counter
90    INTEGER J                        !Comment: Loop counter
100   DIM L$[1]                        !Comment: Current character of M$
110   DIM Lin$(8)[80]                  !Comment: Message array - header
120   INTEGER Line                     !Comment: Line counter
130   DIM M$[160]                      !Comment: Message string
140   INTEGER N                        !Comment: Value of character
150   INTEGER Row                      !Comment: Row counter for loop
160   DIM Setup$(7)[40]                !Comment: Setup strings for
         thermal printer
170   DIM X$[160]                      !Comment: Temporary string for
         character definition
180   DIM Storage$[4]                  !Comment: Mass storage devive code
190   INTEGER Width                    !Comment: Width of character
         matrix

Comment: Initalization

210   Esc$=CHR$(27)
220   GOSUB Headchars                           !Comment: Define new
         characters
230   PRINT PAGE,LIN(2)
240   Line=0

Comment: Introduction

260   PRINT TAB(20);"HEADER PRINTING PROGRAM",LIN(3)
270   PRINT "     This program can store your header message in the
         form"
280   PRINT "of a subprogram named 'Header' on a specified file.",LIN(
         1)
290   PRINT "     The subprogram can then be attached to your own
         program"
300   PRINT "using a GET statement.  The produced subprogram will
         print "
310   PRINT "out the message on the Internal Thermal Printer when it
         is"
```



```
HEADER


320   PRINT "called from the main program."

Comment: Example

340   INPUT "WOULD YOU LIKE TO SEE A SAMPLE? (YES/NO)",A$
350   IF A$[1,1]="N" THEN 420
360   IF A$[1,1]="Y" THEN 390
370   BEEP
380   GOTO 340
390   CALL Header                               !Comment: Print header
         message
400   GOSUB Headchars

Comment: Set up subprogram file?

420   INPUT "DO YOU WISH TO CREATE A SUBPROGRAM FILE? (YES/NO)",A$
430   IF A$[1,1]="N" THEN 830
440   IF A$[1,1]="Y" THEN 470
450   BEEP
460   GOTO 420
470   File=1

Comment: Enter mass storage device code

490   INPUT "ENTER THE MASS STORAGE DEVICE CODE (eg. T14 or F8)",
         Storage$
500   IF LEN(Storage$)<4 THEN 540
510   BEEP
520   GOTO 490

Comment: Enter file name

540   INPUT "ENTER THE NAME OF THE FILE YOU WISH TO USE",File$
550   IF LEN(File$)<7 THEN 580
560   BEEP
570   GOTO 540
580   ON ERROR GOTO File_error
590   DISP CHR$(131);"CREATING FILE";CHR$(34);File$;CHR$(34);CHR$(128)
600   CREATE File$&":"&Storage$,28
610   ASSIGN #1 TO File$&":"&Storage$
620   OFF ERROR

Comment: Set up Header subprogram

640   PRINT #1;"10 SUB Header"
650   PRINT #1;"20 PRINTER IS 0"
660   PRINT #1;"15 Esc$=CHR$(27)"
670   FOR I=1 TO 6
680   PRINT #1;CHR$(I+50)&"0 PRINT Esc$&"&CHR$(34)&Setup$(I)[2]&CHR$(
          34)&";"
```

```
690  NEXT I
700  PRINT #1;CHR$(57)&"0 PRINT Esc$&"&CHR$(34)&Setup$(7)[2]&CHR$(34)
710  GOTO Message

Comment: Error routine

730 File_error: !
740  IF ERRN=54 THEN GOTO Duplicate
750  BEEP
760  PRINT CHR$(129);"MASS STORAGE ERROR #";ERRN;".  FILE ";CHR$(34);
        File$&":"&Storage$;CHR$(34);" NOT ACCESSIBLE";CHR$(128)
770  PRINT CHR$(129);"PLEASE CORRECT PROBLEM AND REENTER INFORMATION";
        CHR$(128)
780  GOTO 490
790 Duplicate:  !
800  BEEP
810  PRINT CHR$(129);"FILE NAME `";File$;"' ALREADY USED; PLEASE
        SELECT ANOTHER";CHR$(128)
820  GOTO 540
830  File=0

Comment: Enter message

850 Message:  PRINTER IS 16
860  PRINT PAGE,LIN(2)
870  PRINT TAB(20);"HEADER PRINTING PROGRAM",LIN(3)
880  PRINT TAB(10);"DEFINED CHARACTERS:",LIN(1)
890  PRINT TAB(15);"A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,
        Z",LIN(1)
900  PRINT TAB(15);"a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,
        z",LIN(1)
910  PRINT TAB(15);"1,2,3,4,5,6,7,8,9,0",LIN(1)
920  PRINT TAB(15);"Math symbols:  * / = + - ^ < > ",LIN(1)
930  PRINT TAB(15);"Puncuation: ! @ # $ % & ( ) _ ~ ` { } | [ ] \ : ;
        ' , . ? ";CHR$(34),LIN(1)
940  PRINT TAB(15);"Blank"
950 Input: DISP "                         ENTER A LINE OF HEADING ";
960  IF File THEN DISP "(8 lines of 13 Characters maximum)";
970  IF NOT File THEN DISP "(13 Characters maximum)";
980  LINPUT "",M$
990  IF LEN(M$)<14 THEN 1050
1000 BEEP
1010 DISP "LINE TOO LONG; PLEASE REENTER"
1020 WAIT 2000
1030 GOTO 950

Comment: Print message

1050 FOR I=1 TO LEN(M$)
1060 Defined=1
```



HEADER

```
1070 L$=M$[I,I]
1080 PRINTER IS 0
1090 GOSUB Chars
1100 IF NOT Defined THEN GOTO 1120
1110 GOSUB Header_print
1120 NEXT I
1130 FOR I=1 TO 8
1140 PRINT Lin$(I)
1150 NEXT I
1160 PRINT LIN(2)
1170 IF NOT File THEN Clear
```

Comment: Put this line in the subprogram?

```
1190 INPUT "DO YOU WISH TO STORE THIS LINE? (YES/NO)",A$
1200 IF A$[1,1]="Y" THEN Store
1210 IF A$[1,1]="N" THEN GOTO Clear
1220 BEEP
1230 GOTO 1190
```

Comment: Clear out message array

```
1250 Clear: FOR J=1 TO 8
1260 Lin$(J)=""
1270 NEXT J
1280 INPUT "DO YOU WISH TO ENTER MORE LINES? (YES/NO)",A$
1290 IF A$[1,1]="Y" THEN GOTO Input
1300 IF A$[1,1]="N" THEN GOTO Done
1310 BEEP
1320 GOTO 1280
```

Comment: Store line in subprogram

```
1340 Store: Line=Line+1
1350 FOR I=1 TO 8
1360   PRINT #1;CHR$(48+Line)&CHR$(48+I)&"0 PRINT "&CHR$(34)&Lin$(I)&
          CHR$(34)
1370 Lin$(I)=""
1380 NEXT I
1390   PRINT #1;CHR$(48+Line)&"90 PRINT LIN(2)"
1400 IF Line=8 THEN Done
```

Comment: Enter more lines?

```
1420 INPUT "DO YOU WISH TO ENTER ANOTHER LINE TO THE HEADER (YES/NO)
       ",A$
1430 IF A$[1,1]="N" THEN Done
1440 IF A$[1,1]="Y" THEN Input
1450 BEEP
1460 GOTO 1420
```

Comment: Finished up subprogram

```
1480 Done:!
1490 IF NOT File THEN 1640
1500 PRINT #1;CHR$(Line+48)&"90 PRINT CHR$(27)&"&CHR$(34)&"E"&CHR$(34)
1510 PRINT #1;CHR$(Line+48)&"95 SUBEND"
1520 ASSIGN #1 TO *
1530 PRINT Esc$&"E"
1540 PRINTER IS 16
1550 PRINT PAGE,LIN(3);"YOUR HEADER SUBPROGRAM IS NOW SAVED ON FILE
        ‘";File$;"’"
1560 PRINT LIN(1);"IN ORDER TO USE THE SUBPROGRAM:"
1570 PRINT TAB(10),"a.  Use the statement  ",LIN(1)
1580 PRINT TAB(10),"          GET ";CHR$(34);File$;CHR$(34);",n",LIN(
        1)
1590 PRINT TAB(10),"    where ‘n’ is a line number greater than the"
1600 PRINT TAB(10),"    largest one in your program.",LIN(1)
1610 PRINT TAB(10),"b.  To print out message using subprogram use:",
        LIN(1)
1620 PRINT TAB(10),"         100 CALL Header",LIN(5)
1630 PRINT TAB(10),"    where ‘100’ can be any line in your program."
1640 PRINTER IS 16
1650 PRINT "          END OF PROGRAM"
1660 PRINTER IS 0
1670 PRINT Esc$&"E"                !Comment: Clear printer of
        special features
1680 STOP
```

Comment: Character definitions

```
1700 Chars:IF (NUM(L$)<65) OR (NUM(L$)>90) THEN GOSUB Specials
1710 IF (NUM(L$)<65) OR (NUM(L$)>90) THEN GOTO 1100
1720 N=NUM(L$)-64
1730 ON N GOSUB A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z
1740 RETURN
1750 A:!
1760 X$="}\\\|\   \\   \\\\\\\   \\   \\   \      "
1770 RETURN
1780 B:  X$="\\\\| \  \ \  \ \\\( \  \ \  \\\\\(      "
1790 RETURN
1800 C:X$="}\\\|\   \\    \    \    \   \‘\\\(      "
1810 RETURN
1820 D:X$="\\\\|\   \\   \\   \\   \\   \\\\\(      "
1830 RETURN
1840 E:X$="\\\\\\   \    \\\  \    \    \\\\\       "
1850 RETURN
1860 F:X$="\\\\\\   \    \\\  \    \    \           "
1870 RETURN
1880 G:X$="}\\\|\   ‘\    \    \ \\\\   \‘\\\(      "
```

HEADER

```
1890 RETURN
1900 H:X$="\   \\   \\   \\\\\\\   \\   \\   \     "
1910 RETURN
1920 I:X$=" \\\   \    \    \    \    \   \\\      "
1930 RETURN
1940 J:X$="  \\\   \    \    \    \ ¦  \ `\\(      "
1950 RETURN
1960 K:X$="\  )(\ )( \)(  \\   \`¦  \ `¦ \  `¦     "
1970 RETURN
1980 L:X$="\    \    \    \    \    \    \\\\\     "
1990 RETURN
2000 M:X$="\¦ )\\`\(\\ \ \\   \\   \\   \\   \     "
2010 RETURN
2020 N:X$="\   \\¦  \\`¦ \\ `¦\\  `\\   \\   \     "
2030 RETURN
2040 O:X$=")\\\¦\   \\   \\   \\   \\   \`\\\(     "
2050 RETURN
2060 P:X$="\\\\¦\   \\   \\\\\(\    \    \         "
2070 RETURN
2080 Q:X$=")\\\¦\   \\   \\   \\ ¦ \\ `¦\`\\\(   ` "
2090 RETURN
2100 R:X$="\\\\¦\   \\   \\\\\(\  `¦\   \\   \     "
2110 RETURN
2120 S:X$=")\\\¦\   \\    `\\\¦    \\   \`\\\(     "
2130 RETURN
2140 T:X$="\\\\\  \    \    \    \    \    \       "
2150 RETURN
2160 U:X$="\   \\   \\   \\   \\   \\   \`\\\(     "
2170 RETURN
2180 V:X$="\   \\   \\   \\   \\   \`¦ )(  `\(     "
2190 RETURN
2200 W:X$="\   \\   \\   \\   \\ \ \\)\¦\`( `(     "
2210 RETURN
2220 X:X$="\   \\   \`¦ )( \\\ )( `¦\   \\   \     "
2230 RETURN
2240 Y:X$="\   \\   \\   \`\\\(  \    \    \       "
2250 RETURN
2260 Z:X$="\\\\\    \   )( )\( )(   \    \\\\\     "
2270 RETURN
2280 PAUSE
2290 Specials: !
2300 IF (NUM(L$)>64) OR (NUM(L$)<48) THEN Puncs
2310 ON NUM(L$)-46 GOSUB Notdefined,Zero,One,Two,Three,Four,Five,Six,
        Seven,Eight,Nine,Col,Sem,Lt,Eq,Gt,Ques,Each
2320 RETURN
2330 Zero:X$=")\\\¦\   \\  )\\ )(\\)( \\(  \`\\\(     "
2340 RETURN
2350 One:X$=" )\    \    \    \    \    \   \\\      "
2360 RETURN
2370 Two:X$=")\\\¦\   \    \ )\\()(   \    \\\\\     "
```

```
2380 RETURN
2390 Three:X$="}\\\¦\   \    \   \\¦    \\   \`\\\{     "
2400 RETURN
2410 Four: X$="\  \ \  \ \  \ \\\\\   \    \    \      "
2420 RETURN
2430 Five:X$="\\\\\\    \    \\\\¦    \\   \`\\\{     "
2440 RETURN
2450 Six: X$="\\\\\\   \\    \\\\\\   \\   \\\\\\     "
2460 RETURN
2470 Seven:X$="\\\\\\   \   }{  }{   \    \    \      "
2480 RETURN
2490 Eight:X$="\\\\\\   \\   \\\\\\\   \\   \\\\\\     "
2500 RETURN
2510 Nine: X$="\\\\\\   \\   \\\\\\   \    \    \     "
2520 RETURN
2530 Ques:X$="}\\\¦\   \    \ }\\{ \    \     "
2540 RETURN
2550 Eq:  X$="        \\\\\     \\\\\          "
2560 RETURN
2570 Col: X$="           \         \            "
2580 RETURN
2590 Sem: X$="                \        \    {      "
2600 RETURN
2610 Lt: X$="        }   }{  }{   `¦    `¦    `       "
2620 RETURN
2630 Gt: X$="      ¦    `¦    `¦   }{  }{   {        "
2640 RETURN
2650 Each:X$="}\\\¦\   \\ }\\\ \ \\ `\{\   }`\\\{     "
2660 RETURN
2670 Puncs:!
2680 IF L$<>" " THEN 2710
2690 X$="                                          "
2700 RETURN
2710 IF (NUM(L$)<33) OR (NUM(L$)>47) THEN Small
2720 ON NUM(L$)-32 GOSUB Ex,Quo,Num,Dol,Perc,And,Apo,Rpar,Lpar,Ast,
        Plus,Com,Minus,Per,Div
2730 RETURN
2740 Ex: X$="  \    \    \    \    \         \       "
2750 RETURN
2760 Quo:X$=" \ \  ` `                              "
2770 RETURN
2780 Num:X$="      \ \ \\\\\ \ \ \\\\\ \ \          "
2790 RETURN
2800 Dol:X$="  \  }\\\\\ \  `\\\¦  \ \\\\\{  \       "
2810 RETURN
2820 Lpar:X$="  `¦    `¦    \    \    \   }{  }{     "
2830 RETURN
2840 Rpar:X$=" }{  }{   \    \    \    `¦    `¦      "
2850 RETURN
2860 Ast: X$="     `¦ }{ `\{ \\\\\ }\¦ }{ `¦         "
```

HEADER

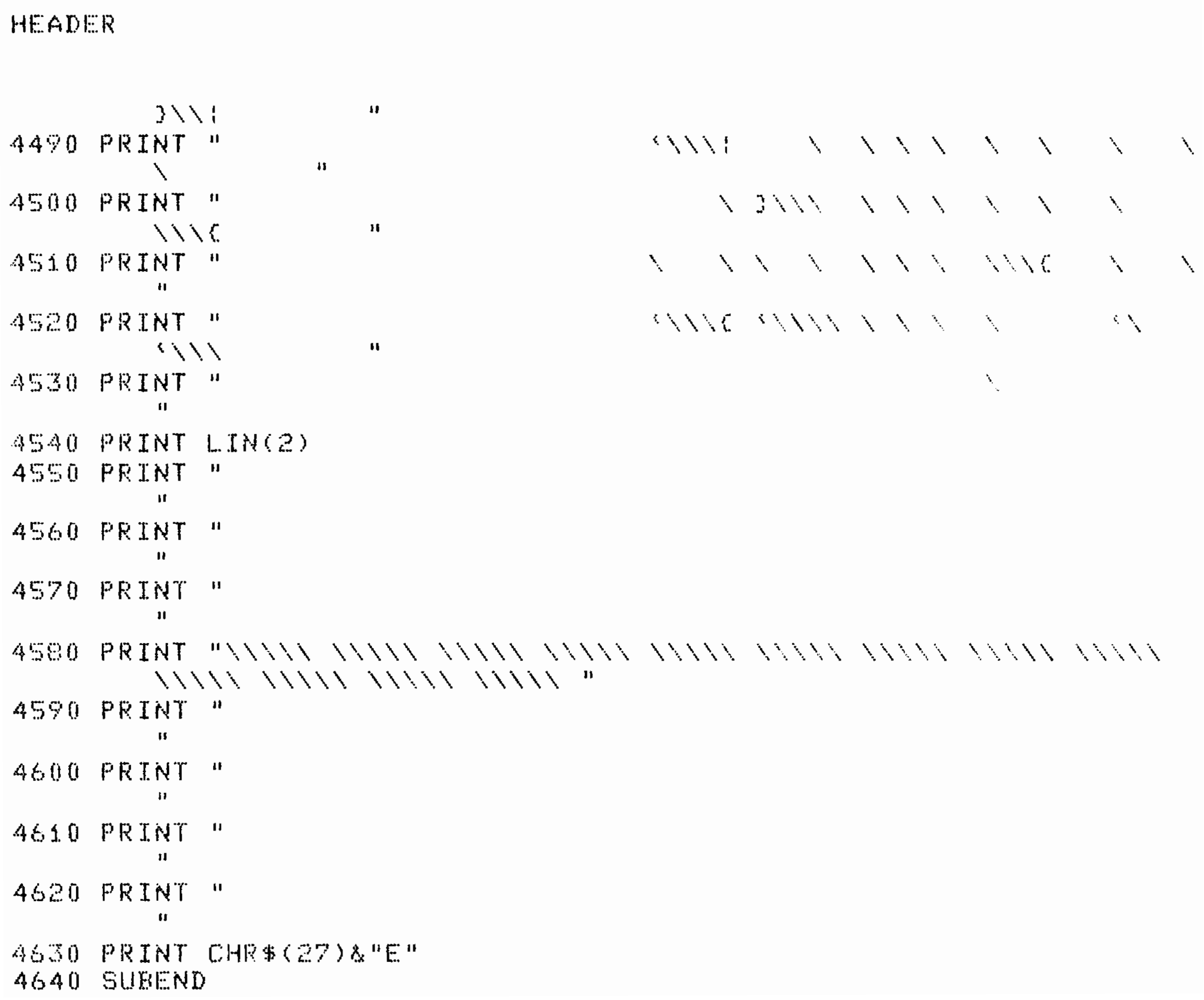
```
2870 RETURN
2880 Plus:X$="       \    \  \\\\\  \    \            "
2890 RETURN
2900 Com: X$="                                \    {  "
2910 RETURN
2920 And: X$="}\!  \ \  `\{  }\!  \ `\ \  \!`\\{`     "
2930 RETURN
2940 Apo: X$="  \    {                                "
2950 RETURN
2960 Minus: X$="               \\\\\                    "
2970 RETURN
2980 Per:   X$="                                \       "
2990 RETURN
3000 Div: X$="    }   }{  }{  }{  }{   {              "
3010 RETURN
3020 Perc:X$="      \  }   }{  }{  }{  }{   {  \      "
3030 RETURN
3040 Small:  IF (NUM(L$)<91) OR (NUM(L$)>127) THEN Notdefined
3050  ON NUM(L$)-90 GOSUB Lb,Bs,Rb,Ha,Ul,Ra,Sa,Sb,Sc,Sd,Se,Sf,Sg,Sh,
         Si,Sj,Sk,Sl,Sm,Sn,So,Sp,Sq,Sr,Ss,St,Su,Sv,Sw,Sx,Sy,Sz,Cl,Bu,
         Cr,Ti,De
3060  RETURN
3070 Lb:  X$=" \\\  \    \    \    \    \    \\\      "
3080 RETURN
3090 Rb:  X$=" \\\    \    \    \    \    \  \\\   .  "
3100 RETURN
3110 Bs:  X$="!    `!    `!    `!    `!    `          "
3120 RETURN
3130 Ha:  X$="      }!  }{`!                          "
3140 RETURN
3150 Ul:  X$="                                   \\\\\"
3160 RETURN
3170 Ra:  X$="  \    `                                "
3180 RETURN
3190 Cl:  X$=" }\\  \    \   }{   `!    \    \    `\\ "
3200 RETURN
3210 Cr:  X$=" \\!    \    \    `!   }{   \    \  \\{ "
3220 RETURN
3230 Bu:  X$="  \    \    \    \    \    \    \    \  "
3240 RETURN
3250 Ti:  X$="          }\! }{ `\{                    "
3260 RETURN
3270 De:  X$=" \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \      "
3280 RETURN
3290 Sa:  X$="          \\\!    \ }\\\ \  \ `\\\\     "
3300  RETURN
3310 Sb:  X$="\    \    \\\! \  \ \  \ \  \ \\\{      "
3320  RETURN
3330 Sc:  X$="          }\\\ \    \    \    `\\\      "
3340  RETURN
```

```
3350 Sd:  X$="   \    \ ]\\\ \  \ \  \ \  \ `\\\      "
3360  RETURN
3370 Se:  X$="          ]\\! \  \ \\\[ \    `\\\      "
3380  RETURN
3390 Sf:  X$=" ]\!  \ \  \   \\\   \    \    \        "
3400  RETURN
3410 Sg:  X$="          ]\\! \  \ \  \ `\\\ \  \ `\\[ "
3420  RETURN
3430 Sh:  X$="\    \    \\\\ \  \ \  \ \  \ \  \      "
3440  RETURN
3450 Si:  X$="  \        ]\    \    \    \    `\      "
3460  RETURN
3470 Sj:  X$="       \         \    \    \  \ \  `\[  "
3480  RETURN
3490 Sk:  X$="     \    \ ]  \][  \\   \`!  \ `!      "
3500  RETURN
3510 Sl:  X$=" ]\    \    \    \    \    \    `\      "
3520  RETURN
3530 Sm:  X$="          \]\\\\ \ \\ \ \\ \ \\ \ \     "
3540  RETURN
3550 Sn:  X$="          \]\\ \  \ \  \ \  \ \  \      "
3560  RETURN
3570 So:  X$="          ]\\! \  \ \  \ \  \ `\\[      "
3580  RETURN
3590 Sp:  X$="          \\\! \  \ \  \ \\\[ \    \    "
3600  RETURN
3610 Sq:  X$="          ]\\\ \  \ \  \ `\\\    \    `\"
3620  RETURN
3630 Sr:  X$="           \]\[ \[   \    \    \        "
3640  RETURN
3650 Ss:  X$="           ]\\  \    `\!    \  \\[      "
3660  RETURN
3670 St:  X$="       \   \\\   \    \    \    `\      "
3680  RETURN
3690 Su:  X$="          \  \ \  \ \  \ \  \ `\\[      "
3700  RETURN
3710 Sv:  X$="          \  \ \  \ \  \ `!][  `[       "
3720 RETURN
3730 Sw:  X$="          \ \ \\ \ \\ \ \\]\!\`[ `[     "
3740 RETURN
3750 Sx:  X$="          \  \ `!][ ][`! \  \ \  \      "
3760 RETURN
3770 Sy:  X$="          \  \ \  \ `!][  ][  ][        "
3780 RETURN
3790 Sz:  X$="          \\\\   ][  ][  ][   \\\\      "
3800 RETURN

Comment: Character undefined

3820 Notdefined:!
```

```
HEADER


3830 PRINTER IS 16
3840 Defined=0
3850 RETURN

Comment: Character converted to Header Characters

3870 Header_print:!
3880 Width=5
3890 Newletter: ! Comment: Start a new letter

3900 FOR Row=1 TO 8
3910 Lin$(Row)=Lin$(Row)&X$[1,Width]&" "
3920 X$=X$[Width+1,LEN(X$)]
3930 NEXT Row
3940 RETURN

Comment: Redefine characters for Header Character definition

3960 Headchars: PRINTER IS 0
3970 Setup$(1)=Esc$&"&18S"
3980 Setup$(2)=Esc$&"&lT"
3990 Setup$(3)=Esc$&"&n140c376p376q176r77s36t16u6v2W"
4000 Setup$(4)=Esc$&"&n175c2p6q16r37s76t176u376v376W"
4010 Setup$(5)=Esc$&"&n173c376p376q374r371s360t340u300v200W"
4020 Setup$(6)=Esc$&"&n174c200p300q340r361s370t374u376v376W"
4030 Setup$(7)=Esc$&"&n134c376p376q376r377s376t376u376v376W"
4040 PRINT Setup$(1);Setup$(2);Setup$(3);Setup$(4);Setup$(5);Setup$(6)
        ;Setup$(7)
4050 PRINTER IS 16
4060 RETURN
4070 END

        SUBPROGRAM PRODUCED BY THIS PROGRAM

4090 SUB Header
4100 Esc$=CHR$(27)
4110 PRINTER IS 0
4120 PRINT Esc$&"&18S";
4130 PRINT Esc$&"&lT";
4140 PRINT Esc$&"&n140c376p376q176r77s36t16u6v2W";
4150 PRINT Esc$&"&n175c2p6q16r37s76t176u376v376W";
4160 PRINT Esc$&"&n173c376p376q374r371s360t340u300v200W";
4170 PRINT Esc$&"&n174c200p300q340r361s370t374u376v376W";
4180 PRINT Esc$&"&n134c376p376q376r377s376t376u376v376W"
4190 PRINT "
       "
4200 PRINT "
       "
4210 PRINT "
       "
```

HEADER

```
4220 PRINT "\\\\\ \\\\\ \\\\\ \\\\\ \\\\\ \\\\\ \\\\\ \\\\\ \\\\\
        \\\\\ \\\\\ \\\\\ \\\\\ "
4230 PRINT "
        "
4240 PRINT "
        "
4250 PRINT "
        "
4260 PRINT "
        "
4270 PRINT LIN(2)
4280 PRINT "                    \   \   \                \
        \   "
4290 PRINT "                    <   \   \                \
        [   "
4300 PRINT "                        \   \ ]\\!  \\\!  ]\\\  ]\\!
        \]\[       "
4310 PRINT "                        \\\\\ \  \     \  \  \  \  \   \[
        "
4320 PRINT "                        \   \ \\\[  ]\\\  \  \  \\\[   \
        "
4330 PRINT "                        \   \ \     \  \  \  \  \      \
        "
4340 PRINT "                        \   \ <\\\  <\\\\ <\\\  <\\\   \
        "
4350 PRINT "
        "
4360 PRINT LIN(2)
4370 PRINT "                  \\\\!
        "
4380 PRINT "                  \   \
        "
4390 PRINT "                  \   \  \]\[ ]\\!  ]\\!   \]\[ \\\!
        \]\\\       "
4400 PRINT "                  \\\\[  \[   \  \  \  \   \[      \  \ \
        \       "
4410 PRINT "                  \      \    \  \  \  \   \    ]\\\  \ \
        \       "
4420 PRINT "                  \      \    \  \  <\\\   \    \  \  \ \
        \       "
4430 PRINT "                  \      \    <\\[  \  \   \    <\\\\ \ \
        \       "
4440 PRINT "                                    <\\[
        "
4450 PRINT LIN(2)
4460 PRINT "                        ]\\\!                    ]\
        "
4470 PRINT "                        \   \                     \
        "
4480 PRINT "                        \     \\\!  \]\\\  \\\!   \
```



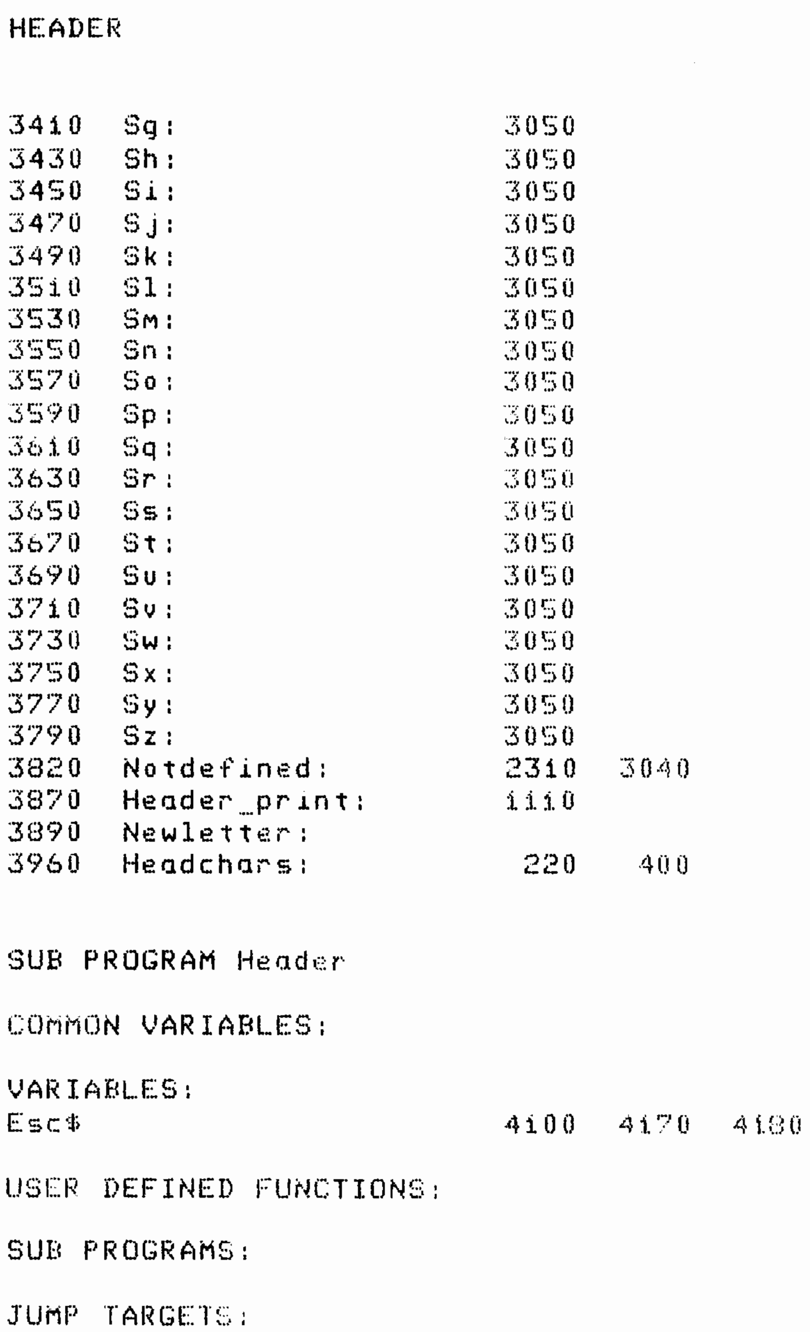
```
HEADER


        }\\|        "
4490 PRINT "                        `\\\|    \  \ \ \  \  \   \   \
        \        "
4500 PRINT "                            \ }\\\  \ \ \  \  \   \
        \\\{        "
4510 PRINT "                        \   \ \  \  \ \ \  \\\{   \   \
        "
4520 PRINT "                        `\\\{ `\\\\ \ \ \  \      `\
        `\\\        "
4530 PRINT "                                           \
        "
4540 PRINT LIN(2)
4550 PRINT "
        "
4560 PRINT "
        "
4570 PRINT "
        "
4580 PRINT "\\\\\ \\\\\ \\\\\ \\\\\ \\\\\ \\\\\ \\\\\ \\\\\ \\\\\
        \\\\\ \\\\\ \\\\\ \\\\\ "
4590 PRINT "
        "
4600 PRINT "
        "
4610 PRINT "
        "
4620 PRINT "
        "
4630 PRINT CHR$(27)&"E"
4640 SUBEND
```

# HEADER

```
HEADER

MAIN PROGRAM

COMMON VARIABLES:

VARIABLES:
A$                       30   340   350   360   420   430   440  1190
                             1200  1210  1280  1290  1300  1420  1430
                             1440
Defined                  40  1060  1100  3840
Esc$                     50   210  1670  3970  3980  3990  4000  4010
                             4020  4030
File                     60   470   830   960   970  1170  1490
File$                    70   540   550   590   600   610   760   810
                             1550  1580
I                        80   670   680   690  1050  1070  1120  1130
                             1140  1150  1350  1360  1370  1380
J                        90  1250  1260  1270
L$                      100  1070  1700  1710  1720  2300  2310  2680
                             2710  2720  3040  3050
Lin$(*)                 110  1140  1260  1360  1370  3910
Line                    120   240  1340  1360  1390  1400  1500  1510
M$                      130   980   990  1050  1070
N                       140  1720  1730
Row                     150  3900  3910  3930
Setup$(*)               160   680   700  3970  3980  3990  4000  4010
                             4020  4030  4040
Storage$                180   490   500   600   610   760
Width                   190  3880  3910  3920
X$                      170  1760  1780  1800  1820  1840  1860  1880
                             1900  1920  1940  1960  1980  2000  2020
                             2040  2060  2080  2100  2120  2140  2160
                             2180  2200  2220  2240  2260  2330  2350
                             2370  2390  2410  2430  2450  2470  2490
                             2510  2530  2550  2570  2590  2610  2630
                             2650  2690  2740  2760  2780  2800  2820
                             2840  2860  2880  2900  2920  2940  2960
                             2980  3000  3020  3070  3090  3110  3130
                             3150  3170  3190  3210  3230  3250  3270
                             3290  3310  3330  3350  3370  3390  3410
                             3430  3450  3470  3490  3510  3530  3550
                             3570  3590  3610  3630  3650  3670  3690
                             3710  3730  3750  3770  3790  3910  3920

USER DEFINED FUNCTIONS:

SUB PROGRAMS:
Header                  390

JUMP TARGETS:
```

```
HEADER


340                         380
390                         360
420                         350   460
470                         440
490                         520   780
540                         500   570   820
580                         550
730   File_error:           580
790   Duplicate:            740
830                         430
850   Message:              710
950   Input:               1030  1290  1440
1050                        990
1100                       1710
1120                       1100
1190                       1230
1250  Clear:               1170  1210
1280                       1320
1340  Store:               1200
1420                       1460
1480  Done:                1300  1400  1430
1640                       1490
1700  Chars:               1090
1750  A:                   1730
1780  B:                   1730
1800  C:                   1730
1820  D:                   1730
1840  E:                   1730
1860  F:                   1730
1880  G:                   1730
1900  H:                   1730
1920  I:                   1730
1940  J:                   1730
1960  K:                   1730
1980  L:                   1730
2000  M:                   1730
2020  N:                   1730
2040  O:                   1730
2060  P:                   1730
2080  Q:                   1730
2100  R:                   1730
2120  S:                   1730
2140  T:                   1730
2160  U:                   1730
2180  V:                   1730
2200  W:                   1730
2220  X:                   1730
2240  Y:                   1730
2260  Z:                   1730
2290  Specials:            1700
2330  Zero:                2310
```

```
2350   One:                  2310
2370   Two:                  2310
2390   Three:                2310
2410   Four:                 2310
2430   Five:                 2310
2450   Six:                  2310
2470   Seven:                2310
2490   Eight:                2310
2510   Nine:                 2310
2530   Ques:                 2310
2550   Eq:                   2310
2570   Col:                  2310
2590   Sem:                  2310
2610   Lt:                   2310
2630   Gt:                   2310
2650   Each:                 2310
2670   Puncs:                2300
2710                         2680
2740   Ex:                   2720
2760   Quo:                  2720
2780   Num:                  2720
2800   Dol:                  2720
2820   Lpar:                 2720
2840   Rpar:                 2720
2860   Ast:                  2720
2880   Plus:                 2720
2900   Com:                  2720
2920   And:                  2720
2940   Apo:                  2720
2960   Minus:                2720
2980   Per:                  2720
3000   Div:                  2720
3020   Perc:                 2720
3040   Small:                2710
3070   Lb:                   3050
3090   Rb:                   3050
3110   Bs:                   3050
3130   Ha:                   3050
3150   Ul:                   3050
3170   Ra:                   3050
3190   Cl:                   3050
3210   Cr:                   3050
3230   Bu:                   3050
3250   Ti:                   3050
3270   De:                   3050
3290   Sa:                   3050
3310   Sb:                   3050
3330   Sc:                   3050
3350   Sd:                   3050
3370   Se:                   3050
3390   Sf:                   3050
```



```
HEADER


3410  Sg:                 3050
3430  Sh:                 3050
3450  Si:                 3050
3470  Sj:                 3050
3490  Sk:                 3050
3510  Sl:                 3050
3530  Sm:                 3050
3550  Sn:                 3050
3570  So:                 3050
3590  Sp:                 3050
3610  Sq:                 3050
3630  Sr:                 3050
3650  Ss:                 3050
3670  St:                 3050
3690  Su:                 3050
3710  Sv:                 3050
3730  Sw:                 3050
3750  Sx:                 3050
3770  Sy:                 3050
3790  Sz:                 3050
3820  Notdefined:         2310  3040
3870  Header_print:       1110
3890  Newletter:
3960  Headchars:           220   400


SUB PROGRAM Header

COMMON VARIABLES:

VARIABLES:
Esc$                      4100  4170  4180

USER DEFINED FUNCTIONS:

SUB PROGRAMS:

JUMP TARGETS:
```

# Banner Program

## Object of Program

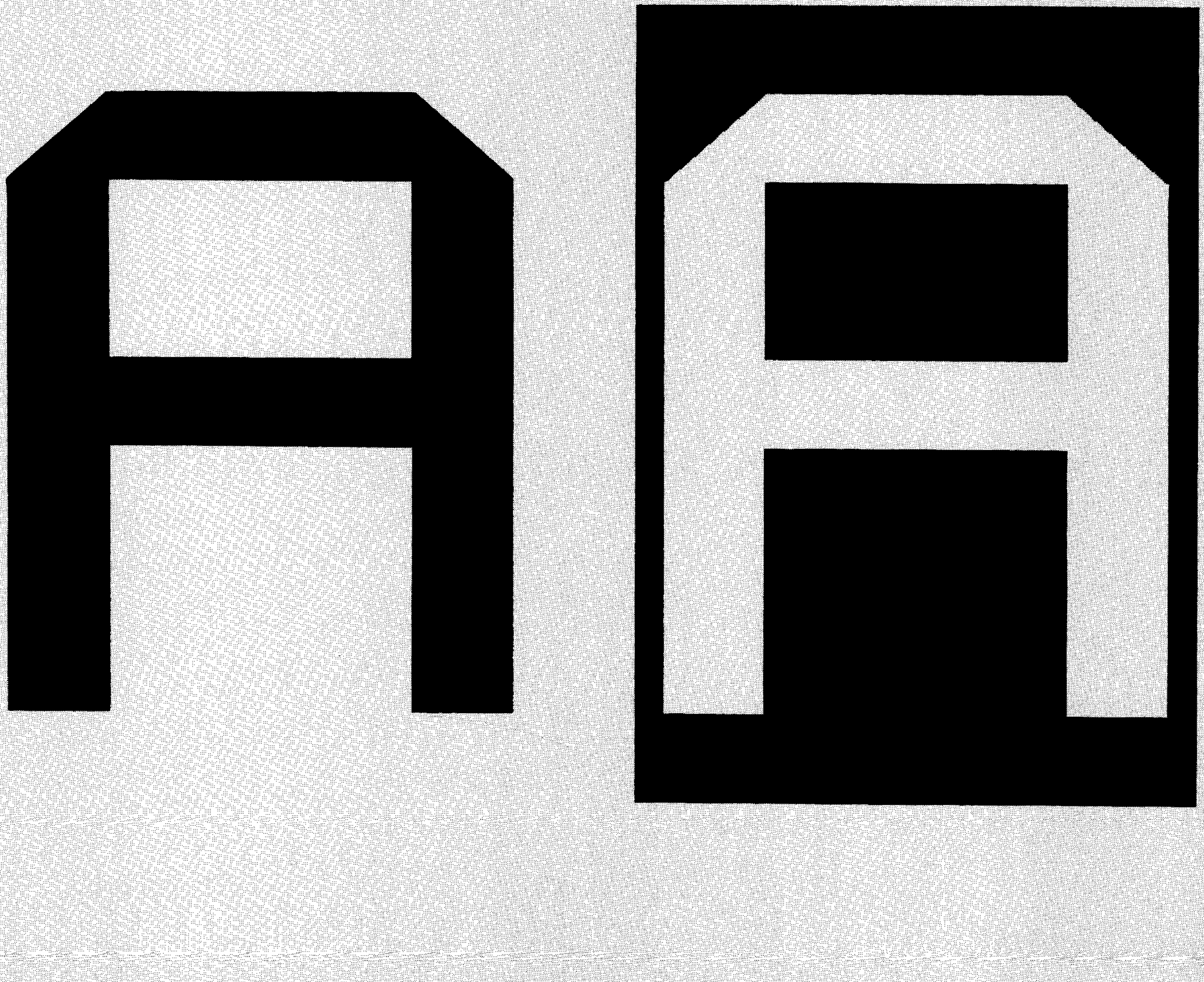
This program prints extremely large block characters using the internal thermal printer. Only one character will fit across the page on the internal printer and each character is printed sideways so that the result is a BANNER of indefinite length. A message of up to 160 characters can be printed at one time. This can be increased by simply adding another message to the end of the first.

This program is very similar to the HEADER program except that where HEADER characters are built from shapes made of one character, BANNER characters are built of shapes made of arrays of characters. For a discussion of character definitions see the section in the manual on the HEADER program. Where a triangular shape is a single character in the HEADER program it will be a 7X7 block of characters with the same triangular shape.

All characters printable on the thermal printer can be printed by the BANNER program.

A message may be printed in normal letters or in inverse, with dark around a light letter.

An "A" in BANNER characters in regular and inverse looks like this:



## User Instructions

The program BANNER is in the DATA file ''BANNER''

a. Type: GET ''BANNER''

b. Press: EXECUTE

The internal thermal line printer is necessary for this program.

### Data to be entered

You will be asked to enter the message, a string of characters, you wish to print out. When you have entered the string and pressed CONT the message will be printed out in BANNER characters. When the printing is done a message will be printed on the CRT to indicate completion. If you wish to print further messages, press RUN and the program will begin again.

# BANNER

```
Comment: Print a banner


Comment: Declarations

30    DIM M$[160]                       !Comment: Message string
40    DIM X$[160]                       !Comment: Temporary for
         character definition
50    DIM Ans$[3]                       !Comment: Answer to question


70    DIM L$(7)[7]                      !Comment: Lower left triangle
         solid
80    DIM R$(7)[7]                      !Comment: Upper left triangle
         solid
90    DIM T$(7)[7]                      !Comment: Lower right triangle
         solid
100   DIM U$(7)[7]                      !Comment: Upper right triangle
         solid
110   DIM B$(7)[7]                      !Comment: Block solid


130   DIM Esc$[1]                       !Comment: Escape character
         CHR$(27)
140   INTEGER Defined                   !Comment: Flag for defined
         characters
150   INTEGER Inv                       !Comment: Inverse flag

Comment: Set up for message

170   Esc$=CHR$(27)
180   GOSUB Headchars                   !Comment: Set up new characters
190   PRINTER IS 16
200   PRINT PAGE,LIN(2)
210 Message:  PRINTER IS 16
220   PRINT PAGE,LIN(2)
230   PRINT TAB(20);"BANNER PRINTING PROGRAM",LIN(3)
240   PRINT TAB(10);"DEFINED CHARACTERS:",LIN(1)
250   PRINT TAB(15);"A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,
         Z",LIN(1)
260   PRINT TAB(15);"a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,
         z",LIN(1)
270   PRINT TAB(15);"1,2,3,4,5,6,7,8,9,0",LIN(1)
280   PRINT TAB(15);"Math Symbols:  + - / * > < = ^ ",LIN(1)
290   PRINT TAB(15);"Puncuation:  ! @ # $ % & ( ) _ ~ ` ' { } [ ] | \
         : ; , . ? ";CHR$(34),LIN(1)
300   PRINT TAB(15);"Blank"

Comment: Inverse?
```

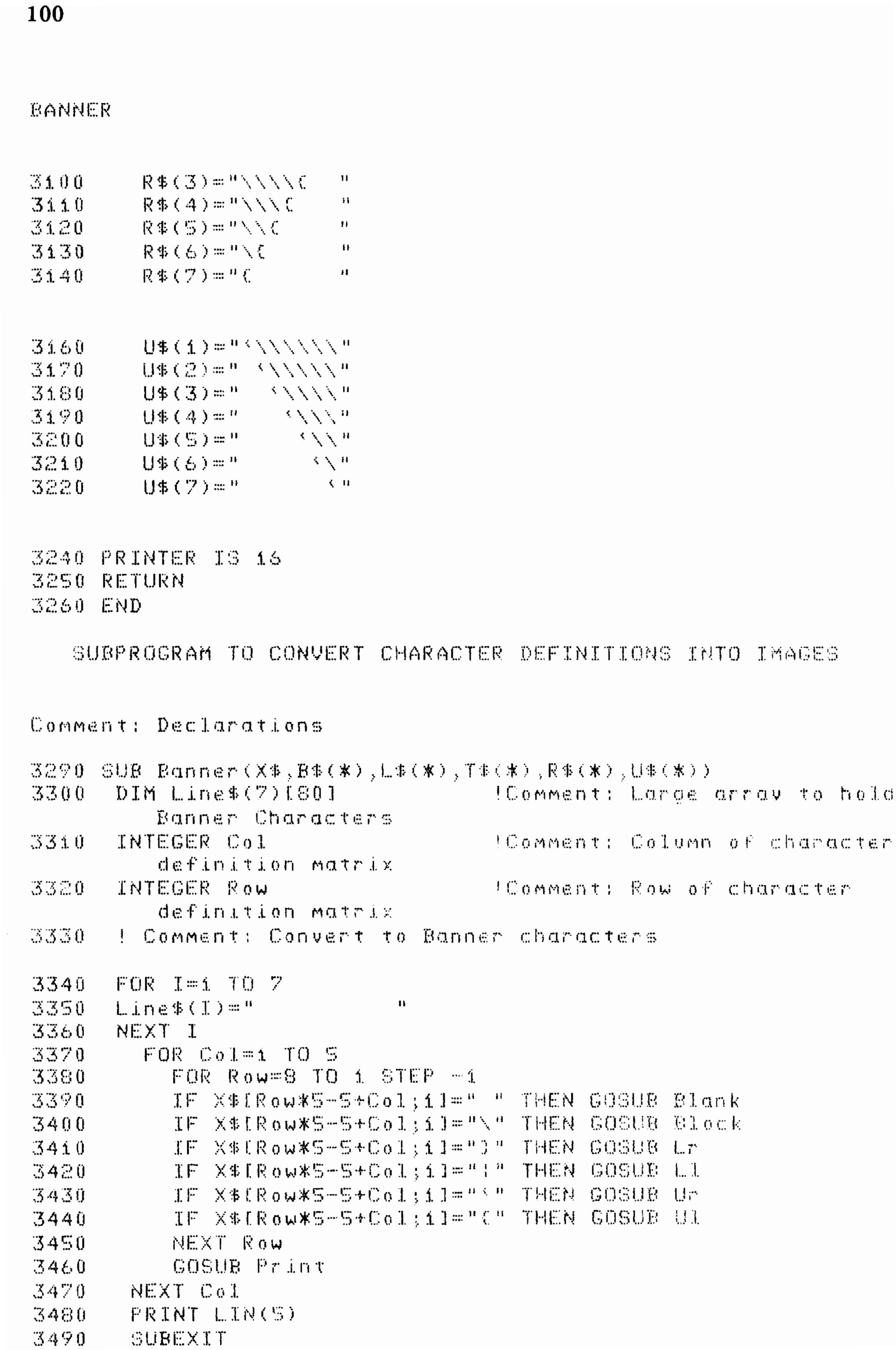
```
BANNER


320   Inv=0
330   INPUT "WOULD YOU LIKE THIS IN INVERSE CHARACTERS (YES/NO)",Ans$
340      IF Ans$[1,1]="Y" THEN 380
350      IF Ans$[1,1]="N" THEN 400
360      BEEP
370      GOTO 330
380      Inv=1

Comment: Enter message

400 Input: DISP "ENTER THE MESSAGE TO BE PRINTED";
410   LINPUT "",M$

Comment: Print message

430   DISP M$
440   FOR I=1 TO LEN(M$)
450   Defined=1
460   L$=M$[I;1]
470   PRINTER IS 0
480   GOSUB Chars
490   IF NOT Defined THEN GOTO 520
500   IF NOT Inv THEN CALL Banner(X$,B$(*),L$(*),T$(*),R$(*),U$(*))
510   IF Inv THEN CALL Inver(X$,B$(*),L$(*),T$(*),R$(*),U$(*))
520   NEXT I

Comment: Message completed

540   PRINT Esc$&"E"               !Comment: Clear printer of special
         features
550   PRINTER IS 16
560   PRINT PAGE,LIN(5);"                         MESSAGE PRINTED   "
570   DISP "   Done"
580   STOP

Comment: Banner Character definitions

600 Chars: IF (NUM(L$)<65) OR (NUM(L$)>90) THEN GOSUB Specials
610   IF (NUM(L$)<65) OR (NUM(L$)>90) THEN 490
620   N=NUM(L$)-64
630   ON N GOSUB A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z
640   RETURN
650 A: !
660   X$=")\\\!\   \\   \\\\\\\\   \\   \\   \     "
670   RETURN
680 B:  X$="\\\\! \  \ \  \ \\\( \  \ \  \\\\\(     "
690   RETURN
700 C: X$=")\\\!\   \\    \    \    \   \'\\\(     "
710   RETURN
720 D: X$="\\\\!\   \\   \\   \\   \\   \\\\\(     "
730  RETURN
740 E: X$="\\\\\\    \    \\\  \    \    \\\\\     "
750  RETURN
760 F: X$="\\\\\\    \    \\\  \    \    \         "
770  RETURN
780 G: X$=")\\\¦\   `\    \    \ \\\\   \`\\\(     "
790  RETURN
800 H: X$="\   \\   \\   \\\\\\\   \\   \\   \     "
810  RETURN
820 I: X$=" \\\   \    \    \    \    \   \\\      "
830  RETURN
840 J: X$="  \\\   \    \    \    \ ¦  \ `\\(      "
850  RETURN
860 K: X$="\  )(\ )( \)(  \¦   \`¦  \ `¦ \  `¦     "
870  RETURN
880 L: X$="\    \    \    \    \    \\\\\     "
890  RETURN
900 M: X$="\¦ )\\`\(\\ \ \\   \\   \\   \\   \     "
910  RETURN
920 N: X$="\   \\¦  \\`¦ \\ `¦\\  `\\   \\   \     "
930  RETURN
940 O: X$=")\\\¦\   \\   \\   \\   \\   \`\\\(     "
950  RETURN
960 P: X$="\\\\¦\   \\   \\\\\(\    \    \         "
970  RETURN
980 Q: X$=")\\\¦\   \\   \\   \\ ¦ \\ `¦\`\\\(   ` "
990  RETURN
1000 R:X$="\\\\¦\   \\   \\\\\(\  `¦\   \\   \     "
1010 RETURN
1020 S:X$=")\\\¦\   \\    `\\\¦    \\   \`\\\(     "
1030 RETURN
1040 T:X$="\\\\\  \    \    \    \    \    \       "
1050 RETURN
1060 U:X$="\   \\   \\   \\   \\   \\   \`\\\(     "
1070 RETURN
1080 V:X$="\   \\   \\   \\   \\   \`¦ )( `\(      "
1090 RETURN
1100 W:X$="\   \\   \\   \\   \\ \ \\)\¦\`( `(     "
1110 RETURN
1120 X:X$="\   \\   \`¦ )( \\\ )( `¦\   \\   \     "
1130 RETURN
1140 Y:X$="\   \\   \\   \`\\\(  \    \    \       "
1150 RETURN
1160 Z:X$="\\\\\    \   )( )\( )(   \    \\\\\     "
1170 RETURN
1180 PAUSE
1190 Specials: !
1200 IF (NUM(L$)>64) OR (NUM(L$)<48) THEN Puncs
1210 ON NUM(L$)-46 GOSUB Notdefined,Zero,One,Two,Three,Four,Five,Six,
         Seven.Eight,Nine,Col,Sem,Lt,Eq,Gt,Ques,Each
1220 RETURN
1230 Zero:X$="]\\\!\   \\  ]\\ ][\\][ \\[  \^\\\[     "
1240 RETURN
1250 One:X$=" ]\    \    \    \    \    \   \\\      "
1260 RETURN
1270 Two:X$="]\\\!\   \    \ ]\\[][   \    \\\\\     "
1280 RETURN
1290 Three:X$="]\\\!\   \    \  \\!    \\   \^\\\[     "
1300 RETURN
1310 Four: X$="\  \ \  \ \  \ \\\\\   \    \    \      "
1320 RETURN
1330 Five:X$="\\\\\\    \    \\\      \\   \^\\\[     "
1340 RETURN
1350 Six: X$="\\\\\\   \\    \\\\\\   \\   \\\\\\     "
1360 RETURN
1370 Seven:X$="\\\\\\   \   ][  ][   \    \    \       "
1380 RETURN
1390 Eight:X$="\\\\\\   \\   \\\\\\\   \\   \\\\\\     "
1400 RETURN
1410 Nine: X$="\\\\\\   \\   \\\\\\    \    \    \     "
1420 RETURN
1430 Ques:X$="]\\\!\   \    \ ]\\[ \         \        "
1440 RETURN
1450 Eq: X$="          \\\\\     \\\\\               "
1460 RETURN
1470 Col:X$="            \         \                 "
1480 RETURN
1490 Sem:X$="                           \         [  "
1500 RETURN
1510 Lt: X$="        ]   ][  ][   ^!    ^!    ^      "
1520 RETURN
1530 Gt: X$="      !    ^!    ^!   ][  ][   [        "
1540 RETURN
1550 Each:X$="]\\\!\   \\ ]\\\ \ \\ ^\[\   ]^\\\[     "
1560 RETURN
1570 Puncs:!
1580 IF L$<>" " THEN 1610
1590 X$="                                        "
1600 RETURN
1610 IF (NUM(L$)<33) OR (NUM(L$)>47) THEN Small
1620 ON NUM(L$)-32 GOSUB Ex,Quo,Num,Dol,Perc,And,Apo,Rpar,Lpar,Ast,
        Plus,Com,Minus,Per,Div
1630 RETURN
1640 Ex: X$="  \    \    \    \    \         \       "
1650 RETURN
1660 Quo:X$=" \ \  ^ ^                               "
1670 RETURN
1680 Num:X$="      \ \ \\\\\ \ \ \\\\\ \ \           "
1690 RETURN
1700 Dol:X$="  \  ]\\\\\ \  ^\\\!  \ \\\\\[  \       "
```

```
1710 RETURN
1720 Lpar:X$="  `¦    `¦    \    \    \   )(  )(      "
1730 RETURN
1740 Rpar:X$=" )(  )(   \    \    \    `¦    `¦       "
1750 RETURN
1760 Ast: X$="     `¦ )( `\( \\\\\ )\¦ )( `¦          "
1770 RETURN
1780 Plus:X$="       \    \  \\\\\  \    \            "
1790 RETURN
1800 Com: X$="                                \    (  "
1810 RETURN
1820 And: X$=")\¦  \ \  `\(  )\¦  \ `\ \  \¦`\\(`     "
1830 RETURN
1840 Apo: X$="  \    (                                "
1850 RETURN
1860 Minus: X$="               \\\\\                    "
1870 RETURN
1880 Per:   X$="                                \       "
1890 RETURN
1900 Div: X$="         )   )(  )(  )(  )(   (         "
1910 RETURN
1920 Perc:X$="      \  )   )(  )(  )(  )(   (  \      "
1930 RETURN
1940 Small:  IF (NUM(L$)<91) OR (NUM(L$)>127) THEN Notdefined
1950  ON NUM(L$)-90 GOSUB Lb,Bs,Rb,Ha,Ul,Ra,Sa,Sb,Sc,Sd,Se,Sf,Sg,Sh,
         Si,Sj,Sk,Sl,Sm,Sn,So,Sp,Sq,Sr,Ss,St,Su,Sv,Sw,Sx,Sy,Sz,Cl,Bu,
         Cr,Ti,De
1960  RETURN
1970 Lb:  X$=" \\\  \    \    \    \    \    \\\      "
1980 RETURN
1990 Rb:  X$=" \\\    \    \    \    \    \  \\\      "
2000 RETURN
2010 Bs:  X$="¦    `¦    `¦    `¦    `¦    `          "
2020 RETURN
2030 Ha:  X$="      )¦  )(`¦                          "
2040 RETURN
2050 Ul:  X$="                                   \\\\\"
2060 RETURN
2070 Ra:  X$="  \    `                                "
2080 RETURN
2090 Cl:  X$=" )\\  \    \   )(   `¦    \    \    `\\ "
2100 RETURN
2110 Cr:  X$=" \\¦    \    \    `¦   )(   \    \  \\( "
2120 RETURN
2130 Bu:  X$="  \    \    \    \    \    \    \    \  "
2140 RETURN
2150 Ti:  X$="          )\¦ )( `\(                    "
2160 RETURN
2170 De:  X$=" \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \      "
2180 RETURN
```

BANNER

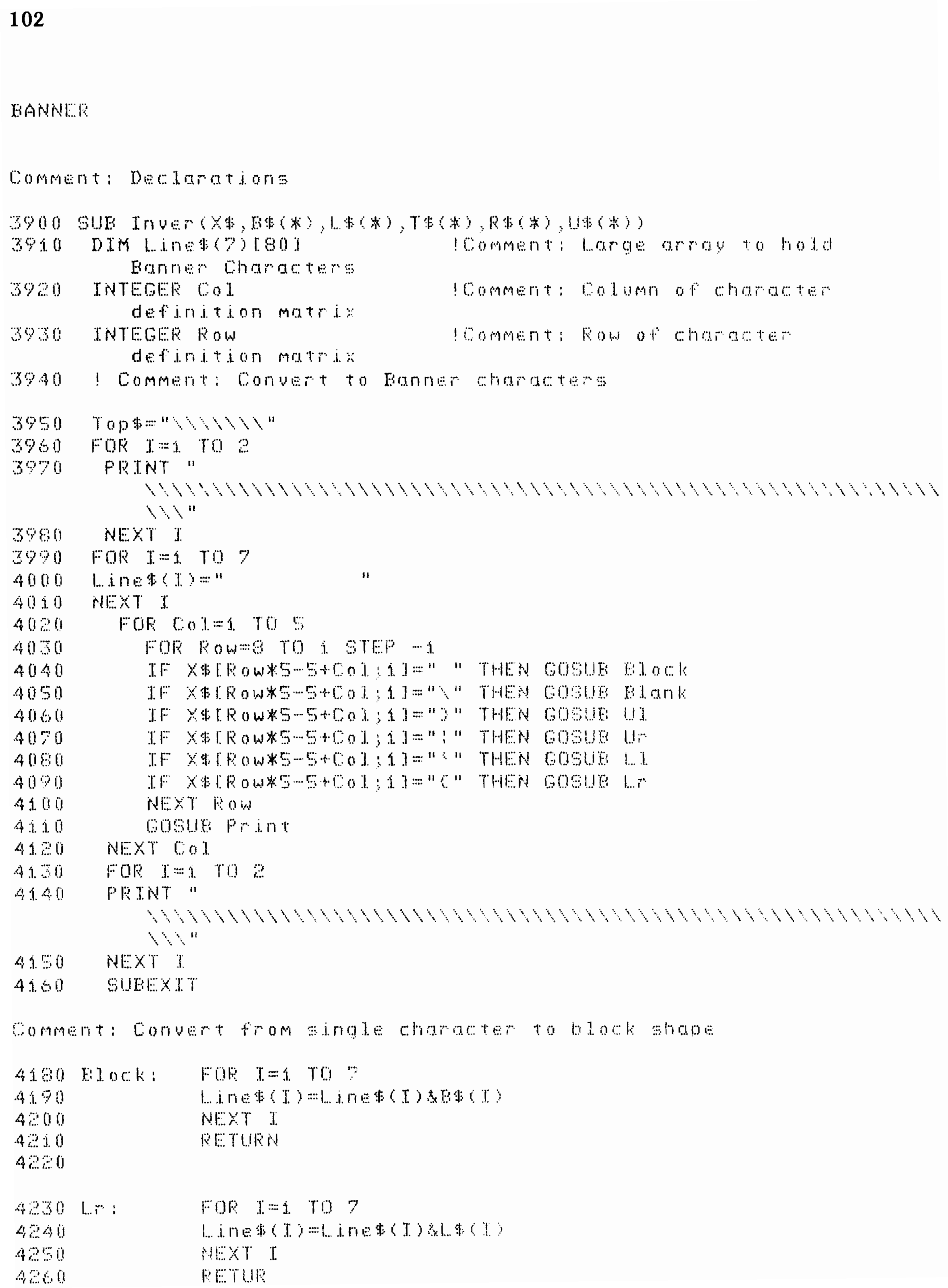
```
2190 Sa:  X$="          \\\!    \ 3\\\ \  \ '\\\\     "
2200  RETURN
2210 Sb:  X$="\    \    \\\! \  \ \  \ \  \ \\\C      "
2220  RETURN
2230 Sc:  X$="          3\\\ \    \    \    '\\\      "
2240  RETURN
2250 Sd:  X$="   \    \ 3\\\ \  \ \  \ \  \ '\\\      "
2260  RETURN
2270 Se:  X$="          3\\! \  \ \\\C \    '\\\      "
2280  RETURN
2290 Sf:  X$=" 3\!  \ \  \   \\\   \    \    \        "
2300  RETURN
2310 Sq:  X$="          3\\! \  \ \  \ '\\\ \  \ '\\C "
2320  RETURN
2330 Sh:  X$="\    \    \\\\ \  \ \  \ \  \ \  \      "
2340  RETURN
2350 Si:  X$="  \        3\    \    \    \    '\      "
2360  RETURN
2370 Sj:  X$="       \         \    \    \  \ \  '\C  "
2380  RETURN
2390 Sk:  X$="     \    \ 3  \3C  \\   \'!  \ '!      "
2400  RETURN
2410 Sl:  X$=" 3\    \    \    \    \    \    \\      "
2420  RETURN
2430 Sm:  X$="          \3\\\\ \ \\ \ \\ \ \\ \ \     "
2440  RETURN
2450 Sn:  X$="          \3\\ \  \ \  \ \  \ \  \      "
2460  RETURN
2470 So:  X$="          3\\! \  \ \  \ \  \ '\\C      "
2480  RETURN
2490 Sp:  X$="          \\\! \  \ \  \ \\\C \    \    "
2500  RETURN
2510 Sq:  X$="          3\\\ \  \ \  \ '\\\    \    '\"
2520  RETURN
2530 Sr:  X$="           \3\C \C   \    \    \        "
2540  RETURN
2550 Ss:  X$="           3\\  \    '\!    \  \\C      "
2560  RETURN
2570 St:  X$="       \   \\\   \    \    \    '\      "
2580  RETURN
2590 Su:  X$="          \  \ \  \ \  \ \  \ '\\C      "
2600  RETURN
2610 Sv:  X$="          \  \ \  \ \  \ '!3C  'C       "
2620 RETURN
2630 Sw:  X$="          \ \ \\ \ \\ \ \\3\!\'C 'C     "
2640 RETURN
2650 Sx:  X$="          \  \ '!3C 3C'! \  \ \  \      "
2660 RETURN
2670 Sy:  X$="          \  \ \  \ '!3C  3C  3C        "
2680 RETURN
```

```
2690 Sz:  X$="          \\\\   }{  }{  }{   \\\\      "
2700 RETURN

Comment:  Character not defined

2720 Notdefined:!
2730 PRINTER IS 16
2740 Defined=0
2750 RETURN

Comment: Set definitions for printing Banner characters

2770 Headchars: PRINTER IS 0
2780 DIM Setup$(7)[40]
2790 Setup$(1)=Esc$&"&l8S"
2800 Setup$(2)=Esc$&"&lT"
2810 Setup$(3)=Esc$&"&n140c376p376q176r77s36t16u6v2W"
2820 Setup$(4)=Esc$&"&n175c2p6q16r37s76t176u376v376W"
2830 Setup$(5)=Esc$&"&n173c376p376q374r371s360t340u300v200W"
2840 Setup$(6)=Esc$&"&n174c200p300q340r361s370t374u376v376W"
2850 Setup$(7)=Esc$&"&n134c376p376q376r377s376t376u376v376W"
2860 PRINT Setup$(1);Setup$(2);Setup$(3);Setup$(4);Setup$(5);Setup$(6)
        ;Setup$(7)


2880    FOR I=1 TO 7
2890    B$(I)="\\\\\\\"
2900    NEXT I


2920    L$(1)="|      "
2930    L$(2)="\|     "
2940    L$(3)="\\|    "
2950    L$(4)="\\\|   "
2960    L$(5)="\\\\|  "
2970    L$(6)="\\\\\| "
2980    L$(7)="\\\\\\|"


3000    T$(1)="      }"
3010    T$(2)="     }\"
3020    T$(3)="    }\\"
3030    T$(4)="   }\\\"
3040    T$(5)="  }\\\\"
3050    T$(6)=" }\\\\\"
3060    T$(7)="}\\\\\\"


3080    R$(1)="\\\\\\{"
3090    R$(2)="\\\\\{ "
```



```
BANNER


3100     R$(3)="\\\\{   "
3110     R$(4)="\\\{    "
3120     R$(5)="\\{     "
3130     R$(6)="\{      "
3140     R$(7)="{       "


3160     U$(1)="`\\\\\\\"
3170     U$(2)=" `\\\\\\"
3180     U$(3)="  `\\\\\"
3190     U$(4)="   `\\\\"
3200     U$(5)="    `\\\"
3210     U$(6)="     `\\"
3220     U$(7)="      `"


3240 PRINTER IS 16
3250 RETURN
3260 END

   SUBPROGRAM TO CONVERT CHARACTER DEFINITIONS INTO IMAGES


Comment: Declarations

3290 SUB Banner(X$,B$(*),L$(*),T$(*),R$(*),U$(*))
3300  DIM Line$(7)[80]              !Comment: Large array to hold
         Banner Characters
3310  INTEGER Col                   !Comment: Column of character
         definition matrix
3320  INTEGER Row                   !Comment: Row of character
         definition matrix
3330  ! Comment: Convert to Banner characters

3340  FOR I=1 TO 7
3350  Line$(I)="          "
3360  NEXT I
3370    FOR Col=1 TO 5
3380      FOR Row=8 TO 1 STEP -1
3390      IF X$[Row*5-5+Col;1]=" " THEN GOSUB Blank
3400      IF X$[Row*5-5+Col;1]="\" THEN GOSUB Block
3410      IF X$[Row*5-5+Col;1]="}" THEN GOSUB Lr
3420      IF X$[Row*5-5+Col;1]="|" THEN GOSUB Ll
3430      IF X$[Row*5-5+Col;1]="`" THEN GOSUB Ur
3440      IF X$[Row*5-5+Col;1]="{" THEN GOSUB Ul
3450      NEXT Row
3460      GOSUB Print
3470   NEXT Col
3480   PRINT LIN(5)
3490   SUBEXIT
```

```
Comment: Convert from single character to block shape

3510 Block:    FOR I=1 TO 7
3520           Line$(I)=Line$(I)&B$(I)
3530           NEXT I
3540           RETURN
3550

3560 Lr:       FOR I=1 TO 7
3570           Line$(I)=Line$(I)&L$(I)
3580           NEXT I
3590           RETURN
3600

3610 Ll:       FOR I=1 TO 7
3620           Line$(I)=Line$(I)&R$(I)
3630           NEXT I
3640           RETURN
3650

3660 Ur:       FOR I=1 TO 7
3670           Line$(I)=Line$(I)&T$(I)
3680           NEXT I
3690           RETURN

3710 Ul:       FOR I=1 TO 7
3720           Line$(I)=Line$(I)&U$(I)
3730           NEXT I
3740           RETURN


3760 Blank:    FOR I=1 TO 7
3770           Line$(I)=Line$(I)&"       "
3780           NEXT I
3790           RETURN


3810 Print:      FOR I=1 TO 7
3820             PRINT Line$(I
3830             Line$(I)="          "
3840             NEXT I
3850             RETURN

3870 SUBEND

   SUBPROGRAM TO CONVERT TO INVERSE IMAGES
```



```
BANNER

Comment: Declarations

3900 SUB Inver(X$,B$(*),L$(*),T$(*),R$(*),U$(*))
3910  DIM Line$(7)[80]              !Comment: Large array to hold
         Banner Characters
3920  INTEGER Col                   !Comment: Column of character
         definition matrix
3930  INTEGER Row                   !Comment: Row of character
         definition matrix
3940  ! Comment: Convert to Banner characters

3950  Top$="\\\\\\\\"
3960  FOR I=1 TO 2
3970   PRINT "
          \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
          \\\"
3980   NEXT I
3990  FOR I=1 TO 7
4000  Line$(I)="          "
4010  NEXT I
4020    FOR Col=1 TO 5
4030      FOR Row=8 TO 1 STEP -1
4040      IF X$[Row*5-5+Col;1]=" " THEN GOSUB Block
4050      IF X$[Row*5-5+Col;1]="\" THEN GOSUB Blank
4060      IF X$[Row*5-5+Col;1]=")" THEN GOSUB Ul
4070      IF X$[Row*5-5+Col;1]="!" THEN GOSUB Ur
4080      IF X$[Row*5-5+Col;1]="`" THEN GOSUB Ll
4090      IF X$[Row*5-5+Col;1]="(" THEN GOSUB Lr
4100      NEXT Row
4110      GOSUB Print
4120   NEXT Col
4130   FOR I=1 TO 2
4140   PRINT "
          \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
          \\\"
4150   NEXT I
4160   SUBEXIT

Comment: Convert from single character to block shape

4180 Block:    FOR I=1 TO 7
4190           Line$(I)=Line$(I)&B$(I)
4200           NEXT I
4210           RETURN
4220

4230 Lr:       FOR I=1 TO 7
4240           Line$(I)=Line$(I)&L$(I)
4250           NEXT I
4260           RETUR
```

```
4270

4280 L1:         FOR I=1 TO 7
4290             Line$(I)=Line$(I)&R$(I)
4300             NEXT I
4310             RETURN
4320

4330 Ur:         FOR I=1 TO 7
4340             Line$(I)=Line$(I)&T$(I)
4350             NEXT I
4360             RETURN

4380 Ul:         FOR I=1 TO 7
4390             Line$(I)=Line$(I)&U$(I)
4400             NEXT I
4410             RETUR


4430 Blank:      FOR I=1 TO 7
4440             Line$(I)=Line$(I)&"       "
4450             NEXT I
4460             RETURN


4480 Print:        FOR I=1 TO 7
4490               PRINT Line$(I)&Top$
4500               Line$(I)="          "
4510               NEXT I
4520               RETURN


4540 SUBEND
```

# BANNER

```
BANNER

MAIN PROGRAM

COMMON VARIABLES:

VARIABLES:
Ans$                        50   330   340   350
B$(*)                      110   500   510  2890
Defined                    140   450   490  2740
Esc$                       130   170   540  2790  2800  2810  2820  2830
                                2840  2850
I                          440   460   520  2880  2890  2900
Inv                        150   320   380   500   510
L$                         460   600   610   620  1200  1210  1580  1610
                                1620  1940  1950
L$(*)                       70   500   510  2920  2930  2940  2950  2960
                                2970  2980
M$                          30   410   430   440   460
N                          620   630
R$(*)                       80   500   510  3080  3090  3100  3110  3120
                                3130  3140
Setup$(*)                 2780  2790  2800  2810  2820  2830  2840  2850
                                2860
T$(*)                       90   500   510  3000  3010  3020  3030  3040
                                3050  3060
U$(*)                      100   500   510  3160  3170  3180  3190  3200
                                3210  3220
X$                          40   500   510   660   680   700   720   740
                                 760   780   800   820   840   860   880
                                 900   920   940   960   980  1000  1020
                                1040  1060  1080  1100  1120  1140  1160
                                1230  1250  1270  1290  1310  1330  1350
                                1370  1390  1410  1430  1450  1470  1490
                                1510  1530  1550  1590  1640  1660  1680
                                1700  1720  1740  1760  1780  1800  1820
                                1840  1860  1880  1900  1920  1970  1990
                                2010  2030  2050  2070  2090  2110  2130
                                2150  2170  2190  2210  2230  2250  2270
                                2290  2310  2330  2350  2370  2390  2410
                                2430  2450  2470  2490  2510  2530  2550
                                2570  2590  2610  2630  2650  2670  2690

USER DEFINED FUNCTIONS:

SUB PROGRAMS:
Banner                     500
Inver                      510

JUMP TARGETS:
210   Message:
```

```
BANNER


330                         370
380                         340
400   Input:                350
490                         610
520                         490
600   Chars:                480
650   A:                    630
680   B:                    630
700   C:                    630
720   D:                    630
740   E:                    630
760   F:                    630
780   G:                    630
800   H:                    630
820   I:                    630
840   J:                    630
860   K:                    630
880   L:                    630
900   M:                    630
920   N:                    630
940   O:                    630
960   P:                    630
980   Q:                    630
1000  R:                    630
1020  S:                    630
1040  T:                    630
1060  U:                    630
1080  V:                    630
1100  W:                    630
1120  X:                    630
1140  Y:                    630
1160  Z:                    630
1190  Specials:             600
1230  Zero:                1210
1250  One:                 1210
1270  Two:                 1210
1290  Three:               1210
1310  Four:                1210
1330  Five:                1210
1350  Six:                 1210
1370  Seven:               1210
1390  Eight:               1210
1410  Nine:                1210
1430  Ques:                1210
1450  Eq:                  1210
1470  Col:                 1210
1490  Sem:                 1210
1510  Lt:                  1210
1530  Gt:                  1210
1550  Each:                1210
1570  Puncs:               1200
```

```
1610                 1580
1640  Ex:            1620
1660  Quo:           1620
1680  Num:           1620
1700  Dol:           1620
1720  Lpar:          1620
1740  Rpar:          1620
1760  Ast:           1620
1780  Plus:          1620
1800  Com:           1620
1820  And:           1620
1840  Apo:           1620
1860  Minus:         1620
1880  Per:           1620
1900  Div:           1620
1920  Perc:          1620
1940  Small:         1610
1970  Lb:            1950
1990  Rb:            1950
2010  Bs:            1950
2030  Ha:            1950
2050  Ul:            1950
2070  Ra:            1950
2090  Cl:            1950
2110  Cr:            1950
2130  Bu:            1950
2150  Ti:            1950
2170  De:            1950
2190  Sa:            1950
2210  Sb:            1950
2230  Sc:            1950
2250  Sd:            1950
2270  Se:            1950
2290  Sf:            1950
2310  Sg:            1950
2330  Sh:            1950
2350  Si:            1950
2370  Sj:            1950
2390  Sk:            1950
2410  Sl:            1950
2430  Sm:            1950
2450  Sn:            1950
2470  So:            1950
2490  Sp:            1950
2510  Sq:            1950
2530  Sr:            1950
2550  Ss:            1950
2570  St:            1950
2590  Su:            1950
2610  Sv:            1950
2630  Sw:            1950
```

```
BANNER


2650  Sx:                 1950
2670  Sy:                 1950
2690  Sz:                 1950
2720  Notdefined:         1210  1940
2770  Headchars:           180


SUB PROGRAM Banner

COMMON VARIABLES:

VARIABLES:
B$(*)                     3290  3520
Col                       3310  3370  3390  3400  3410  3420  3430  3440
                                3470
I                         3340  3350  3360  3510  3520  3530  3560  3570
                                3580  3610  3620  3630  3660  3670  3680
                                3710  3720  3730  3760  3770  3780  3810
                                3820  3830  3840
L$(*)                     3290  3570
Line$(*)                  3300  3350  3520  3570  3620  3670  3720  3770
                                3820  3830
R$(*)                     3290  3620
Row                       3320  3380  3390  3400  3410  3420  3430  3440
                                3450
T$(*)                     3290  3670
U$(*)                     3290  3720
X$                        3290  3390  3400  3410  3420  3430  3440

USER DEFINED FUNCTIONS:

SUB PROGRAMS:

JUMP TARGETS:
3510  Block:              3400
3560  Lr:                 3410
3610  Ll:                 3420
3660  Ur:                 3430
3710  Ul:                 3440
3760  Blank:              3390
3810  Print:              3460


SUB PROGRAM Inver

COMMON VARIABLES:

VARIABLES:
B$(*)                     3900  4190
Col                       3920  4020  4040  4050  4060  4070  4080  4090
                                4120
```

```
                                                       BANNER


I                          3960  3980  3990  4000  4010  4130  4150  4180
                                 4190  4200  4230  4240  4250  4280  4290
                                 4300  4330  4340  4350  4380  4390  4400
                                 4430  4440  4450  4480  4490  4500  4510
L$(*)                      3900  4240
Line$(*)                   3910  4000  4190  4240  4290  4340  4390  4440
                                 4490  4500
R$(*)                      3900  4290
Row                        3930  4030  4040  4050  4060  4070  4080  4090
                                 4100
T$(*)                      3900  4340
Top$                       3950  4490
U$(*)                      3900  4390
X$                         3900  4040  4050  4060  4070  4080  4090

USER DEFINED FUNCTIONS:

SUB PROGRAMS:

JUMP TARGETS:
4180  Block:               4040
4230  Lr:                  4090
4280  Ll:                  4080
4330  Ur:                  4070
4380  Ul:                  4060
4430  Blank:               4050
4480  Print:               4110
```

# File Manager Program

## Object of Program

This program allows you to manage the files on a tape cartridge or 9885 Flexible Disk. Three separate functions can be performed:

1) Availability tables for the mass storage device can be built.
2) A selective PURGE of the files may be performed.
3) A selective COPY of files to another mass storage device can be made.

The availability tables are built by first reading in the directory of the mass storage device selected. If the device is a flexible disk the files are reordered to reflect sequential track and record addresses. The directory is then scanned for empty files on the tape or flexible disk. The altered directory, with the empty files clearly marked and addressed, is then displayed. A hard copy of this directory can be obtained. The program will then display a list of only the empty files, this time ordered by size, largest to smallest. This list may also be dumped to a printer.

The selective purge portion of the program allows you to indicate which files on the mass storage medium should be purged. When the files are marked to your satisfaction all the files indicated will then be purged. If the file is protected or the file is of the wrong type (there is a question mark following the file type) then the files will NOT be purged and a message to this effect will be displayed.

The copy portion of program allows you to select any number of files from the selected mass storage medium to be copied to a second medium. A list of all files available will be displayed. You may then mark the files you want copied. When you are satisfied with the files marked they will be copied to the second medium. Should the files be of the wrong type or should the second medium not have enough room to hold a file, you will be notified.

## User Instructions

The program FILE MANAGER is in DATA file "FILEMN".

a. Type: GET "FILEMN"
b. Press: EXECUTE

This program requires a 9845B since it uses selective addressing and the CAT TO statement.

### Data to be entered

Initially you will be asked to enter the mass storage unit specifier of the device you wish to manipulate (e.g.T15 or F8). When this has been entered the directory of the medium will be read in and, if necessary, reordered to show the files with sequential addresses.

When the directory is ready a list of the commands will be displayed. A cursor will be positioned to the left of the first command. This cursor may be moved up and down by using the up and down arrows in the Display Key section of the keyboard. When the cursor is positioned to the left of the command you wish to be performed, press CONT and the program will begin to process that command.

### Availability Table

You will have to wait while the program searches the directory to find empty file space. How long this process takes will depend on the number of files already in the directory. When the program is completed a new directory will be available. This one will include entries for empty files indicating where and how large they are. The first fifteen files of the modified directory will be displayed. The up and down arrows will allow you to "page" through the complete directory. Each page will hold fifteen files of the modified directory.

Should you wish to have a hard copy of the directory, press EXECUTE and the entire directory will be dumped to your hard copy printer.

When you are done with this section of the Availability portion of the program, press CONT. A new list of files will now be displayed. This list will include ONLY the empty files. It will be ordered by the size of the empty file. This list will represent all available space on your mass storage medium. A total of the number of records available will also be included. The up and down arrows can be used to page through this list.

This list can also be dumped to your hard copy printer by pressing EXECUTE. Pressing CONT will return you to the list of commands.

### Purge Files

The first fifteen entries in your directory will be displayed on the CRT. A cursor will appear to the left of the first entry. You can use the up and down arrows to position the cursor to the left of any file name in the directory. If you move the cursor down past the fifteenth entry then a new page of fifteen entries (assuming there are further entries) will be displayed with the cursor next to the first. If you move the cursor up past the first entry, and you are not on the first page, the previous page will be displayed with the cursor next to the last entry on the page.

When the cursor is to the left of an entry which you wish to purge, press EXECUTE. A special marker will be displayed in the cursor column next to the file name. This marker indicates that the file will be purged when CONT is pressed. If you make a mistake and wish to remove the marker, position the cursor over the marker and press EXECUTE a second time. This will erase the marker. When you are satisfied with the files marked for purging, press CONT. At this time, the files which are marked, and only those files, will be purged.

If a file is protected or if it is of the wrong file type (if it has a question mark next to its file type) that file will NOT be purged. A message to indicate this will be displayed on the CRT. When all indicated files have been purged you will be requested to press CONT. This will return you to the command list.

### Copy Files To Another Medium

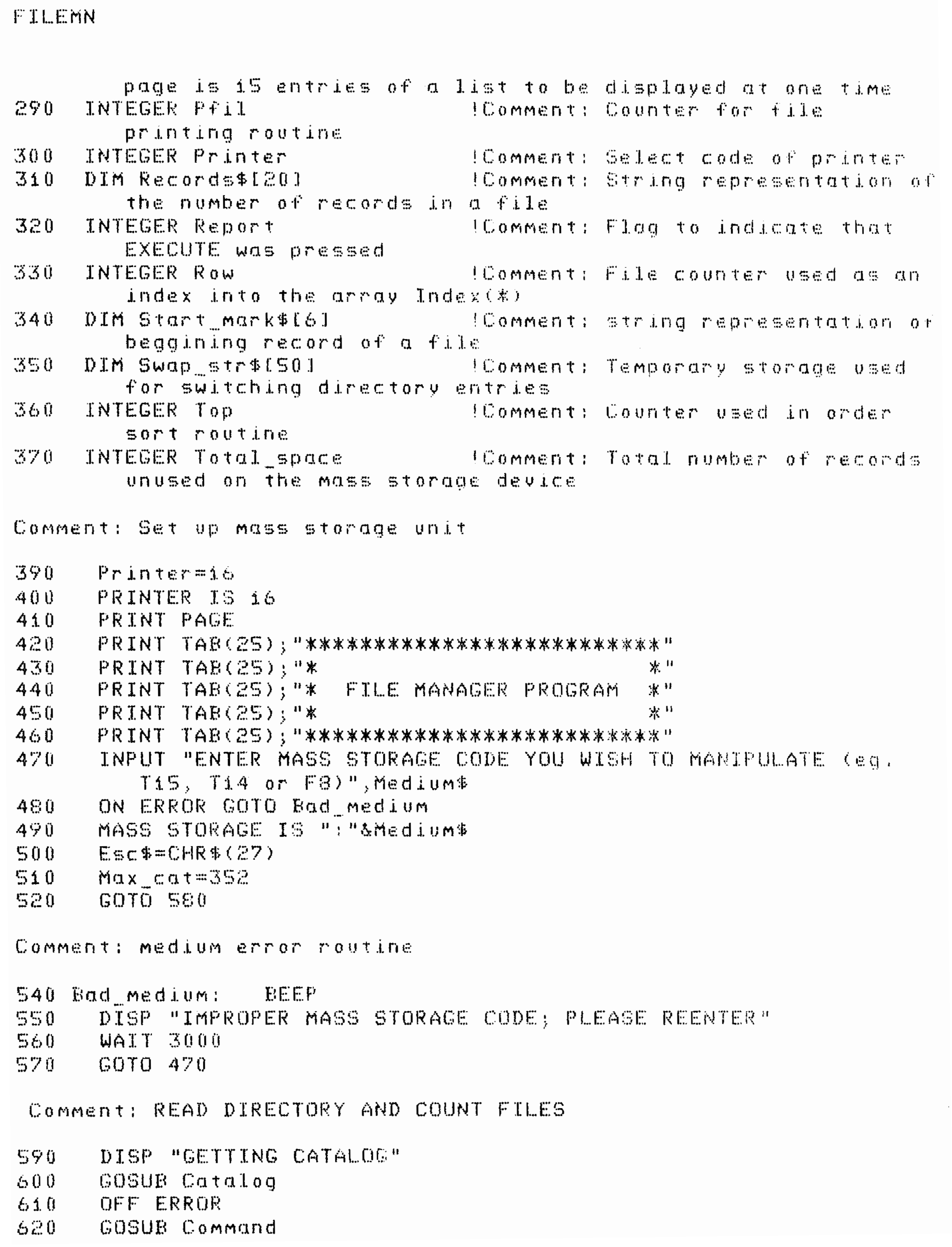
This portion of the program will ask you to enter the device code of the mass storage medium which is to receive the files (e.g., T14 or F8). If there is some problem in accessing this medium you will be notified.

The directory will be displayed as in the PURGE portion of this program. You can move the cursor and mark the entries as was indicated in the directions to PURGE, above. When the files are marked to your satisfaction, press CONT. The files you have marked will be copied to the second mass storage medium. Should a file be of the wrong type (with a question mark following the file type), should there not be room on the receiving mass storage medium or should the name of the file being copied already be present in the directory of the receiving mass storage medium you will be notified. The program will then continue processing the next file.

When the copying is complete you will be asked to press CONT to continue. This will return you to the list of commands.

# FILEMN

```
10   OPTION BASE 1

Comment: Declarations:

30   INTEGER Begin_mark           !Comment: Begining record of an
        empty file
40   INTEGER Bubble               !Comment: counter used in order
        sorting routine
50   DIM Buffer$[160]             !Comment: Buffer to hold keyboard
        characters
60   DIM Cat$(353)[42]            !Comment: Array to hold directory
70   DIM Cat2$(328)[42]           !Comment: Array to hold empty file
        info
80   INTEGER Catpointer(706)      !Comment: Array to hold pointers
        into Cat$(*) and Cat2$(*)
90   INTEGER Changes              !Comment: Flag to indicate a swith
        was done in the order sorting routine
100  DIM Chr$[1]                  !Comment: Current character from
        keyboard input
110  INTEGER Command              !Comment: Number of current command
120  DIM Command_names$(10)[40]    !Comment: Command names
130  DIM Destination$[4]          !Comment: Recipient mass storage
        device in copy routine
140  INTEGER Done                 !Comment: Flag to indicate CONT
        was pressed
150  DIM Dummy$(3)[42]            !Comment: Dummy directory
        requested to check for availability of Destination$ mass
        storage unit
160  INTEGER Empty                !Comment: number of empty files on
        device
170  INTEGER End_mark             !Comment: Last physical record in
        empty file
180  DIM Esc$[1]                  !Comment: Escape character - CHR$(
        27)
190  INTEGER I                    !Comment: Loop counter
200  INTEGER Index(352)           !Comment: Holds indications of
        whether or not a file has been marked for COPY or PURGE
210  INTEGER Max_cat              !Comment: maximum number of files
220  INTEGER Maxrow               !Comment: maximum row to which
        cursor can be moved
230  DIM Medium$[7]               !Comment: Mass storage device to
        be manipulated
240  INTEGER Minrow               !Comment: minimum row to which
        cursor can be moved
250  INTEGER Num_of_files         !Comment: Number of files on mass
        storage medium
260  INTEGER Number               !Comment: Number of empty files on
        mass storage medium
270  INTEGER One                  !Comment: Constant one
280  INTEGER Page                 !Comment: Number of page where a
```



```
FILEMN


        page is 15 entries of a list to be displayed at one time
290   INTEGER Pfil                   !Comment: Counter for file
        printing routine
300   INTEGER Printer                !Comment: Select code of printer
310   DIM Records$[20]               !Comment: String representation of
        the number of records in a file
320   INTEGER Report                 !Comment: Flag to indicate that
        EXECUTE was pressed
330   INTEGER Row                    !Comment: File counter used as an
        index into the array Index(*)
340   DIM Start_mark$[6]             !Comment: string representation or
        beggining record of a file
350   DIM Swap_str$[50]              !Comment: Temporary storage used
        for switching directory entries
360   INTEGER Top                    !Comment: Counter used in order
        sort routine
370   INTEGER Total_space            !Comment: Total number of records
        unused on the mass storage device

Comment: Set up mass storage unit

390    Printer=16
400    PRINTER IS 16
410    PRINT PAGE
420    PRINT TAB(25);"**************************"
430    PRINT TAB(25);"*                        *"
440    PRINT TAB(25);"*  FILE MANAGER PROGRAM  *"
450    PRINT TAB(25);"*                        *"
460    PRINT TAB(25);"**************************"
470    INPUT "ENTER MASS STORAGE CODE YOU WISH TO MANIPULATE (eg.
         T15, T14 or F8)",Medium$
480    ON ERROR GOTO Bad_medium
490    MASS STORAGE IS ":"&Medium$
500    Esc$=CHR$(27)
510    Max_cat=352
520    GOTO 580

Comment: medium error routine

540 Bad_medium:   BEEP
550    DISP "IMPROPER MASS STORAGE CODE; PLEASE REENTER"
560    WAIT 3000
570    GOTO 470

 Comment: READ DIRECTORY AND COUNT FILES

590    DISP "GETTING CATALOG"
600    GOSUB Catalog
610    OFF ERROR
620    GOSUB Command
```

```
630   ON Command GOTO Comm1,Comm2,Comm3,Comm4


650 Comm1:     GOSUB Availability
660  GOTO 620


680 Comm2:   !
690   PRINT Esc$&"m",PAGE,LIN(4),TAB(10);CHR$(132);"COPYING FILES
         FROM ";Medium$;CHR$(128)
700   INPUT "ENTER YOUR DESTINATION MEDIUM (eg. T14,T15 or F8)",
         Destination$
710   One=1
720   ON ERROR GOTO Dest_err
730   CAT TO Dummy$(*),0,One;":"&Destination$
740   GOSUB Copy
750   GOTO 620
760 Dest_err:BEEP
770   DISP "INVALID DESTINATION DEVICE - Press CONT to continue"
780   PAUSE
790   GOTO 620
800 Comm3:  !
810   GOSUB Purge
820   GOTO 620
830 Comm4: !
840   PRINT PAGE;LIN(4);"                        PROGRAM STOPPED"
850   DISP ""
860   STOP

Comment: AVAILABILITY TABLE

880 Availability:   !
890   DISP "ORDERING BY ADDRESS"
900   GOSUB Catalog
910  IF Medium$[1,1]="F" THEN GOSUB Order
920   DISP "FINDING AVAILABLE SPACE"
930   GOSUB Avail
940   GOSUB Print_cat
950   DISP "Up, Down arrows to position; EXECUTE to obtain hard-
         copy; CONT to continue"
960   Minrow=1
970   Maxrow=Number
980   GOSUB Keys
990   GOSUB Avail_tabl
1000  Command=4
1010  Minrow=1
1020  Maxrow=Empty+1
1030  GOSUB Keys
1040  RETURN
```

FILEMN

```
PURGE
1060 Purge:  PRINT Esc$&"m";PAGE
1070  FOR I=1 TO Num_of_files
1080  IF Medium$[1,1]="F" THEN Cat$(I)[39;1]="/"
1090  NEXT I
1100  DISP "PURGING: Position Cursor; EXECUTE to mark (unmark)
         file; CONT purges files"
1110  PRINT "   NAME  PRO TYPE REC/FILE  BYTES/REC
         ADDRESS",LIN(1);"    ";Medium$;LIN(2)
1120  PRINT USING Image;Esc$&"1"
1130    FOR I=1 TO 15
1140 IF (I<=Num_of_files) AND (Medium$[1,1]="F") THEN Cat$(I)[39;1]="/
      "
1150 IF I<=Num_of_files THEN PRINT "  "&Cat$(I)
1160    NEXT I
1170    Minrow=1
1180    Maxrow=Num_of_files
1190    GOSUB Keys
```

Comment: Purge files

```
1210    PRINT Esc$&"m";PAGE,"NOW PURGING FILES",LIN(1)
1220   ON ERROR GOTO Purge_err
1230      FOR I=1 TO Num_of_files
1240      IF Index(I)=0 THEN Next_purge
1250      IF Cat$(I)[15;1]<>"?" THEN 1290
1260      BEEP
1270      PRINT Cat$(I)[1,6];" NOT purged - incorrect file type"
1280      GOTO Next_purge
1290      IF Cat$(I)[8;1]<>"*" THEN 1330
1300      BEEP
1310      PRINT Cat$(I)[1,6];" NOT purged - protected file"
1320      GOTO Next_purge
1330      PURGE Cat$(I)[1,6]&":"&Medium$
1340      PRINT Cat$(I)[1,6];" has been PURGED"
1350 Next_purge:     NEXT I
1360      DISP "RE-CATALOGING DIRECTORY"
1370      GOSUB Catalog
1380      PRINT LIN(2);"PURGING COMPLETE"
1390      DISP " Press CONT to continue"
1400      PAUSE
1410      DISP ""
1420      OFF ERROR
1430      RETURN
```

Comment: Capture errors in purge routine

```
1450 Purge_err:!
1460    PRINT "ERROR: error number";ERRN;" encountered in ";CHR$(34)
           ;Cat$(I)[1,6];CHR$(34);"; CONTINUING WITH NEXT FILE"
```

```
1470     GOTO Next_purge

Comment: Copy subroutine

1490 Copy:    PRINT Esc$&"m";PAGE
1500   DISP "COPY:  Position Cursor; EXECUTE to mark (unmark) file;
          CONT Copies files"
1510   PRINT "    NAME  PRO TYPE REC/FILE  BYTES/REC
          ADDRESS";LIN(1);"    ";Medium$;LIN(2)
1520   PRINT USING Image;Esc$&"1"
1530   FOR I=1 TO Num_of_files
1540   IF Medium$[1,1]="F" THEN Cat$(I)[39;1]="/"
1550   NEXT I
1560     FOR I=1 TO 15
1570 IF (I<=Num_of_files) AND (Medium$[1,1]="F") THEN Cat$(I)[39;1]="/
     "
1580 IF I<=Num_of_files THEN PRINT "  "&Cat$(I)
1590     NEXT I
1600     Minrow=1
1610     Maxrow=Num_of_files
1620     GOSUB Keys
1625     OFF KBD
 Comment: Copy files

1640     PRINT Esc$&"m";PAGE,"NOW COPYING FILES",LIN(1)
1650      ON ERROR GOTO Copy_error
1660       FOR I=1 TO Num_of_files
1670       IF Index(I)=0 THEN Next_copy
1680       IF Cat$(I)[15;1]<>"?" THEN 1720
1690       BEEP
1700       PRINT Cat$(I)[1,6];" NOT copied - incorrect file type"
1710       GOTO Next_copy
1720       COPY Cat$(I)[1,6]&":"&Medium$ TO Cat$(I)[1,6]&":"&
              Destination$
1730       PRINT Cat$(I)[1,6];" has been COPIED"
1740 Next_copy: NEXT 1
1750       PRINT LIN(2);"COPYING COMPLETE"
1760       DISP " Press CONT to continue"
1770       PAUSE
1780       DISP ""
1790       RETURN
1800 Copy_error:  IF ERRN<>55 THEN 1830
1810      PRINT "DIRECTORY OVERFLOW-NO MORE COPYING POSSIBLE"
1820      GOTO 1760
1830      IF ERRN<>83 THEN 1860
1840      PRINT "WRITE PROTECTED-NO COPYING POSSIBLE"
1850      GOTO 1760
1860    IF ERRN<>54 THEN 1890
1870      PRINT "DUPLICATE FILE NAME ENCOUNTERED-";CHR$(34);Cat$(I)[
             1,6];CHR$(34);"-CONTINUING WITH NEXT FILE"
```

FILEMN

```
1880     GOTO Next_copy
1890   IF ERRN<>64 THEN 1920
1900     PRINT CHR$(34);Cat$(I)[1,6];CHR$(34);" TOO LARGE TO FIT ON
            DESTINATION-CONTINUING WITH NEXT FILE"
1910     GOTO Next_copy
1920   PRINT "ERROR - ERROR NUMBER";ERRN;"ENCOUNTERED IN ";CHR$(34);
          Cat$(I)[1,6];CHR$(I);CHR$(34);" - CONTINUING WITH NEXT FILE"
1930   GOTO Next_copy
1940   RETURN
1950 Save_mark:       !
1960  Report=0
1970  IF Index(Row)=0 THEN Save
1980   Index(Row)=0
1990  PRINT USING Image;Esc$&"&a-2C "&Esc$&"&a-2C"
2000   RETURN
2010 Save:   Index(Row)=1
2020  PRINT USING Image;Esc$&"&a-1C*"&Esc$&"&a-2C"
2030  RETURN
```

Comment:  Get directory

```
2050 Catalog: !
2060  CAT TO Cat$(*)
2070 FOR I=1 TO Max_cat
2080 IF Cat$(I)="" THEN Counted
2090 IF Medium$[1,1]="F" THEN Cat$(I)[39;1]="."
2100 NEXT I
2110 Counted: Num_of_files=I-1
2120  RETURN
```

 Comment: FIND AND MARK AVAILABLE SPACE-DISC

```
2140 Avail:IF Medium$[1,1]="T" THEN Tape_avail
2150 Empty=1
2160 Number=1
2170 Cat$(Num_of_files+1)=RPT$(" ",20)&"0"&RPT$(" ",6)&"256"&RPT$(" ",
        6)&"67.00"
2180 End_mark=60
2190    FOR I=1 TO Num_of_files+1
2200    Begin_mark=30*VAL(Cat$(I)[37;2])+VAL(Cat$(I)[40;2])
2210    IF Begin_mark-End_mark=0 THEN 2290
2220    Records$="     "&VAL$(Begin_mark-End_mark)
2230    Start_mark$=VAL$(End_mark DIV 300)&VAL$(End_mark MOD 300 DIV
           30)&"/"&VAL$(End_mark MOD 30 DIV 10)&VAL$(End_mark MOD 10)
2240    IF End_mark DIV 300=0 THEN Start_mark$[1,1]=" "
2250    Cat2$(Empty)=RPT$(CHR$(127),5)&"    empty  "&Records$[LEN(
           Records$)-4;5]&"      256      "&Start_mark$
2260    Catpointer(Number)=2*1000+Empty
2270    Number=Number+1
2280    Empty=Empty+1
```

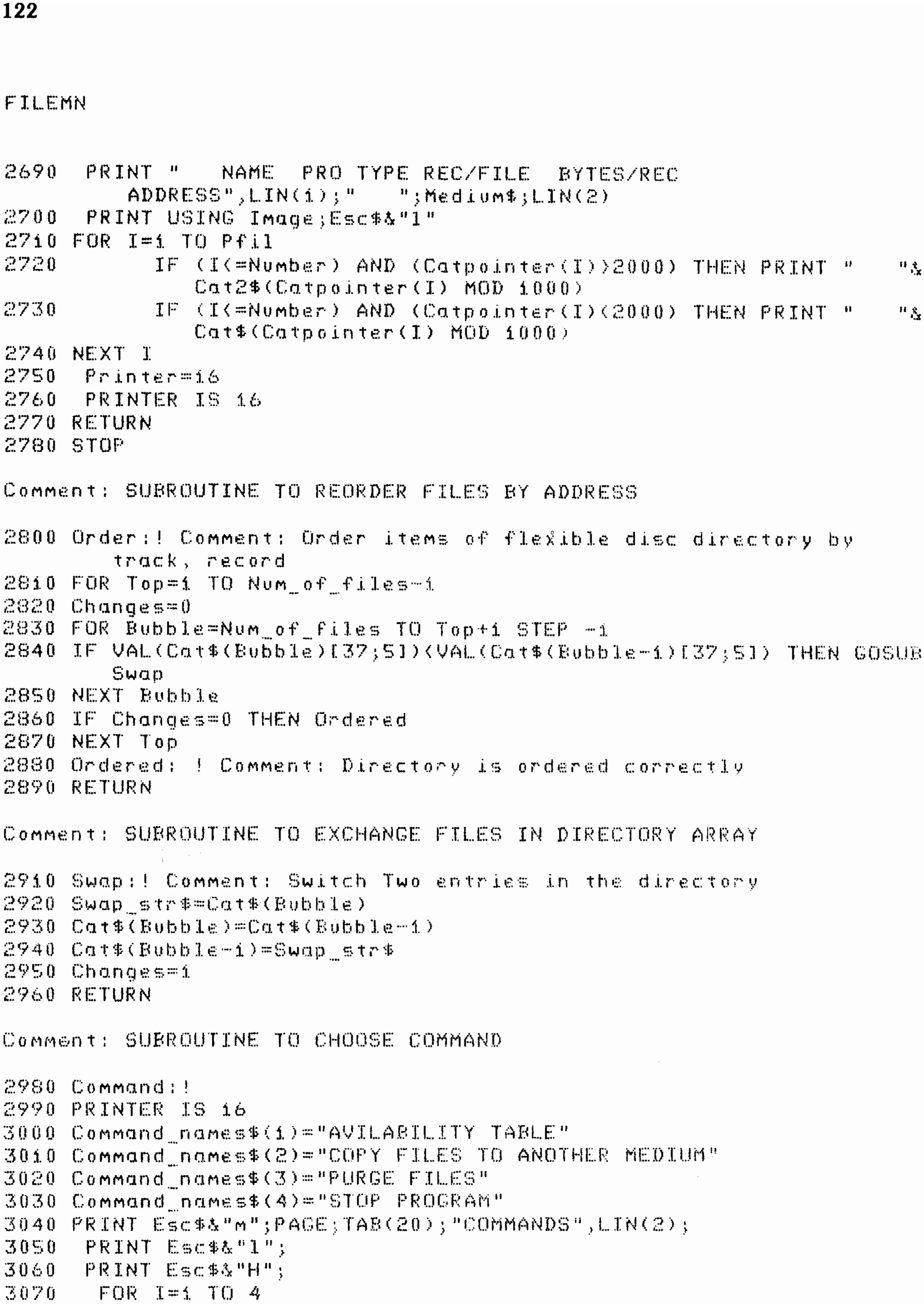
```
2290    Catpointer(Number)=1*1000+I
2300    End_mark=Begin_mark+VAL(Cat$(I)[17,21])*VAL(Cat$(I)[26;5])
           DIV 256+(VAL(Cat$(I)[17,21])*VAL(Cat$(I)[26;5]) MOD 256>0)
2310    Cat$(I)[39;1]="/"
2320    Number=Number+1
2330    NEXT I
2340 Empty=Empty-1
2350 Number=Number-2
2360 RETURN
```

Comment: FIND AND MARK AVAILABLE SPACE-TAPE

```
2380 Tape_avail: !
2390 Cat$(Num_of_files+1)=RPT$(" ",20)&"0"&RPT$(" ",6)&"256"&RPT$(" ",
        8)&"852"
2400 Empty=1
2410 Number=1
2420 End_mark=5
2430    FOR I=1 TO Num_of_files+1
2440    Begin_mark=VAL(Cat$(I)[39;3])
2450    IF Begin_mark-End_mark=0 THEN 2520
2460    Records$="     "&VAL$(Begin_mark-End_mark)
2470    End_mark$="     "&VAL$(End_mark)
2480    Cat2$(Empty)=RPT$(CHR$(127),5)&"    empty   "&Records$[LEN(
           Records$)-4;5]&"      256      "&End_mark$[LEN(End_mark$)-
           4;5]
2490    Catpointer(Number)=2*1000+Empty
2500    Number=Number+1
2510    Empty=Empty+1
2520    Catpointer(Number)=1*1000+I
2530 !  End_mark=Begin_mark+VAL(Cat$(I)[17,21])*VAL(Cat$(I)[26;5])
        DIV 256+(VAL(Cat$(I)[26;5]) MOD 256>0)
2540    End_mark=Begin_mark+VAL(Cat$(I)[17,21])*VAL(Cat$(I)[26;5])
           DIV 256+(VAL(Cat$(I)[17,21])*VAL(Cat$(I)[26;5]) MOD 256>0)
2550    Number=Number+1
2560    NEXT I
2570  Empty=Empty-1
2580  Number=Number-2
2590  RETURN
```

Comment: PRINT MODIFIED DIRECTORY

```
2610 Print_cat:!
2620  IF Printer=16 THEN 2660
2630    Pfil=Number
2640    PRINTER IS 0
2650    GOTO 2690
2660 PRINTER IS 16
2670  Pfil=15
2680  PRINT Esc$&"m",PAGE
```



FILEMN

```
2690  PRINT "    NAME  PRO TYPE REC/FILE  BYTES/REC
         ADDRESS",LIN(1);"    ";Medium$;LIN(2)
2700  PRINT USING Image;Esc$&"1"
2710 FOR I=1 TO Pfil
2720       IF (I<=Number) AND (Catpointer(I)>2000) THEN PRINT "    "&
              Cat2$(Catpointer(I) MOD 1000)
2730       IF (I<=Number) AND (Catpointer(I)<2000) THEN PRINT "    "&
              Cat$(Catpointer(I) MOD 1000)
2740 NEXT I
2750  Printer=16
2760  PRINTER IS 16
2770 RETURN
2780 STOP
```

Comment: SUBROUTINE TO REORDER FILES BY ADDRESS

```
2800 Order:! Comment: Order items of flexible disc directory by
        track, record
2810 FOR Top=1 TO Num_of_files-1
2820 Changes=0
2830 FOR Bubble=Num_of_files TO Top+1 STEP -1
2840 IF VAL(Cat$(Bubble)[37;5])<VAL(Cat$(Bubble-1)[37;5]) THEN GOSUB
        Swap
2850 NEXT Bubble
2860 IF Changes=0 THEN Ordered
2870 NEXT Top
2880 Ordered: ! Comment: Directory is ordered correctly
2890 RETURN
```

Comment: SUBROUTINE TO EXCHANGE FILES IN DIRECTORY ARRAY

```
2910 Swap:! Comment: Switch Two entries in the directory
2920 Swap_str$=Cat$(Bubble)
2930 Cat$(Bubble)=Cat$(Bubble-1)
2940 Cat$(Bubble-1)=Swap_str$
2950 Changes=1
2960 RETURN
```

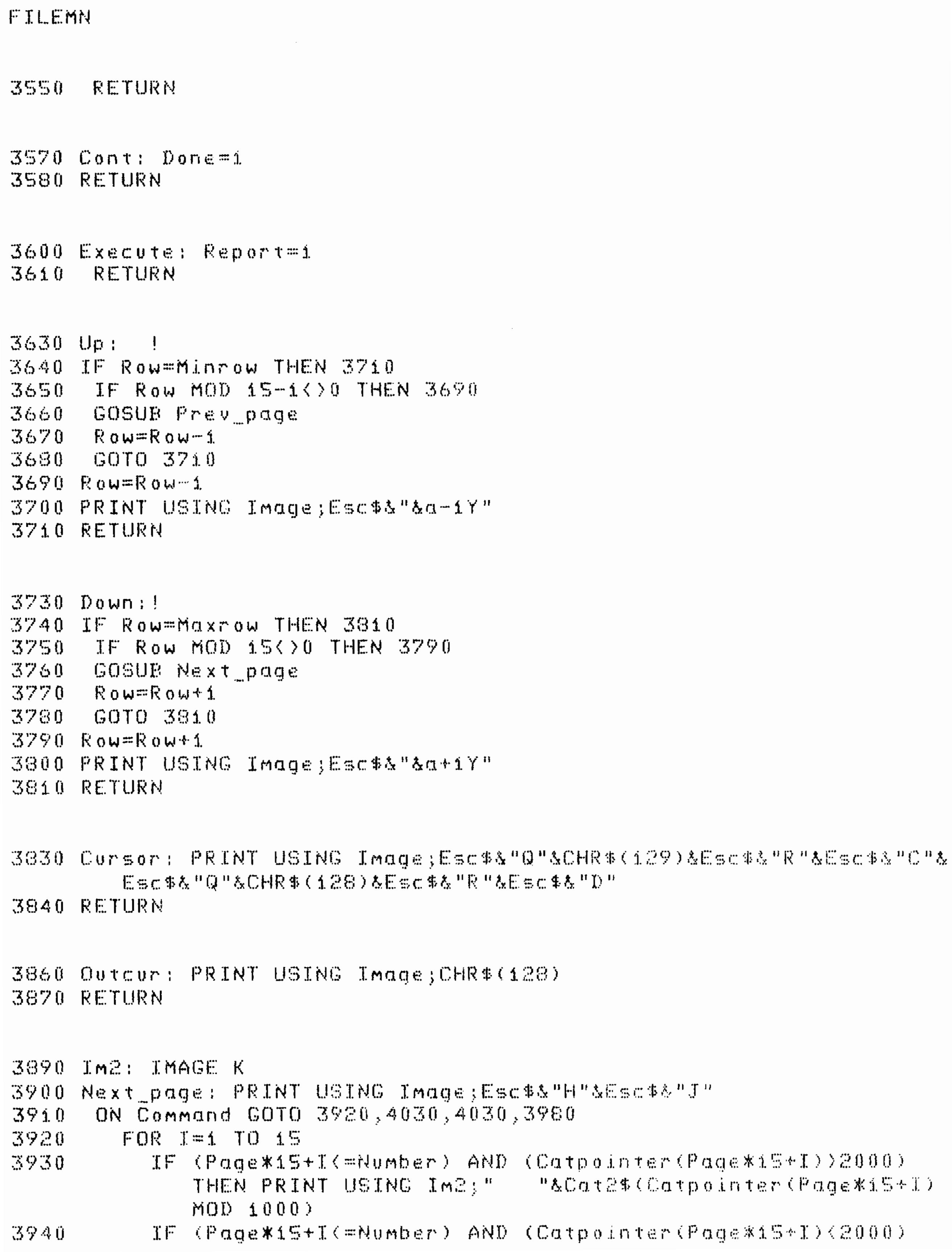
Comment: SUBROUTINE TO CHOOSE COMMAND

```
2980 Command:!
2990 PRINTER IS 16
3000 Command_names$(1)="AVILABILITY TABLE"
3010 Command_names$(2)="COPY FILES TO ANOTHER MEDIUM"
3020 Command_names$(3)="PURGE FILES"
3030 Command_names$(4)="STOP PROGRAM"
3040 PRINT Esc$&"m";PAGE;TAB(20);"COMMANDS",LIN(2);
3050  PRINT Esc$&"1";
3060  PRINT Esc$&"H";
3070   FOR I=1 TO 4
```

```
3080   PRINT "  ";I;"- ";Command_names$(I)
3090   NEXT I
3100 DISP "Use UP and DOWN arrows to move cursor to desired
        command and press CONT"
3110 Minrow=1
3120 Maxrow=4
3130 GOSUB Keys
3140 Command=Row
3150  PRINT Esc$&"m"
3160 RETURN
```

SUBROUTINE TO DECODE KEYS PRESSED

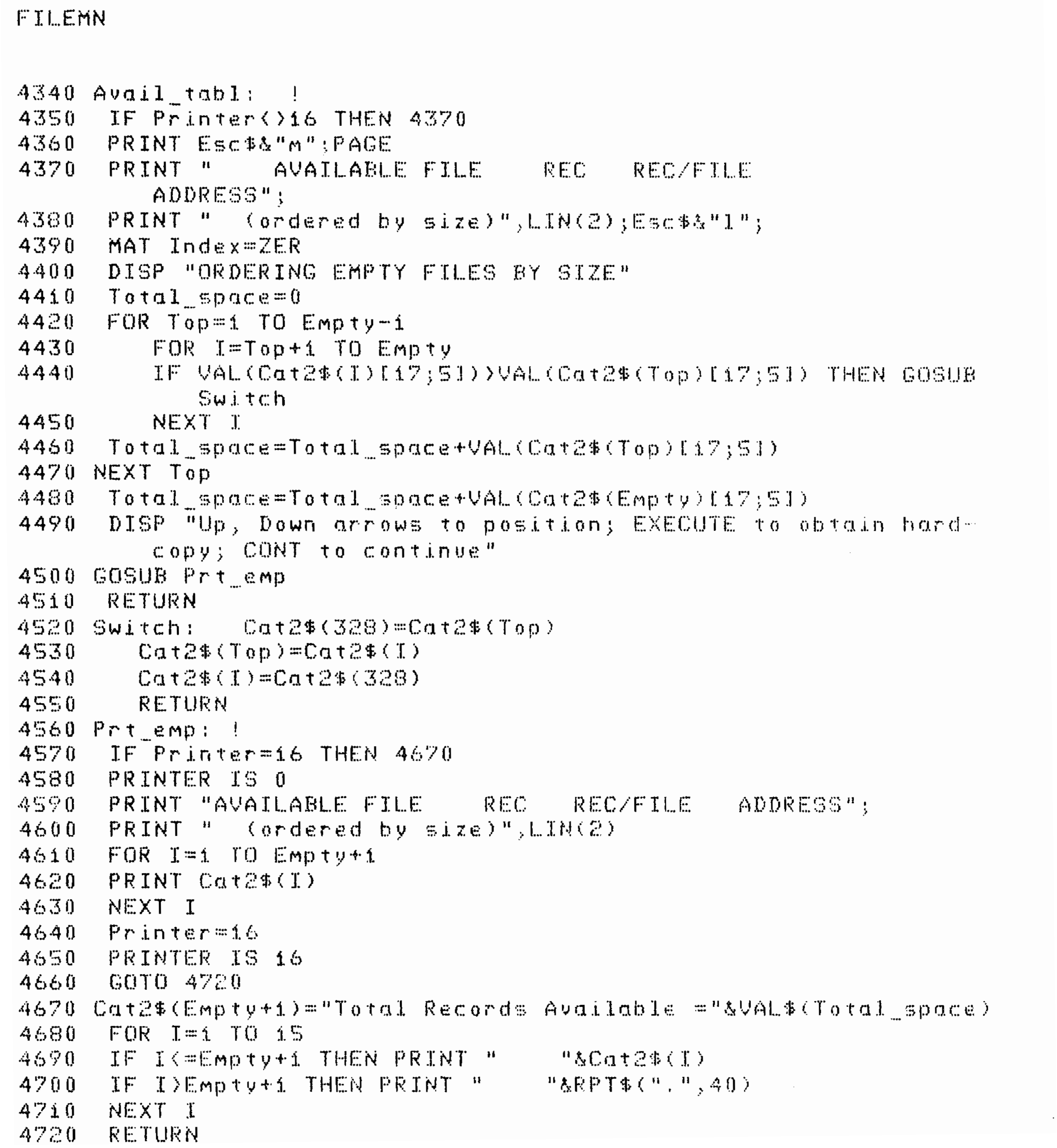
```
3180 Keys:!
3190 Done=0
3200  Page=1
3210 Report=0
3220  MAT Index=ZER
3230  PRINTER IS 16
3240  PRINT USING Image;Esc$&"H"
3250  Row=Minrow
3260  GOSUB Cursor
3270  Chr$="Q"
3280    ON KBD GOSUB Decode ,ALL
3290   IF Done THEN RETURN
3300   IF Report AND ((Command=2) OR (Command=3)) THEN GOSUB Save_mark
3310   IF NOT Report OR (Command<>4) THEN 3360
3320     Printer=0
3330     GOSUB Prt_emp
3340     Report=0
3350     GOTO 3290
3360   IF NOT Report OR (Command<>1) THEN 3400
3370     Printer=0
3380     GOSUB Print_cat
3390     Report=0
3400   GOTO 3290
3410 Image: IMAGE #,K
3420 Decode:          Buffer$=KBD$
3430  IF LEN(Buffer$)<1 THEN RETURN
3440  Chr$=Buffer$[1,1]
3450  Buffer$=Buffer$[2]
3460  IF NUM(Chr$)=255 THEN GOSUB Specials
3470  RETURN
3480 Specials:   !
3490  Chr$=Buffer$[1,1]
3500  Buffer$=Buffer$[2]
3510  IF (NUM(Chr$)>25) OR (NUM(Chr$)<19) THEN 3430
3520  GOSUB Outcur
3530  ON NUM(Chr$)-18 GOSUB Cont,3540,Execute,3540,3540,Up,Down
3540  GOSUB Cursor
```



FILEMN

```
3550  RETURN

3570 Cont: Done=1
3580 RETURN

3600 Execute: Report=1
3610  RETURN

3630 Up:  !
3640 IF Row=Minrow THEN 3710
3650  IF Row MOD 15-1<>0 THEN 3690
3660  GOSUB Prev_page
3670  Row=Row-1
3680  GOTO 3710
3690 Row=Row-1
3700 PRINT USING Image;Esc$&"&a-1Y"
3710 RETURN

3730 Down:!
3740 IF Row=Maxrow THEN 3810
3750  IF Row MOD 15<>0 THEN 3790
3760  GOSUB Next_page
3770  Row=Row+1
3780  GOTO 3810
3790 Row=Row+1
3800 PRINT USING Image;Esc$&"&a+1Y"
3810 RETURN

3830 Cursor: PRINT USING Image;Esc$&"Q"&CHR$(129)&Esc$&"R"&Esc$&"C"&
        Esc$&"Q"&CHR$(128)&Esc$&"R"&Esc$&"D"
3840 RETURN

3860 Outcur: PRINT USING Image;CHR$(128)
3870 RETURN

3890 Im2: IMAGE K
3900 Next_page: PRINT USING Image;Esc$&"H"&Esc$&"J"
3910  ON Command GOTO 3920,4030,4030,3980
3920    FOR I=1 TO 15
3930      IF (Page*15+I<=Number) AND (Catpointer(Page*15+I)>2000)
         THEN PRINT USING Im2;"    "&Cat2$(Catpointer(Page*15+I)
         MOD 1000)
3940      IF (Page*15+I<=Number) AND (Catpointer(Page*15+I)<2000)
```

```
             THEN PRINT USING Im2;"    "&Cat$(Catpointer(Page*15+I)
             MOD 1000)
3950        IF Page*15+I>Number THEN PRINT USING Im2;"     "&RPT$(".",40)
3960      NEXT I
3970      GOTO 4090
3980      FOR I=1 TO 15
3990        IF Page*15+I<=Empty+1 THEN PRINT USING Im2;"     "&Cat2$(
             Page*15+I)
4000        IF Page*15+I>Empty+1 THEN PRINT USING Im2;"      "&RPT$(".",
             40)
4010      NEXT I
4020      GOTO 4090
4030      FOR I=1 TO 15
4040       IF Index(Page*15+I)=0 THEN PRINT USING Image;" "
4050       IF Index(Page*15+I)=1 THEN PRINT USING Image;"*"
4060       IF Page*15+I<=Num_of_files THEN PRINT USING Im2;" "&Cat$(
            Page*15+I)
4070       IF Page*15+I>Num_of_files THEN PRINT USING Im2;" "&RPT$(".",
            41)
4080       NEXT I
4090       PRINT USING Image;Esc$&"H"
4100       Page=Page+1
4110       RETURN


4130 Prev_page:     !
4140      PRINT USING Image;Esc$&"H"&Esc$&"J"
4150      Page=Page-1
4160      ON Command GOTO 4170,4260,4260,4220
4170      FOR I=1 TO 15
4180        IF ((Page-1)*15+I<Number) AND (Catpointer((Page-1)*15+I)>
             2000) THEN PRINT USING Im2;"    "&Cat2$(Catpointer((Page-
             1)*15+I) MOD 1000)
4190        IF ((Page-1)*15+I<Number) AND (Catpointer((Page-1)*15+I)<
             2000) THEN PRINT USING Im2;"    "&Cat$(Catpointer((Page-1)
             *15+I) MOD 1000)
4200      NEXT I
4210      GOTO 4310
4220      FOR I=1 TO 15
4230        PRINT USING Im2;"     "&Cat2$((Page-1)*15+I)
4240      NEXT I
4250      GOTO 4310
4260      FOR I=1 TO 15
4270       IF Index((Page-1)*15+I)=1 THEN PRINT USING Image;"*"
4280       IF Index((Page-1)*15+I)=0 THEN PRINT USING Image;" "
4290      PRINT USING Im2;" "&Cat$((Page-1)*15+I)
4300      NEXT I
4310      PRINT USING Image;Esc$&"&a-1Y"
4320      RETURN
```



```
FILEMN


4340 Avail_tabl:  !
4350  IF Printer<>16 THEN 4370
4360  PRINT Esc$&"m";PAGE
4370  PRINT "    AVAILABLE FILE    REC    REC/FILE
         ADDRESS";
4380  PRINT "  (ordered by size)",LIN(2);Esc$&"l";
4390  MAT Index=ZER
4400  DISP "ORDERING EMPTY FILES BY SIZE"
4410  Total_space=0
4420  FOR Top=1 TO Empty-1
4430     FOR I=Top+1 TO Empty
4440     IF VAL(Cat2$(I)[17;5])>VAL(Cat2$(Top)[17;5]) THEN GOSUB
            Switch
4450     NEXT I
4460  Total_space=Total_space+VAL(Cat2$(Top)[17;5])
4470 NEXT Top
4480  Total_space=Total_space+VAL(Cat2$(Empty)[17;5])
4490  DISP "Up, Down arrows to position; EXECUTE to obtain hard-
         copy; CONT to continue"
4500 GOSUB Prt_emp
4510  RETURN
4520 Switch:   Cat2$(328)=Cat2$(Top)
4530    Cat2$(Top)=Cat2$(I)
4540    Cat2$(I)=Cat2$(328)
4550    RETURN
4560 Prt_emp: !
4570  IF Printer=16 THEN 4670
4580  PRINTER IS 0
4590  PRINT "AVAILABLE FILE    REC    REC/FILE    ADDRESS";
4600  PRINT "  (ordered by size)",LIN(2)
4610  FOR I=1 TO Empty+1
4620  PRINT Cat2$(I)
4630  NEXT I
4640  Printer=16
4650  PRINTER IS 16
4660  GOTO 4720
4670 Cat2$(Empty+1)="Total Records Available ="&VAL$(Total_space)
4680  FOR I=1 TO 15
4690  IF I<=Empty+1 THEN PRINT "    "&Cat2$(I)
4700  IF I>Empty+1 THEN PRINT "    "&RPT$(".",40)
4710  NEXT I
4720  RETURN
```

# FILEMN

```
FILEMN

MAIN PROGRAM

COMMON VARIABLES:

VARIABLES:
Begin_mark               30   2200  2210  2220  2300  2440  2450  2460
                              2540
Bubble                   40   2830  2840  2850  2920  2930  2940
Buffer$                  50   3420  3430  3440  3450  3490  3500
Cat$(*)                  60   1080  1140  1150  1250  1270  1290  1310
                              1330  1340  1460  1540  1570  1580  1680
                              1700  1720  1730  1870  1900  1920  2060
                              2080  2090  2170  2200  2300  2310  2390
                              2440  2540  2730  2840  2920  2930  2940
                              3940  4060  4190  4290
Cat2$(*)                 70   2250  2480  2720  3930  3990  4180  4230
                              4440  4460  4480  4520  4530  4540  4620
                              4670  4690
Catpointer(*)            80   2260  2290  2490  2520  2720  2730  3930
                              3940  4180  4190
Changes                  90   2820  2860  2950
Chr$                    100   3270  3440  3460  3490  3510  3530
Command                 110    630  1000  3140  3300  3310  3360  3910
                              4160
Command_names$(*)       120   3000  3010  3020  3030  3080
Destination$            130    700   730  1720
Done                    140   3190  3290  3570
Dummy$(*)               150    730
Empty                   160   1020  2150  2250  2260  2280  2340  2400
                              2480  2490  2510  2570  3990  4000  4420
                              4430  4480  4610  4670  4690  4700
End_mark                170   2180  2210  2220  2230  2240  2300  2420
                              2450  2460  2470  2540
End_mark$              2470   2480
Esc$                    180    500   690  1060  1120  1210  1490  1520
                              1640  1990  2020  2680  2700  3040  3050
                              3060  3150  3240  3700  3800  3830  3900
                              4090  4140  4310  4360  4380
I                       190   1070  1080  1090  1130  1140  1150  1160
                              1230  1240  1250  1270  1290  1310  1330
                              1340  1350  1460  1530  1540  1550  1560
                              1570  1580  1590  1660  1670  1680  1700
                              1720  1730  1740  1870  1900  1920  2070
                              2080  2090  2100  2110  2190  2200  2290
                              2300  2310  2330  2430  2440  2520  2540
                              2560  2710  2720  2730  2740  3070  3080
                              3090  3920  3930  3940  3950  3960  3980
                              3990  4000  4010  4030  4040  4050  4060
                              4070  4080  4170  4180  4190  4200  4220
```

```
FILEMN


                                  4230   4240   4260   4270   4280   4290   4300
                                  4430   4440   4450   4530   4540   4610   4620
                                  4630   4680   4690   4700   4710
Index(*)                   200    1240   1670   1970   1980   2010   3220   4040
                                  4050   4270   4280   4390
Max_cat                    210     510   2070
Maxrow                     220     970   1020   1130   1610   3120   3740
Medium$                    230     470    490    690    910   1080   1110   1140
                                  1330   1510   1540   1570   1720   2090   2140
                                  2690
Minrow                     240     960   1010   1170   1600   3110   3250   3640
Num_of_files               250    1070   1140   1150   1180   1230   1530   1570
                                  1580   1610   1660   2110   2170   2190   2390
                                  2430   2810   2830   4060   4070
Number                     260     970   2160   2260   2270   2290   2320   2350
                                  2410   2490   2500   2520   2550   2580   2630
                                  2720   2730   3930   3940   3950   4180   4190
One                        270     710    730
Page                       280    3200   3930   3940   3950   3990   4000   4040
                                  4050   4060   4070   4100   4150   4180   4190
                                  4230   4270   4280   4290
Pfil                       290    2630   2670   2710
Printer                    300     390   2620   2750   3320   3370   4350   4570
                                  4640
Records$                   310    2220   2250   2460   2480
Report                     320    1960   3210   3300   3310   3340   3360   3390
                                  3600
Row                        330    1970   1980   2010   3140   3250   3640   3650
                                  3670   3690   3740   3750   3770   3790
Start_mark$                340    2230   2240   2250
Swap_str$                  350    2920   2940
Top                        360    2810   2830   2870   4420   4430   4440   4460
                                  4470   4520   4530
Total_space                370    4410   4460   4480   4670

USER DEFINED FUNCTIONS:

SUB PROGRAMS:

JUMP TARGETS:
470                        570
540    Bad_medium:         480
580                        520
620                        660     750    790    820
650    Comm1:              630
680    Comm2:              630
760    Dest_err:           720
800    Comm3:              630
830    Comm4:              630
880    Availability:       650
1060   Purge:              310
```

```
                                                              FILEMN


1290                     1250
1330                     1290
1350  Next_purge:        1240  1280  1320  1470
1450  Purge_err:         1220
1490  Copy:               740
1720                     1680
1740  Next_copy:         1670  1710  1880  1910  1930
1760                     1820  1850
1800  Copy_error:        1650
1830                     1800
1860                     1830
1890                     1860
1920                     1890
1950  Save_mark:         3300
2010  Save:              1970
2050  Catalog:            600   900.  1370
2110  Counted:           2080
2140  Avail:              930
2290                     2210
2380  Tape_avail:        2140
2520                     2450
2610  Print_cat:          940  3380
2660                     2620
2690                     2650
2800  Order:              910
2880  Ordered:           2860
2910  Swap:              2840
2980  Command:            620
3180  Keys:               980  1030  1190  1620  3130
3290                     3350  3400
3360                     3310
3400                     3360
3410  Image:             1120  1520  1990  2020  2700  3240  3700  3800
                               3830  3860  3900  4040  4050  4090  4140
                               4270  4280  4310
3420  Decode:            3280
3430                     3510
3480  Specials:          3460
3540                     3530
3570  Cont:              3530
3600  Execute:           3530
3630  Up:                3530
3690                     3650
3710                     3640  3680
3730  Down:              3530
3790                     3750
3810                     3740  3780
3830  Cursor:            3260  3540
3860  Outcur:            3520
3890  Im2:               3930  3940  3950  3990  4000  4060  4070  4180
                               4190  4230  4290
```

```
FILEMN


3900  Next_page:         3760
3920                     3910
3980                     3910
4030                     3910
4090                     3970  4020
4130  Prev_page:         3660
4170                     4160
4220                     4160
4260                     4160
4310                     4210  4250
4340  Avail_tabl:         990
4370                     4350
4520  Switch:            4440
4560  Prt_emp:           3330  4500
4670                     4570
4720                     4660
```

# 9885 Flexible Disk Backup And Restoration Using Tapes

## Object of Program

These two related programs will allow you to duplicate the contents of a 9885 flexible disk onto three tapes and to later restore the contents of the disk from these tapes. This means you can back up a flexible disk without a second disk drive. The backup is made by reading physical records, one by one, from the disk and copying them to the tapes. The first tape will contain the first 852 physical records from the disk. The second tape will hold the second 852 records and the third tape will hold the final 306 records. The tapes must have been initialized at the time of use.

## Special Considerations

Due to the fact that the physical records are copied directly to the tapes those tapes will not be usable for any other purpose without reinitializing them. They will have no directories. You should be certain that THERE IS NOTHING YOU WISH TO KEEP ON THE TAPES. All information on the tapes prior to their use by the FLEX_BACKUP program will be lost and unrecoverable.

Should you wish to restore the contents of the flexible disk at a later time the RESTORE_FLEX program will reverse the process indicated above. The 852 records from the first tape, the 852 from the second and the 306 from the third will be copied to the corresponding 2010 records of the disk. As with the FLEX_BACKUP program you should make sure that THERE IS NOTHING YOU WISH TO KEEP ON THE 9885 FLEXIBLE DISK before you use the RESTORE_FLEX program. You should also be certain that you know which tape is which since a confusion in the tape order will result in an unusable disk. In this case, the program will have to be rerun.

## User Instructions-Backup

The FLEX_BACKUP program is in PROG file "FLXBAK" and must be LOADED.

a. Type: LOAD "FLXBAK"

b. Press: EXECUTE

A 9845B (NOT a 9845A) is necessary to run this program.

This program utilizes the PHYREC binary to read and write physical records.

### Data to be entered

You will be asked to enter the mass storage unit specifier of the 9885 flexible disk (e.g. F8 or F8,1) and the mass storage unit specifier where the tapes will be inserted (e.g. T14 or T15).

As each tape is required you will be asked to insert the tape in the indicated device and press CONT. The program will then copy the records from the disk to the tape. As each physical record is copied the record number (0 to 2009) will be displayed. When the back up is complete you will be so notified.

## User Instructions-Restore

The RESTORE_FLEX program is in PROG file ''RSTFLX'' and must be LOADED.

a. Type: LOAD ''RSTFLX''

b. Press: EXECUTE

A 9845B (NOT a 9845A) is necessary to run this program.

This program utilizes the PHYREC binary to read and write physical records.

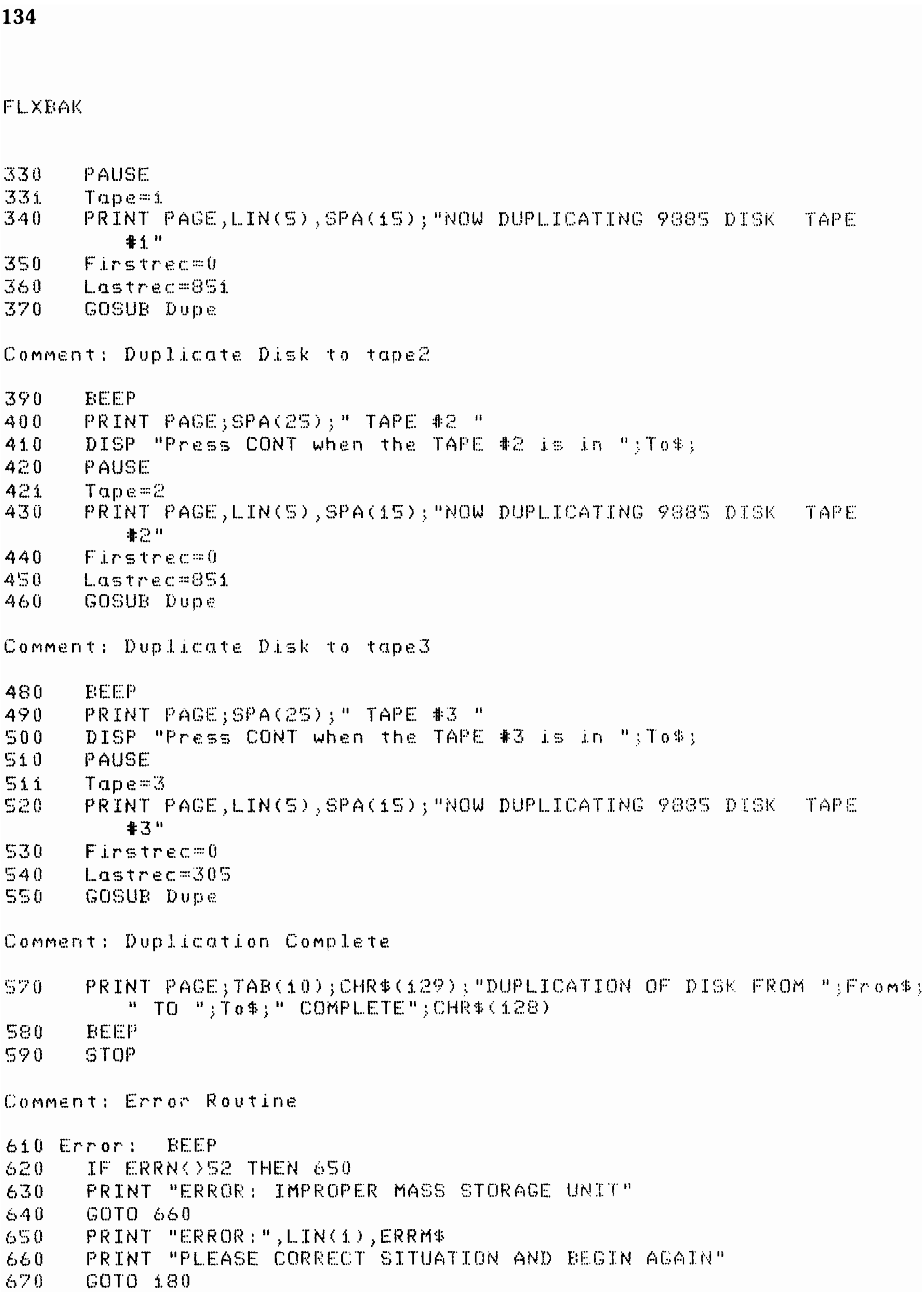
### Data to be entered

You will be asked to enter the mass storage unit specifier of the 9885 flexible disk to be restored (e.g. F8 or F8,1) and the mass storage unit specifier where the tapes will be inserted (e.g. T14 or T15).

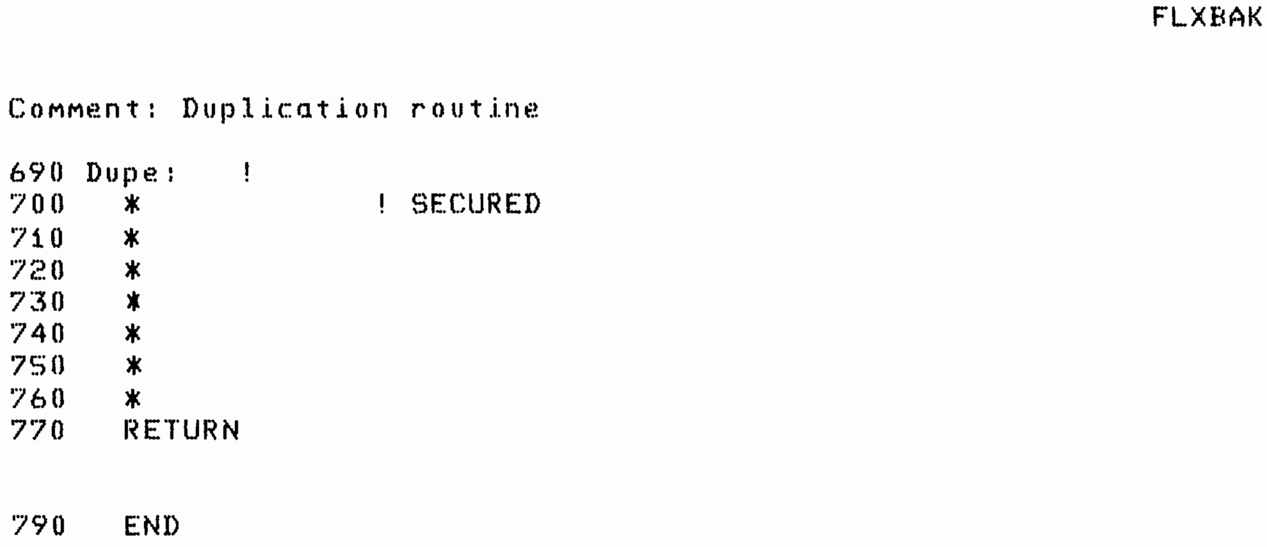
As each tape is required you will be asked to insert the tape in the indicated device and press CONT. The program will then copy the records from the tape to the disk. As each record is copied the record number will be displayed. When the restoration is complete you will be notified.

# FLXBAK

```
Comment: DUPLICATES 9885 FLEXIBLE DISK ONTO 3 TAPES

20     DIM From$[5]                      ! Comment: Origin flexible disk
30     DIM To$[5]                        ! Comment: Destination tape
40     INTEGER Array_rec(127)            ! Comment: Array to Hold
          Physical Record
50     INTEGER Firstrec                  ! Comment: First record to be
          duplicated
60     INTEGER Lastrec                   ! Comment: Last record to be
          duplicated
70     INTEGER Record                    ! Comment: Record counter
80     PRINTER IS 16

Comment: Enter Disk Address of origin

100    ON ERROR GOTO Error
110    PRINT PAGE
120    PRINT TAB(15);"DUPLICATE CONTENTS OF ONE 9885 FLEXIBLE DISK TO
          3 TAPES";LIN(2)
130    PRINT "WARNING:      This program will totally duplicate the
          original "
140    PRINT TAB(10);"disk to the destination tapes. MAKE SURE THERE
          IS NOTHING"
150    PRINT TAB(10);"YOU WISH TO SAVE ON THE DESTINATION TAPES! If
          you wish to"
160    PRINT TAB(10);"append files see the user manual for the file
          manager program.";LIN(3)
170    ON ERROR GOTO Error
180   LINPUT "ENTER the Mass Storage Code for the Original disk (
         eg. F8  or F8,1)",From$
190    IF From$[1,1]="F" THEN 220
200       BEEP
210       GOTO 180
220    MASS STORAGE IS ":"&From$
230    PRINT TAB(10);"FROM: ";From$

Comment: Enter Tape Address of destination

250    INPUT "ENTER the Mass Storage Address for the Destination
          Tapes (eg. T15 or T14)",To$
260    IF To$[1,1]="T" THEN 290
270      BEEP
280      GOTO 250
290    MASS STORAGE IS ":"&To$
300    PRINT TAB(10);"TO: ";To$;LIN(2)

Comment: Wait until ready

320    DISP "Press CONT when the  TAPE #1  is in ";To$;
```



```
FLXBAK


330    PAUSE
331    Tape=1
340    PRINT PAGE,LIN(5),SPA(15);"NOW DUPLICATING 9885 DISK  TAPE
          #1"
350    Firstrec=0
360    Lastrec=851
370    GOSUB Dupe

Comment: Duplicate Disk to tape2

390    BEEP
400    PRINT PAGE;SPA(25);" TAPE #2 "
410    DISP "Press CONT when the TAPE #2 is in ";To$;
420    PAUSE
421    Tape=2
430    PRINT PAGE,LIN(5),SPA(15);"NOW DUPLICATING 9885 DISK  TAPE
          #2"
440    Firstrec=0
450    Lastrec=851
460    GOSUB Dupe

Comment: Duplicate Disk to tape3

480    BEEP
490    PRINT PAGE;SPA(25);" TAPE #3 "
500    DISP "Press CONT when the TAPE #3 is in ";To$;
510    PAUSE
511    Tape=3
520    PRINT PAGE,LIN(5),SPA(15);"NOW DUPLICATING 9885 DISK  TAPE
          #3"
530    Firstrec=0
540    Lastrec=305
550    GOSUB Dupe

Comment: Duplication Complete

570    PRINT PAGE;TAB(10);CHR$(129);"DUPLICATION OF DISK FROM ";From$;
          " TO ";To$;" COMPLETE";CHR$(128)
580    BEEP
590    STOP

Comment: Error Routine

610 Error:  BEEP
620    IF ERRN<>52 THEN 650
630    PRINT "ERROR: IMPROPER MASS STORAGE UNIT"
640    GOTO 660
650    PRINT "ERROR:",LIN(1),ERRM$
660    PRINT "PLEASE CORRECT SITUATION AND BEGIN AGAIN"
670    GOTO 180
```



FLXBAK

```
Comment: Duplication routine

690 Dupe:    !
700   *              ! SECURED
710   *
720   *
730   *
740   *
750   *
760   *
770   RETURN


790   END
```

# FLXBAK

```
FLXBAK

MAIN PROGRAM

COMMON VARIABLES:

VARIABLES:
Array_rec(*)                40   730   750
Firstrec                    50   350   440   530   700
From$                       20   180   190   220   230   570   720
Lastrec                     60   360   450   540   700
Record                      70   700   710   730   750   760
Tape                       331   421   511   730
To$                         30   250   260   290   300   320   410   500
                                 570   740

USER DEFINED FUNCTIONS:

SUB PROGRAMS:

JUMP TARGETS:
180                        210   670
220                        190
250                        280
290                        260
610   Error:               100   170
650                        620
660                        640
690   Dupe:                370   460   550
```

# RSTFLX

Comment: RESTORES 9885 FLEXIBLE DISK FROM 3 TAPES

PREVIOUSLY DUPLICATED FROM FLEXIBLE DISK

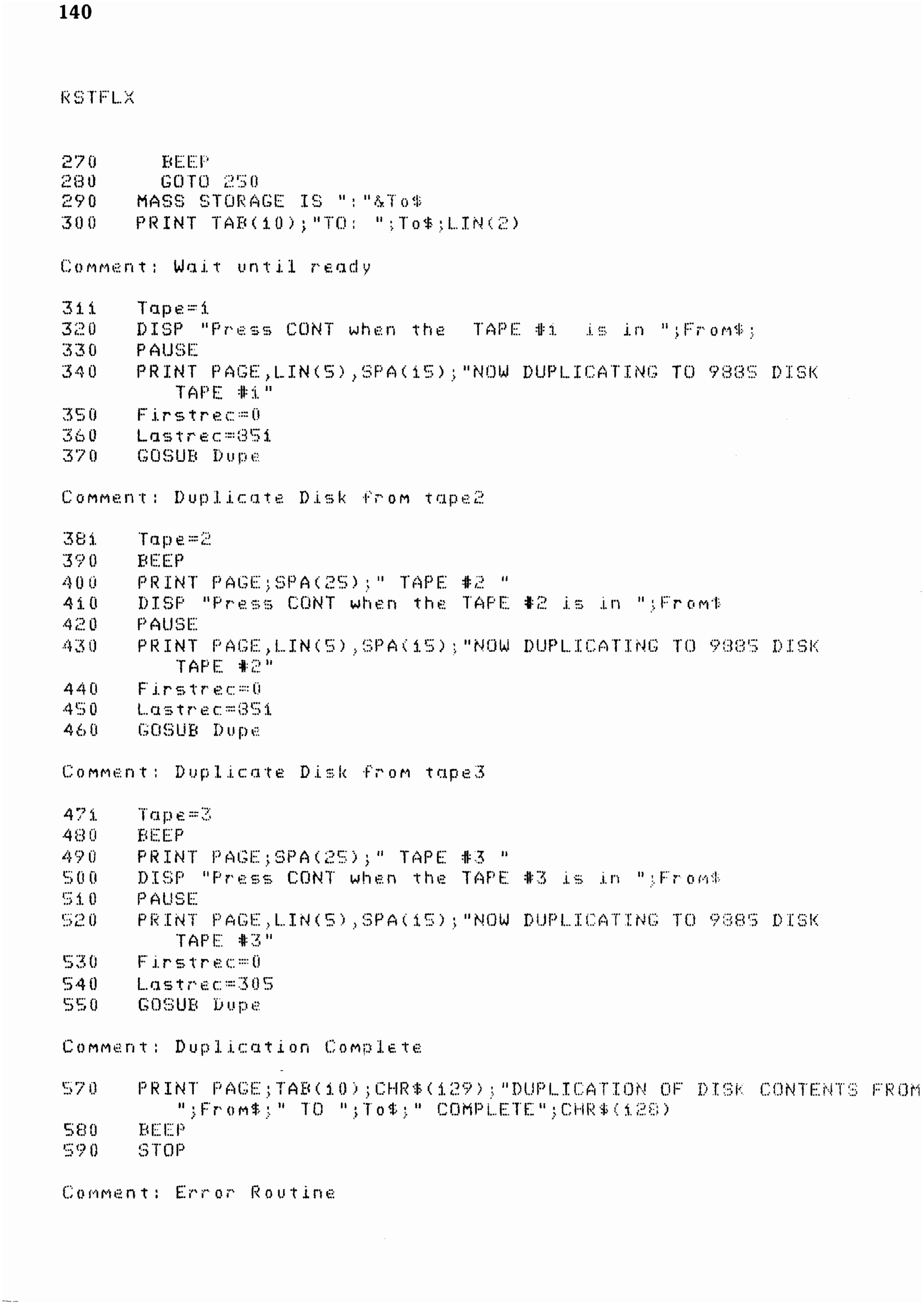
```
20     DIM From$[5]                        ! Comment: Origin address of
          tapes
30     DIM To$[5]                          ! Comment: Destination 9885
          disk
40     INTEGER Array_rec(127)              ! Comment: Array to Hold
          Physical Record
50     INTEGER Firstrec                    ! Comment: First record to be
          duplicated
60     INTEGER Lastrec                     ! Comment: Last record to be
          duplicated
70     INTEGER Record                      ! Comment: Record counter
71     INTEGER Tape                        ! Comment: Number of the
          current tape
80     PRINTER IS 16
```

Comment: Enter Tape Address of origin

```
100    ON ERROR GOTO Error
110    PRINT PAGE
120    PRINT TAB(15);"DUPLICATE CONTENTS OF ONE 3 TAPES TO 9885
          FLEXIBLE DISK";LIN(2)
130    PRINT "WARNING:      This program will totally duplicate the
          original "
135    PRINT TAB(10);"disk contents previously stored on the tree
          tapes back "
140    PRINT TAB(10);"onto a 9885 flexible disk.   MAKE SURE THERE IS
          NOTHING"
150    PRINT TAB(10);"YOU WISH TO SAVE ON THE DESTINATION DISK! If
          you wish to"
160    PRINT TAB(10);"append files see the user manual for the file
          manager program.";LIN(3)
170    ON ERROR GOTO Error
180    INPUT "ENTER the Mass Storage Address for the three tapes (
          eg. T15 or T14)",From$
190    IF From$[1,1]="T" THEN 220
200       BEEP
210       GOTO 180
220    MASS STORAGE IS ":"&From$
230    PRINT TAB(10);"FROM: ";From$
```

Comment: Enter Disk Address of destination

```
250   LINPUT "ENTER the Mass Storage Address for the Destination
         Disk (eg. F8 or F8,1)",To$
260    IF To$[1,1]="F" THEN 290
```



RSTFLX

```
270       BEEP
280       GOTO 250
290     MASS STORAGE IS ":"&To$
300     PRINT TAB(10);"TO: ";To$;LIN(2)

Comment: Wait until ready

311     Tape=1
320     DISP "Press CONT when the  TAPE #1  is in ";From$;
330     PAUSE
340     PRINT PAGE,LIN(5),SPA(15);"NOW DUPLICATING TO 9885 DISK
            TAPE #1"
350     Firstrec=0
360     Lastrec=851
370     GOSUB Dupe

Comment: Duplicate Disk from tape2

381     Tape=2
390     BEEP
400     PRINT PAGE;SPA(25);" TAPE #2 "
410     DISP "Press CONT when the TAPE #2 is in ";From$
420     PAUSE
430     PRINT PAGE,LIN(5),SPA(15);"NOW DUPLICATING TO 9885 DISK
            TAPE #2"
440     Firstrec=0
450     Lastrec=851
460     GOSUB Dupe

Comment: Duplicate Disk from tape3

471     Tape=3
480     BEEP
490     PRINT PAGE;SPA(25);" TAPE #3 "
500     DISP "Press CONT when the TAPE #3 is in ";From$
510     PAUSE
520     PRINT PAGE,LIN(5),SPA(15);"NOW DUPLICATING TO 9885 DISK
            TAPE #3"
530     Firstrec=0
540     Lastrec=305
550     GOSUB Dupe

Comment: Duplication Complete

570     PRINT PAGE;TAB(10);CHR$(129);"DUPLICATION OF DISK CONTENTS FROM
            ";From$;" TO ";To$;" COMPLETE";CHR$(128)
580     BEEP
590     STOP

Comment: Error Routine
```

```
610 Error:  BEEP
611   IF (ERRN<>88) AND (ERRN<>87) AND (ERRN<>84) THEN 620
612   PRINT "ERROR: DATA ERROR IN RECORD";Record+852*(Tape-1);"
         CONTINUING"
613   GOTO 760
620   IF ERRN<>52 THEN 650
630   PRINT "ERROR: IMPROPER MASS STORAGE UNIT"
640   GOTO 660
650   PRINT "ERROR:",LIN(1),ERRM$
660   PRINT "PLEASE CORRECT SITUATION AND BEGIN AGAIN"
670   GOTO 180
```

Comment: Duplication routine

```
690 Dupe:    !
700   *                ! SECURED
710   *
720   *
730   *
740   *
750   *
760   *
770   RETURN


790   END
```

# RSTFLX

```
RSTFLX

MAIN PROGRAM

COMMON VARIABLES:

VARIABLES:
Array_rec(*)                40    730    750
Firstrec                    50    350    440    530    700
From$                       20    180    190    220    230    320    410    500
                                  570    720
Lastrec                     60    360    450    540    700
Record                      70    700    710    730    750    760
Tape                        71    311    381    471    710    750
To$                         30    250    260    290    300    570    740

USER DEFINED FUNCTIONS:

SUB PROGRAMS:

JUMP TARGETS:
180                        210    670
220                        190
250                        280
290                        260
610    Error:              100    170
650                        620
660                        640
690    Dupe:               370    460    550
```

# Duplicate A Tape

## Object of Program

This program will totally duplicate a tape's contents onto a second tape. The first tape is copied, one physical record at a time, onto the destination tape. Both tapes must have been initialized at the time this program is run.

## Special Considerations

Because the destination tape's contents are completely obliterated it is imperative that you make sure that THERE IS NOTHING YOU WISH TO KEEP ON YOUR DESTINATION TAPE. There is no way to recover the lost information on that tape.

When this program is done the destination tape will be an exact replica of the original. The files will be at the same addresses on the tape and the directory will be indistinguishable from the original. Files which are protected WILL be copied, and will still be protected. Files of unknown type, with either a blank file type or a question mark following the file type in the directory, will also be copied.

## User Instructions

The program TAPE_DUP is in PROG file "TAPDUP" and must be LOADED.

a. Type: LOAD "TAPDUP"

b. Press: EXECUTE

A 9845B (not a 9845A) is necessary to run this program.

This program utilizes the PHYREC binary to read and write physical records.

### Data to be entered

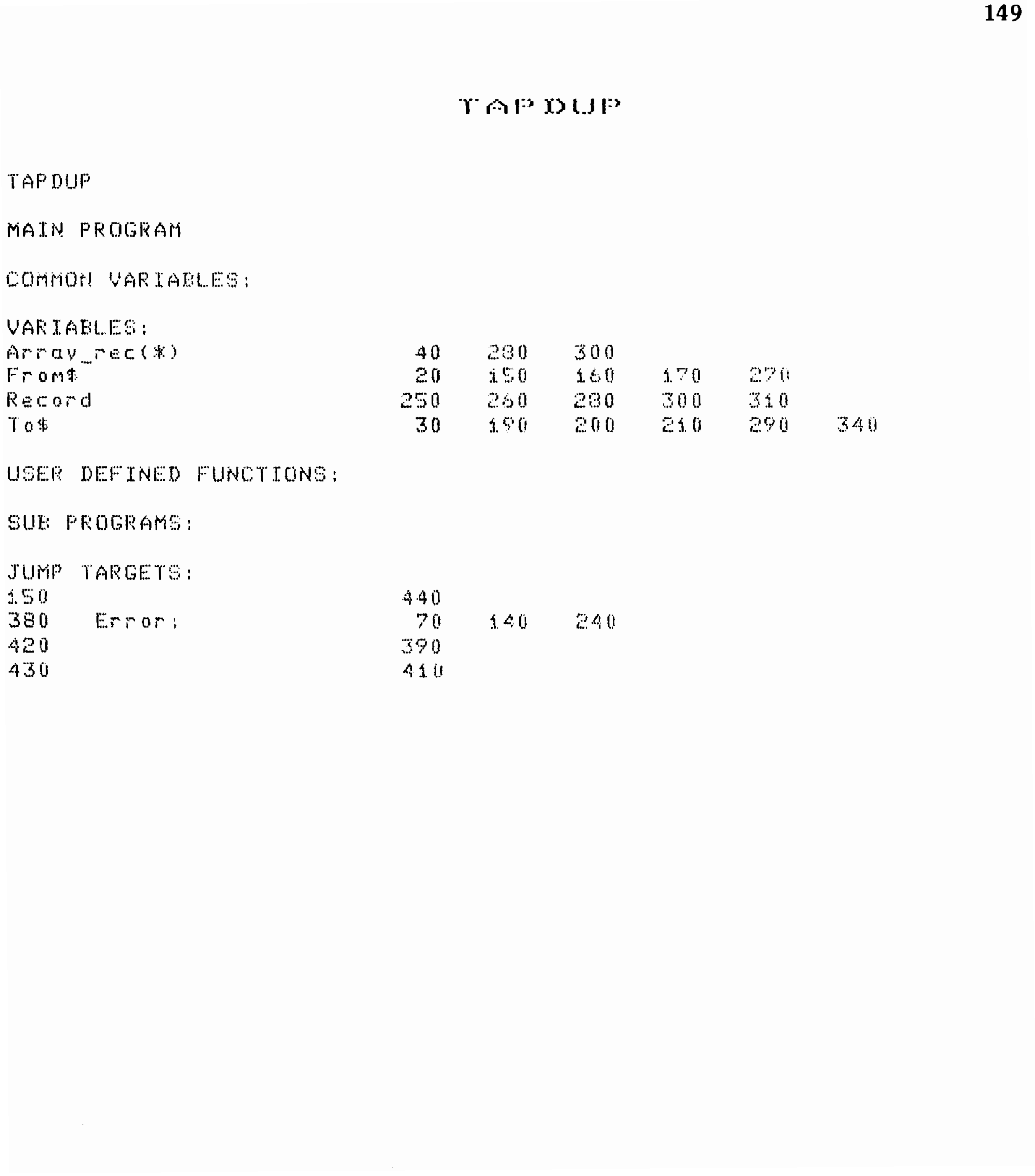
You will be asked to enter the mass storage unit specifier of both the original tape and destination tape (e.g. T14 or T15).

The program will then begin copying physical records from the Original tape to the destination tape, beginning at record 0. The number of the record presently being copied will be displayed. When record 851 has been copied the duplication process will be complete and you will be so notified.

# TAPDUP

```
Comment: THIS PROGRAM DUPLICATES TAPES

20     DIM From$[5]                      ! Comment: ORIGIN TAPE
30     DIM To$[5]                        ! Comment: DESTINATION TAPE
40     INTEGER Array_rec(127)            ! Comment: Array to Hold
          Physical Record
50     PRINTER IS 16

Comment: Enter Tape Address of origin

70     ON ERROR GOTO Error
80     PRINT PAGE
90     PRINT TAB(15);"DUPLICATE CONTENTS OF ONE TAPE TO ANOTHER";LIN(
          2)
100    PRINT "WARNING: This program will totally duplicate the
          original "
110    PRINT TAB(10);"tape to the destination tape. MAKE SURE THERE
          IS NOTHING"
120    PRINT TAB(10);"YOU WISH TO SAVE ON THE DESTINATION TAPE.  If
          you wish to"
130    PRINT TAB(10);"append files see the user manual.";LIN(3)
140    ON ERROR GOTO Error
150    INPUT "ENTER the Mass Storage Code for the Original tape (
          eg. T15 or T14)",From$
160    MASS STORAGE IS ":"&From$
170    PRINT TAB(10);"FROM: ";From$

Comment: Enter Tape Address of destination

190    INPUT "ENTER the mass Storage Code for the Destination Tape
          (eg. T15 or T14)",To$
200    MASS STORAGE IS ":"&To$
210    PRINT TAB(10);"TO: ";To$;LIN(2)
220    PRINT "NOW DUPLICATING TAPE"

Comment: Duplicate Tapes
240    ON ERROR GOTO Error
250    *             ! SECURED
260    *
270    *
280    *
290    *
300    *
310    *
320    PRINT PAGE

Comment: Duplication Complete

340    PRINT TAB(10);CHR$(129);"DUPLICATION OF TAPE FROM ";From$;" TO
```

```
TAPDUP


         ";To$;" COMPLETE";CHR$(128)
350   BEEP
360   STOP

Comment: Error Routine

380 Error:  BEEP
390   IF ERRN()52 THEN 420
400   PRINT "ERROR: IMPROPER MASS STORAGE UNIT"
410   GOTO 430
420   PRINT "ERROR:",LIN(1),ERRM$
430   PRINT "PLEASE CORRECT SITUATION AND BEGIN AGAIN"
440   GOTO 150
450   END
```



```
                               TAPDUP

TAPDUP

MAIN PROGRAM

COMMON VARIABLES:

VARIABLES:
Array_rec(*)                40   280   300
From$                       20   150   160   170   270
Record                     250   260   280   300   310
To$                         30   190   200   210   290   340

USER DEFINED FUNCTIONS:

SUB PROGRAMS:

JUMP TARGETS:
150                        440
380   Error:                70   140   240
420                        390
430                        410
```

# Duplicate A 9885 Flexible Disk

## Object of Program

This program will duplicate a disk's contents onto a second disk. The first disk is copied, one physical record at a time, onto the destination disk. Both disks must have been initialized at the time this program is run.

## Special Considerations

Because the destination disk's contents are completely obliterated it is imperative that you make sure that THERE IS NOTHING YOU WISH TO KEEP ON YOUR DESTINATION DISK. There is no way to recover the lost information on that disk.

When this program is done the destination disk will be an exact replica of the original. The files will be the same addresses on the disk and the directory will be indistinguishable from the original. Files which are protected WILL be copied, and will still be protected. Files of unkown type, with either a blank file type or a question mark following the file type in the directory, will also be copied.

## User Instructions

The program FLEX_DUP is in PROG file "FLXDUP" and must be LOADED.

a. Type: LOAD "FLXDUP"

b. Press: EXECUTE

A 9845B (not a 9845A) is necessary to run this program.

This program utilizes the PHYREC binary to read and write physical records.

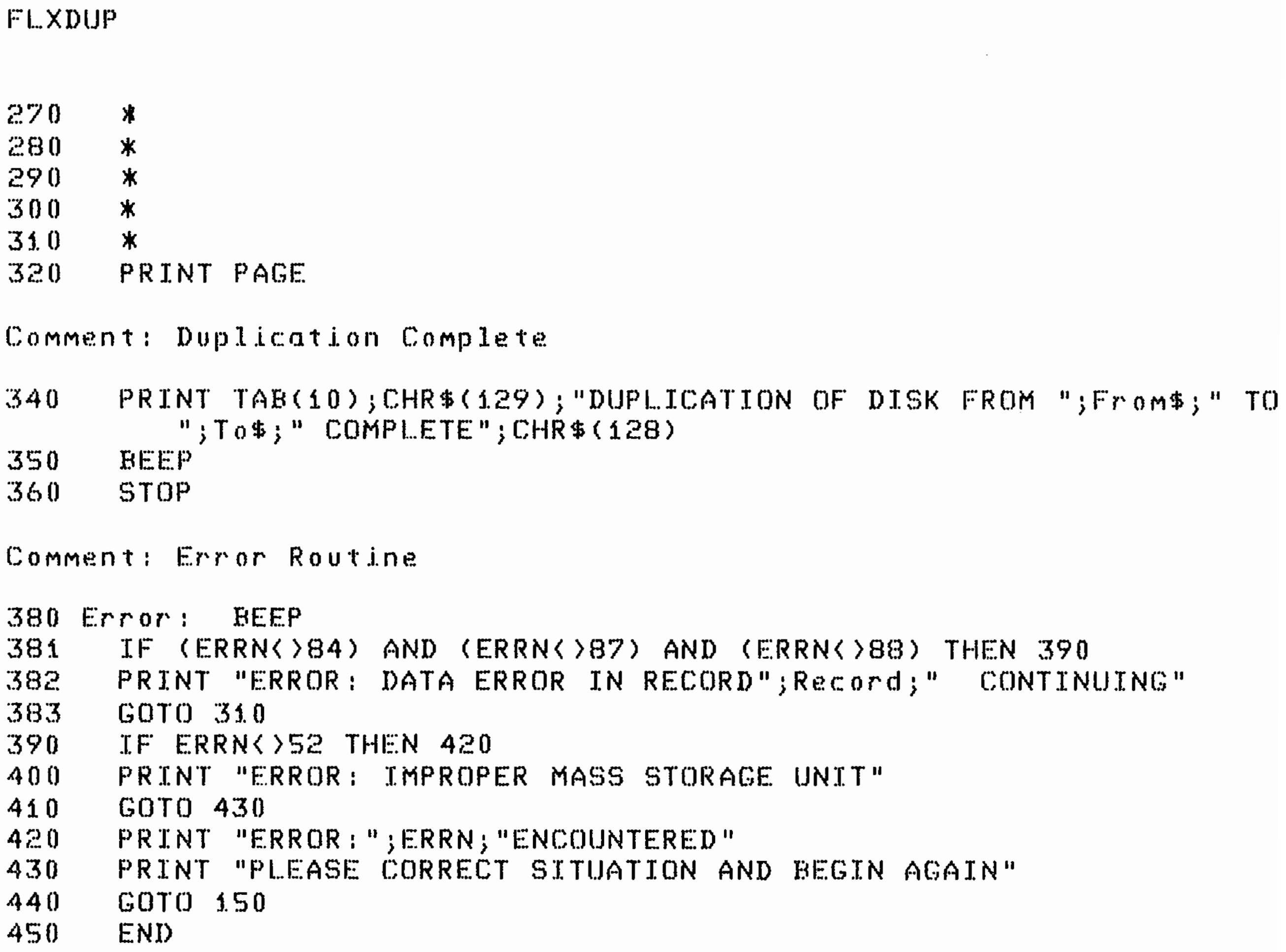
### Data to be entered

You will be asked to enter the mass storage unit specifier of both the original 9885 flexible disk and destination disk (e.g. F8 or F8,1).

The program will then begin copying physical records from the original disk to the destination disk, beginning at record 0. The number of the record presently being copied will be displayed. When record 2009 has been copied the duplication process will be complete and you will be so notified.

# FLXDUP

```
Comment: THIS PROGRAM DUPLICATES 9885 FLEXIBLE DISKS

20     DIM From$[5]                       ! Comment: Origin flexible disk
30     DIM To$[5]                         ! Comment: Destination
          flexible disk
40     INTEGER Array_rec(127)             ! Comment: Array to Hold
          Physical Record
41     INTEGER Record                     ! Comment: Record counter
50     PRINTER IS 16

Comment: Enter Disk Address of origin

70     ON ERROR GOTO Error
80     PRINT PAGE
90     PRINT TAB(15);"DUPLICATE CONTENTS OF ONE 9885 FLEXIBLE DISK TO
          ANOTHER";LIN(2)
100    PRINT "WARNING: This program will totally duplicate the
          original "
110    PRINT TAB(10);"disk to the destination disk. MAKE SURE THERE
          IS NOTHING"
120    PRINT TAB(10);"YOU WISH TO SAVE ON THE DESTINATION DISK!  IF
          you wish to"
130    PRINT TAB(10);"append files see the user manual for the file
          manager program.";LIN(3)
140    ON ERROR GOTO Error
150    LINPUT "ENTER the Mass Storage Code for the Original disk (
          eg. F8  or F8,1)",From$
151    IF From$[1,1]="F" THEN 160
152       BEEP
153       GOTO 150
160    MASS STORAGE IS ":"&From$
170    PRINT TAB(10);"FROM: ";From$

Comment: Enter Disk Address of destination

190    LINPUT "ENTER the Mass Storage Code for the Destination
          disk (eg. F8 or F8,1)",To$
191    IF To$[1,1]="F" THEN 200
192      BEEP
193      GOTO 190
200    MASS STORAGE IS ":"&To$
210    PRINT TAB(10);"TO: ";To$;LIN(2)
220    PRINT "NOW DUPLICATING 9885 DISK"

Comment: Duplicate Tapes

240    ON ERROR GOTO Error
250    *                      ! SECURED
260    *
```



```
FLXDUP


270    *
280    *
290    *
300    *
310    *
320    PRINT PAGE

Comment: Duplication Complete

340    PRINT TAB(10);CHR$(129);"DUPLICATION OF DISK FROM ";From$;" TO
          ";To$;" COMPLETE";CHR$(128)
350    BEEP
360    STOP

Comment: Error Routine

380 Error:  BEEP
381    IF (ERRN<>84) AND (ERRN<>87) AND (ERRN<>88) THEN 390
382    PRINT "ERROR: DATA ERROR IN RECORD";Record;"  CONTINUING"
383    GOTO 310
390    IF ERRN<>52 THEN 420
400    PRINT "ERROR: IMPROPER MASS STORAGE UNIT"
410    GOTO 430
420    PRINT "ERROR:";ERRN;"ENCOUNTERED"
430    PRINT "PLEASE CORRECT SITUATION AND BEGIN AGAIN"
440    GOTO 150
450    END
```

# FLXDUP

```
FLXDUP

MAIN PROGRAM

COMMON VARIABLES:

VARIABLES:
Array_rec(*)                   40     280     300
From$                          20     150     151     160     170     270     340
Record                         41     250     260     280     300     310
To$                            30     190     191     200     210     290     340

USER DEFINED FUNCTIONS:

SUB PROGRAMS:

JUMP TARGETS:
150                           153     440
160                           151
190                           193
200                           191
380   Error:                   70     140     240
420                           390
430                           410
```

# Bases Program

## Object of Program

This program does base conversion on numbers in bases 2 through 36. It will convert whole numbers (e.g. 123), fractional numbers (e.g. .231), as well as combinations (e.g. 132.312). Due to the fact that digits in bases greater than 10 may exceed 10 it is necessary to use the letters A through Z to indicate the digits 10 through 35 (e.g. 15 in base 20 is F, 45 in base 20 is 1F).

## Special Considerations

The fact that some of the digits may have an alphabetic representation necessitates entering the number in string form. The string is initially converted to base 10 from the Input base for evaluation and then converted to the Output base to be presented. The integer and fractional portions of the number are converted separately. Should the fractional portion evaluate to 1 (as may happen through rounding) 1 is added to the integer portion and the fractional portion is dropped.

Due to the fact that numbers represented in the lower bases may be quite long the length of the string which is allowed to be entered is considerable. If a string is entered which evaluates to a number greater than the integer precision of the machine then the output number will be followed by the indication "(Approx.)", although it is possible that the answer is still exact. Should the size of the entered number be too great a message indicating "OVERFLOW" will be shown. No result will be given in this case.

Leading zeroes will be removed from the results. However, fractional results will be printed as obtained, although, the lower order digits may not be significant. It is up to the user to decide what accuracy is desired. Some round off error may be noticeable.

## Conversion Table

| | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 |
| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |

## User Instructions

The program BASES is in the DATA file "BASES"

a. Type: GET "BASES"

b. Press: EXECUTE

### Data to be entered

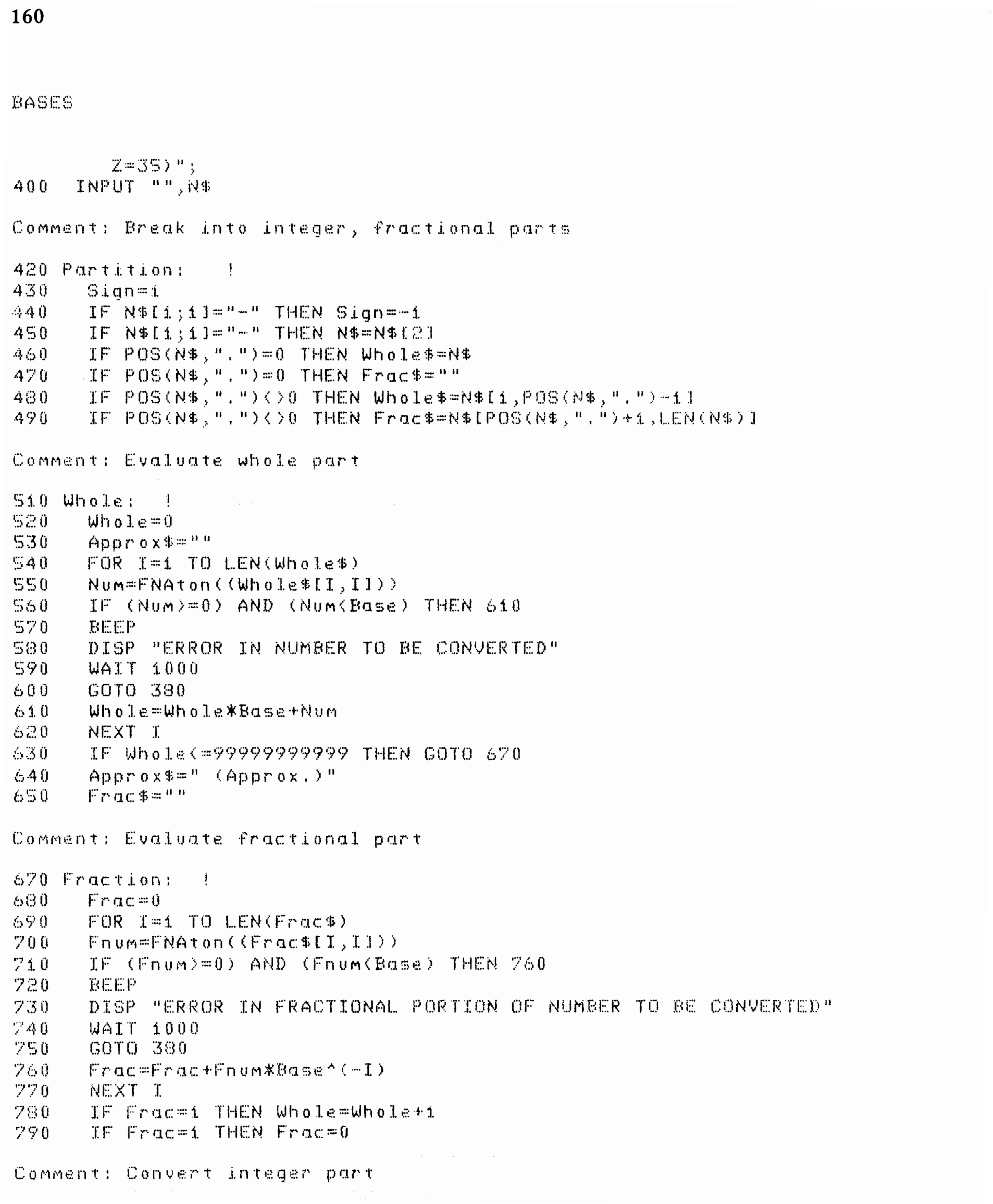
You will be asked to enter the Input base and the Output base. These numbers may be between 2 and 36. They may be entered as numbers (e.g. 15) or, if they are greater than 10, as alphabetic digits (e.g. F). These numbers wil be printed on the CRT. Under them will be printed the numbers you enter in the Input base and the results of the conversion in the Output base.

The number you enter should contain no digits as large as or larger than the Input base as this is not a proper representation of a number in that base. Leading zeroes will cause no problems.

The result of the conversion will be printed under the Output base. When it has been printed you will be asked to enter another number. If you do not wish to do so press STOP. If you wish to convert from or to a different base, then you may press RUN and enter new values for Input base and Output base.

# BASES

```
         CONVERT BASES

20     DIM Approx$[20]                   !Comment: approximation message
30     INTEGER Base                      !Comment: base in
40     DIM Base$[3]                      !Comment: base in - character form
50     INTEGER Cbase                     !Comment: base out
60     REAL Cfnum                        !Comment: converted fraction
70     DIM Cfrac$[40]                    !Comment: fractional part of
          converted number
80     REAL Cnum                         !Comment: converted whole number
90     DIM Cwhole$[40]                   !Comment: whole part of converted
          number
100    INTEGER Flagz                     !Comment: flag for fraction is 0
110    INTEGER Fnum                      !Comment: fractional digit
120    REAL Frac                         !Comment: fractional value
130    DIM Frac$[40]                     !Comment: fractional part of input
          number
140    INTEGER I                         !Comment: loop counter
150    DIM N$[50]                        !Comment: Input Number
160    INTEGER Num                       !Comment: converted number
170    INTEGER Sign                      !Comment: =1 if positive; =-1 if
          negative
180    REAL Whole                        !Comment: converted whole part
190    DIM Whole$[40]                    !Comment: Integer part of input
          number

Comment: Program header

210    PRINT LIN(2),SPA(20),"BASE CONVERSION PROGRAM",LIN(2)

Comment: Input bases and number to be converted
230    INPUT "ENTER INPUT BASE , OUTPUT BASE (values 2 through 36
          accepted)",Base$,Cbase$
240    Base=FNAton(Base$)
250    IF (Base>1) AND (Base<37) THEN GOTO 300
260     BEEP
270     DISP "ERROR IN 'INPUT BASE'"
280     WAIT 1000
290     GOTO 230
300    Cbase=FNAton(Cbase$)
310    IF (Cbase>1) AND (Cbase<37) THEN GOTO 360
320     BEEP
330     DISP "ERROR IN 'OUTPUT BASE'"
340     WAIT 1000
350     GOTO 230
360    PRINT CHR$(132)&"BASE ";Base;CHR$(128),SPA(17);
370    PRINT CHR$(132)&"BASE ";Cbase;CHR$(128)
380 Input:  DISP "ENTER NUMBER TO BE CONVERTED ";
390   IF Base>9 THEN DISP "(for digits greater than 10: A=10,B=11,...,
```



```
BASES

        Z=35)";
400  INPUT "",N$

Comment: Break into integer, fractional parts

420 Partition:   !
430   Sign=1
440   IF N$[1;1]="-" THEN Sign=-1
450   IF N$[1;1]="-" THEN N$=N$[2]
460   IF POS(N$,".")=0 THEN Whole$=N$
470   IF POS(N$,".")=0 THEN Frac$=""
480   IF POS(N$,".")<>0 THEN Whole$=N$[1,POS(N$,".")-1]
490   IF POS(N$,".")<>0 THEN Frac$=N$[POS(N$,".")+1,LEN(N$)]

Comment: Evaluate whole part

510 Whole:  !
520   Whole=0
530   Approx$=""
540   FOR I=1 TO LEN(Whole$)
550   Num=FNAton((Whole$[I,I]))
560   IF (Num>=0) AND (Num<Base) THEN 610
570   BEEP
580   DISP "ERROR IN NUMBER TO BE CONVERTED"
590   WAIT 1000
600   GOTO 380
610   Whole=Whole*Base+Num
620   NEXT I
630   IF Whole<=99999999999 THEN GOTO 670
640   Approx$=" (Approx.)"
650   Frac$=""

Comment: Evaluate fractional part

670 Fraction:  !
680   Frac=0
690   FOR I=1 TO LEN(Frac$)
700   Fnum=FNAton((Frac$[I,I]))
710   IF (Fnum>=0) AND (Fnum<Base) THEN 760
720   BEEP
730   DISP "ERROR IN FRACTIONAL PORTION OF NUMBER TO BE CONVERTED"
740   WAIT 1000
750   GOTO 380
760   Frac=Frac+Fnum*Base^(-I)
770   NEXT I
780   IF Frac=1 THEN Whole=Whole+1
790   IF Frac=1 THEN Frac=0

Comment: Convert integer part
```

```
810 Convertwhole:!
820   Flagz=1
830   Cwhole$=""
840   FOR I=MAX(20,Base-Cbase) TO 1 STEP -1
850   Cnum=Whole DIV Cbase^(I-1)
860   IF Cnum>Cbase-1 THEN GOTO Overflow
870   IF Cnum>0 THEN Flagz=0
880   Whole=Whole MOD Cbase^(I-1)
890   IF NOT Flagz THEN Cwhole$=Cwhole$&FNNtoa$(Cnum)
900   NEXT I
```

Comment: Convert fractional part

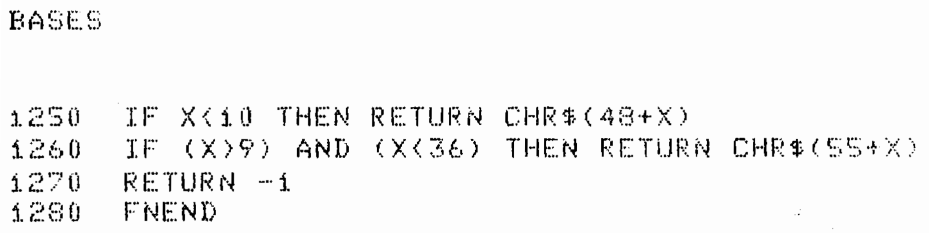
```
920 Convertfrac:  !
930   Cfrac$=""
940   IF Frac=0 THEN 1010
950   FOR I=1 TO 30
960   Cfnum=INT(Frac*Cbase^I)
970   Frac=Frac-Cfnum*Cbase^(-I)
980   Cfrac$=Cfrac$&FNNtoa$(Cfnum)
990   NEXT I
```

Comment: Output result

```
1010  IMAGE 28A,7X,45A
1020  IF (Cwhole$="") AND (Cfrac$="") THEN Cwhole$="0"
1030  IF Sign=-1 THEN Cwhole$="-"&Cwhole$
1040  IF Sign=-1 THEN N$="-"&N$
1050  PRINT USING 1010;N$[1,28],Cwhole$&"."&Cfrac$&Approx$
1060  GOTO Input
1070 Overflow: !
1080  DISP "OVERFLOW"
1090  BEEP
1100  WAIT 1000
1110  GOTO Input
```

Comment:  CONVERSION FUNCTIONS

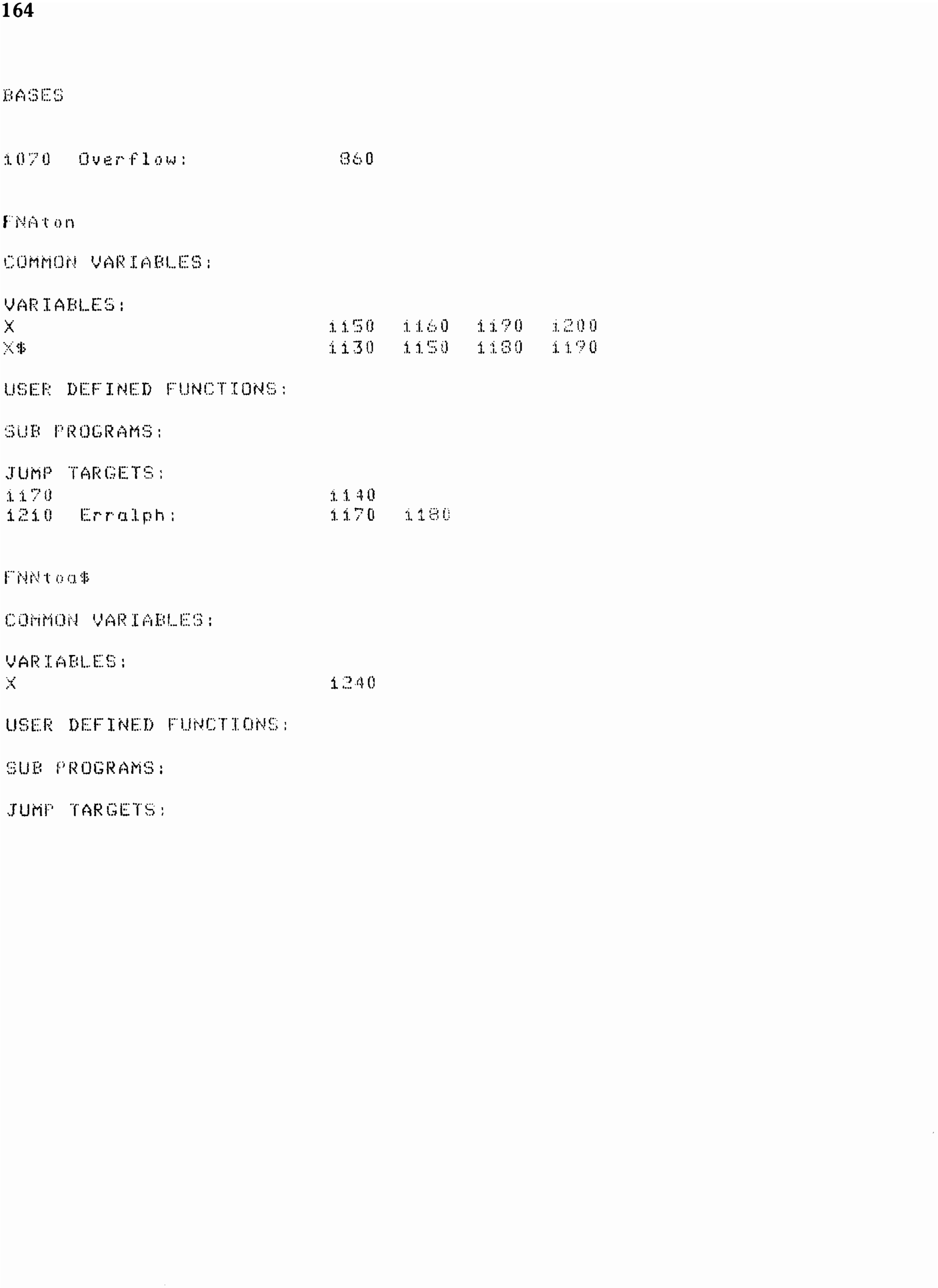
```
1130  DEF FNAton(X$)
1140  ON ERROR GOTO 1170
1150  X=VAL(X$)
1160  RETURN X
1170  ON ERROR GOTO Erralph
1180  IF LEN(X$)<>1 THEN Erralph
1190  X=NUM(X$)
1200  IF (X>64) AND (X<91) THEN RETURN X-55
1210 Erralph:  !
1220  RETURN -1
1230  FNEND
1240  DEF FNNtoa$(X)
```



```
BASES


1250  IF X<10 THEN RETURN CHR$(48+X)
1260  IF (X>9) AND (X<36) THEN RETURN CHR$(55+X)
1270  RETURN -1
1280  FNEND
```

# BASES

```
BASES

MAIN PROGRAM

COMMON VARIABLES:

VARIABLES:
Approx$                   20     530     640    1050
Base                      30     240     250     360     390     560     610     710
                                 760     840
Base$                     40     230     240
Cbase                     50     300     310     370     840     850     860     880
                                 960     970
Cbase$                   230     300
Cfnum                     60     960     970     980
Cfrac$                    70     930     980    1020    1050
Cnum                      80     850     860     870     890
Cwhole$                   90     830     890    1020    1030    1050
Flagz                    100     820     870     890
Fnum                     110     700     710     760
Frac                     120     680     760     780     790     940     960     970
Frac$                    130     470     490     650     690     700
I                        140     540     550     620     690     700     760     770
                                 840     850     880     900     950     960     970
                                 990
N$                       150     400     440     450     460     470     480     490
                                1040    1050
Num                      160     550     560     610
Sign                     170     430     440    1030    1040
Whole                    180     520     610     630     780     850     880
Whole$                   190     460     480     540     550

USER DEFINED FUNCTIONS:
FNAton                   240     300     550     700
FNNtoa$                  890     980

SUB PROGRAMS:

JUMP TARGETS:
230                      290     350
300                      250
360                      310
380    Input:            600     750    1060    1110
420    Partition:
510    Whole:
610                      560
670    Fraction:         630
760                      710
810    Convertwhole:
920    Convertfrac:
1010                     940    1050
```



```
BASES


1070  Overflow:            860


FNAton

COMMON VARIABLES:

VARIABLES:
X                         1150  1160  1170  1200
X$                        1130  1150  1180  1190

USER DEFINED FUNCTIONS:

SUB PROGRAMS:

JUMP TARGETS:
1170                      1140
1210  Erralph:            1170  1180


FNNtoa$

COMMON VARIABLES:

VARIABLES:
X                         1240

USER DEFINED FUNCTIONS:

SUB PROGRAMS:

JUMP TARGETS:
```

# Convert Binary To Decimal

## Object of Function

This multiline function converts numbers entered as a binary string to their decimal equivalent. The binary string is assumed to be the representation of a word of memory (16 bits). If the string is 16 characters long and the leftmost character is "1" then the string is interpreted to be a 2's complement representation of a negative number. For example, "0111111111111111" is converted to 32767 while "1111111111111111" is converted to −1.

If you are not familiar with 2's compliment representation see any book on machine arithmetic.

The input string can be from 1 to 16 characters in length. All characters in the string must either be "1" or "0" ("−" is not accepted).

## User Instructions

The function FNBin_to_dec is in DATA file "CNVb−d"

This is a multiline function and must be attached to your main program. See the Introduction for instructions for linking files.

### Calling Syntax

Y=FNBin_to_dec(X$)

### Input parameters

1) X$:String representation of a binary number; all characters must be either "0" or "1"; negative numbers should be in the form of a 16 bit 2's compliment number; the string should not exceed 16 characters.

### Returns

1) Integer valued REAL value between 32767 and −32768

A simple driver program to use this conversion function can be set up as follows.

```
10 DIM X$[16]
20 INPUT "Enter the binary string",X$
30 Y=FNBin_to_dec(X$)
40 PRINT "Decimal value:",Y
50 GOTO 20
```

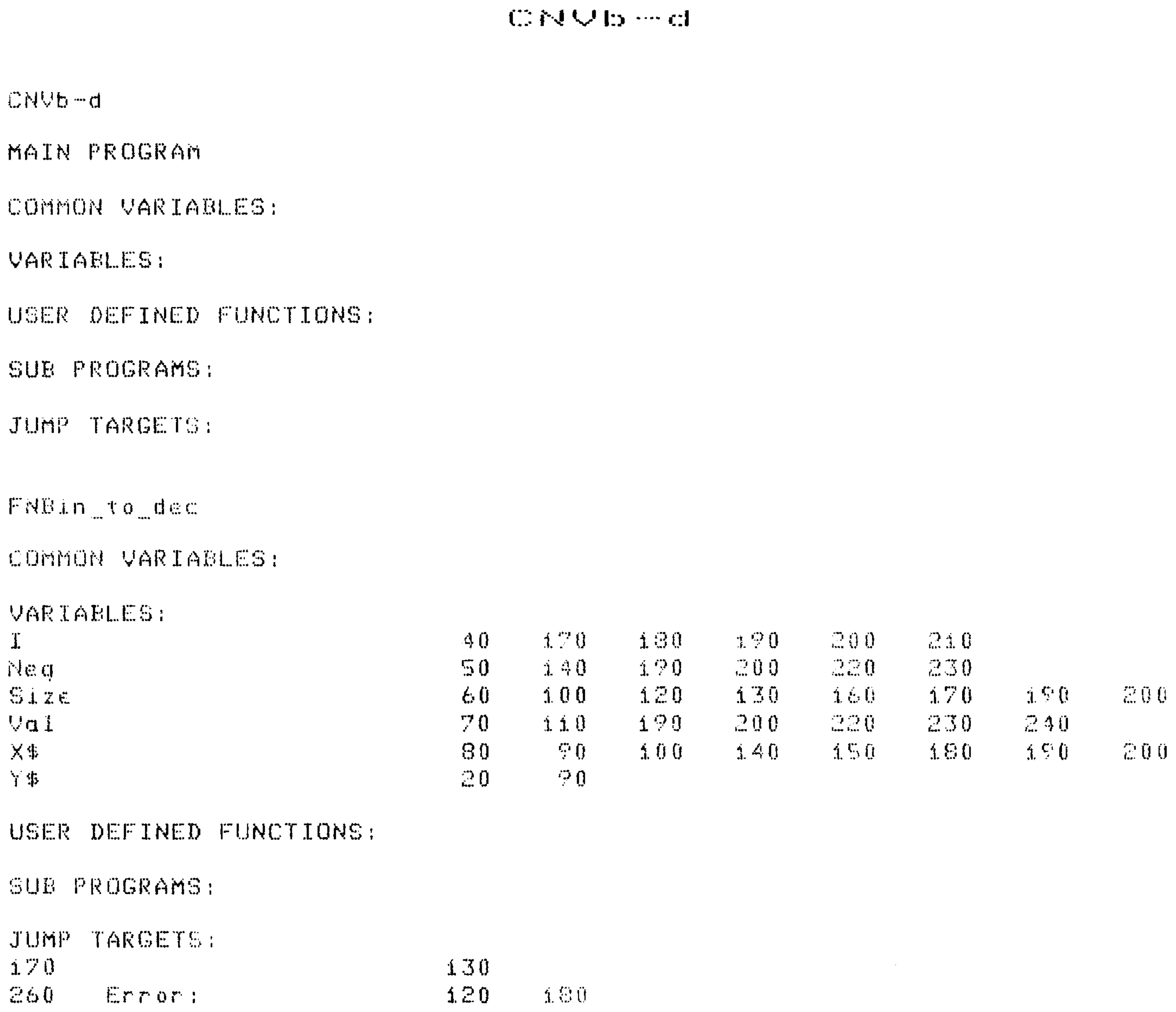
## CNVb-d

Comment: Subprogram to convert binary string to decimal number

```
20    DEF FNBin_to_dec(Y$)


40    INTEGER I                       !Comment: loop counter
50    INTEGER Neg                     !Comment: flag for negative number (
         2's compliment)
60    INTEGER Size                    !Comment: length of parameter string
70    REAL Val                        !Comment: decimal value of parameter
80    DIM X$[20]                      !Comment: substitute for parameter
90      X$=Y$
100     Size=LEN(X$)
110     Val=0
120     IF Size>16 THEN Error
130     IF Size<16 THEN 170
140     IF X$[1;1]="1" THEN Neg=1
150     X$=X$[2]
160     Size=Size-1
170        FOR I=1 TO Size
180        IF (X$[I;1]<>"1") AND (X$[I;1]<>"0") THEN Error
190        IF NOT Neg THEN Val=Val+2^(Size-I)*(X$[I;1]="1")
200        IF Neg THEN Val=Val+2^(Size-I)*(X$[I;1]="0")
210        NEXT I
220        IF Neg THEN Val=-(Val+1)
230        IF Neg AND (Val=0) THEN Val=-32768
240    RETURN Val
```

Comment: Error subroutine: displays error message; returns 0

```
260 Error: DISP "ERROR in parameter to Binary/Decimal conversion
        subprogram"
270   BEEP
280   WAIT 3000
290   RETURN 0
300   FNEND
```



```
                              CNVb-d

CNVb-d

MAIN PROGRAM

COMMON VARIABLES:

VARIABLES:

USER DEFINED FUNCTIONS:

SUB PROGRAMS:

JUMP TARGETS:


FNBin_to_dec

COMMON VARIABLES:

VARIABLES:
I                          40    170   180   190   200   210
Neg                        50    140   190   200   220   230
Size                       60    100   120   130   160   170   190   200
Val                        70    110   190   200   220   230   240
X$                         80     90   100   140   150   180   190   200
Y$                         20     90

USER DEFINED FUNCTIONS:

SUB PROGRAMS:

JUMP TARGETS:
170                       130
260   Error:              120   180
```

# Convert Decimal To Binary

## Object of Function

This multilined function converts a decimal INTEGER to a binary string. The acceptable inputs for the decimal number are all INTEGER values (−32768 to 32767). For example, if the input parameter had value 65 the function would return the string "1000001".

This function assumes that the binary representation is desired in the form of a word in memory. Therefore, negative decimal numbers are represented as a 2's compliment binary number in a 16 bit format. For example, −1 is represented as "1111111111111111" and −5 converts to "1111111111111011".

If you are not familiar with 2's compliment representation you can find a description in any book on machine arithmetic.

If you prefer the conventional notation for handling negative numbers (e.g. −1001101), use the all-purpose base conversion routine, BASES.

## User Instructions

The function FNDec_to_bin$ is in DATA file "CNVd−b"

This is a multiline function and must be attached to your main program. See the Introduction for instructions for linking files.

### Calling Syntax

Y$=FNDec_to_bin$(X)

### Input parameters

1) X: INTEGER value or variable (32767 to −32768)

### Returns

1) String of "1"'s and "0"'s representing the binary value of X; If X is negative then the string will be a 2's compliment representation of that number in a 16 bit format.

A simple driver can be constructed to use this conversion function as follows:

```
10 INTEGER X
20 DIM Y$[20]
30 INPUT "Enter decimal value",X
40 Y$=FNDec_to_bin$(X)
50 PRINT "Binary value:",Y$
60 GOTO 30
```

## CNVd-b

Comment: Function - convert decimal integer to binary string

```
20   DEF FNDec_to_bin$(INTEGER Y)


40     DIM Val$[20]              !Comment: octal value string
50     REAL N                    !Comment: utility number
60     REAL X                    !Comment: substitute for parameter


80     N=65536
90     X=(Y+N) MOD N
100       FOR I=15 TO 0 STEP -1
110       N=N DIV 2
120       Val$=Val$&VAL$(X DIV N)
130       X=X MOD N
140       NEXT I
150   RETURN Val$
160  FNEND
```

## CNVd-b

```
CNVd-b

MAIN PROGRAM

COMMON VARIABLES:

VARIABLES:

USER DEFINED FUNCTIONS:

SUB PROGRAMS:

JUMP TARGETS:


FNDec_to_bin$

COMMON VARIABLES:

VARIABLES:
I                            100   140
N                             50    80    90   110   120   130
Val$                          40   120   150
X                             60    90   120   130
Y                             20    90

USER DEFINED FUNCTIONS:

SUB PROGRAMS:

JUMP TARGETS:
```

# Convert Octal To Decimal

## Object of Function

This multiline function converts numbers in integer octal representation to decimal representation. The octal representation must be a REAL number between −77777 and −77777 and must not contain the digits 8 or 9. The number must be an integer value. (See BASES for fraction conversion.) For example, if the input number was 101, the value returned value would be 65, the decimal equivalent of the octal 101.

## User Instructions

The function FNOct_to_dec is in DATA file "CNVo−d"

This is a multiline function and must be attached to your main program. See the Introduction for instructions for linking files.

### Calling Syntax

Y=FNOct_to_dec(X)

### Input parameters

1) X: INTEGER valued number or variable between −77777 and 77777 with no digit exceeding 7.

### Returns

1) Integer valued REAL representing the decimal equivalent of X

A simple driver can be built to use this conversion function as follows:

```
10 INPUT "Enter octal value",X
20 Y=FNOct_to_dec(X)
30 PRINT "Decimal value:",Y
40 GOTO 10
```

## CNVo-d

Comment: Subprogram to convert octal numbers to decimal

```
20    DEF FNOct_to_dec(Y)


40    DIM Digits$[20]              !Comment: string representation of
         octal number
50    INTEGER I                    !Comment: loop counter
60    INTEGER Sign                 !Comment: =1 for positive; =-1 for
         negative
70    REAL Val                     !Comment: decimal value of number
80    REAL X                       !Comment: substitiute for parameter


100     X=Y
110     Digits$=VAL$(X)
120     IF (POS(Digits$,"8")<>0) OR (POS(Digits$,"9")<>0) THEN Error
130     Val=0
140     Sign=1
150     IF X>=0 THEN 180
160     Sign=-1
170     X=-X
180     IF X>77777 THEN Error
190        FOR I=4 TO 0 STEP -1
200        Val=Val+X DIV 10^I*8^I
210        X=X MOD 10^I
220        NEXT I
230    RETURN Val*Sign
```

Comment: Error routine : returns 0; displays error message

```
250 Error:   DISP "ERROR in parameter to Octal/Decimal conversion
        subprogam"
260   BEEP
270   WAIT 3000
280   RETURN 0
290   FNEND
```

# CNVo-d

```
CNVo-d

MAIN PROGRAM

COMMON VARIABLES:

VARIABLES:

USER DEFINED FUNCTIONS:

SUB PROGRAMS:

JUMP TARGETS:


FNOct_to_dec

COMMON VARIABLES:

VARIABLES:
Digits$                     40   110   120
I                           50   190   200   210   220
Sign                        60   140   160   230
Val                         70   130   200   230
X                           80   100   110   150   170   180   200   210
Y                           20   100

USER DEFINED FUNCTIONS:

SUB PROGRAMS:

JUMP TARGETS:
180                        150
250   Error:               120   180
```

# Convert Decimal To Octal

## Object of Function

This multiline function converts a decimal INTEGER to an octal integer. The octal number is returned as a REAL number and is octal for display purposes only. For example, if the input decimal parameter was 65 this subprogram would return 101, the octal representation of 65. However, if this returned number was then used in a mathematical expression its value would be decimal 101.

This multiline function will convert any decimal integer between −32767 and 32767 to its octal representation. −32768 is not an acceptable input.

## User Instructions

The function FNDec_to_oct is in DATA file "CNVd−o"

This is a multiline function and must be attached to your main program. See the Introduction for instructions for linking files.

### Calling Syntax

Y=FNDec_to_oct(X)

### Input parameters

1) X: INTEGER value or variable between 32767 and −32767

### Returns

1) REAL number representing the octal equivalent of X

A simple driver to use this conversion function can be constructed as follows:

```
10 INTEGER X
20 INPUT "Enter decimal value",X
30 Y=FNDec_to_oct(X)
40 PRINT "Octal value:",Y
50 GOTO 20
```

# CNVd—o

```
Comment: Multiline function: convert decimal to octal integers

20    DEF FNDec_to_oct(INTEGER X)
30      INTEGER Dec                      !Comment: New variable so as not
           to
40                                       !          alter parameter
50      DIM Val$[6]                      !Comment: String to hold returned
           value
60      INTEGER Sign                     !Comment: Sign of parameter (+1
           or -1)
70      REAL Val                         !Comment: Value to be returned
80      Val=0                            !Comment: initialize value to be
           returned
90      Dec=X                            !Comment: Do not alter parameter
100     Sign=1                           !Comment: initialize sign to +
110     IF Dec>=0 THEN 180
120     Sign=-1                          !Comment: sign is -
130       IF Dec<>-32768 THEN 160        !Comment: Avoid overflow of
             integer variable
140       Val$="100000"
150       GOTO 220
160     Dec=-Dec

Comment: Conversion

180        FOR I=4 TO 0 STEP -1
190        Val$=Val$&VAL$(Dec DIV 8^I)
200        Dec=Dec MOD 8^I
210        NEXT I
220    RETURN VAL(Val$)*Sign
230   FNEND
```

# CNVd-o

```
CNVd-o

MAIN PROGRAM

COMMON VARIABLES:

VARIABLES:

USER DEFINED FUNCTIONS:

SUB PROGRAMS:

JUMP TARGETS:


FNDec_to_oct

COMMON VARIABLES:

VARIABLES:
Dec                         30    90   110   130   160   190   200
I                          180   190   200   210
Sign                        60   100   120   220
Val                         70    80
Val$                        50   140   190   220
X                           20    90

USER DEFINED FUNCTIONS:

SUB PROGRAMS:

JUMP TARGETS:
160                        130
180                        110
220                        150
```

# 98036A Setup Subprogram-Prompter Version

## Object of Subprogram

This subprogram allows you to set up a 98036A I/O card. It asks you to enter the select code of the 98036A card, the number of stop bits, whether or not parity is to be enabled, if the parity is to be odd or even (if enabled), the number of bits per character, the baud rate (for checking purposes only) and the bit rate factor.

When the information has all been entered, a series of I/O commands are used to address the proper I/O register and set the factors specified. Any information entered that does not conform to the parameters required by the 98036A will be flagged so that you can reenter them.

This subprogram checks for a usable interface and for the existence of a 98036A card.

For more detailed information concerning the setup of the 98036A card, see the HP System 45B I/O ROM Programming Manual or the 98036A manual.

This version of the subprogram explicitly requests each piece of information. It is useful if you wish to change your 98036A set up often or are somewhat unfamiliar with the parameters to be entered. If you only wish to use one setup and are comfortable with the parameters you may wish to use the parameter version of this program (see 98036A SETUP—Parameter Version).

## User Instructions

The 98036A SETUP Subprogram—Prompter version is in DATA file "SET36"

The I/O ROM is necessary for this subprogram, as is a 98036A card.

This is a subprogram and must be attached to your main program. See the Introduction for instructions for linking files.

**Calling Syntax**

```
CALL Set_up_98036
```

**Data to be entered**

You will be asked to enter the following data:

1) The select code of the 98036A

2) The number of stop bits (1,1.5 or 2)

3) Whether or not the parity is to be enabled

4) Whether the parity is odd or even (if enabled)

5) The number of bits per character (5,6,7 or 8)

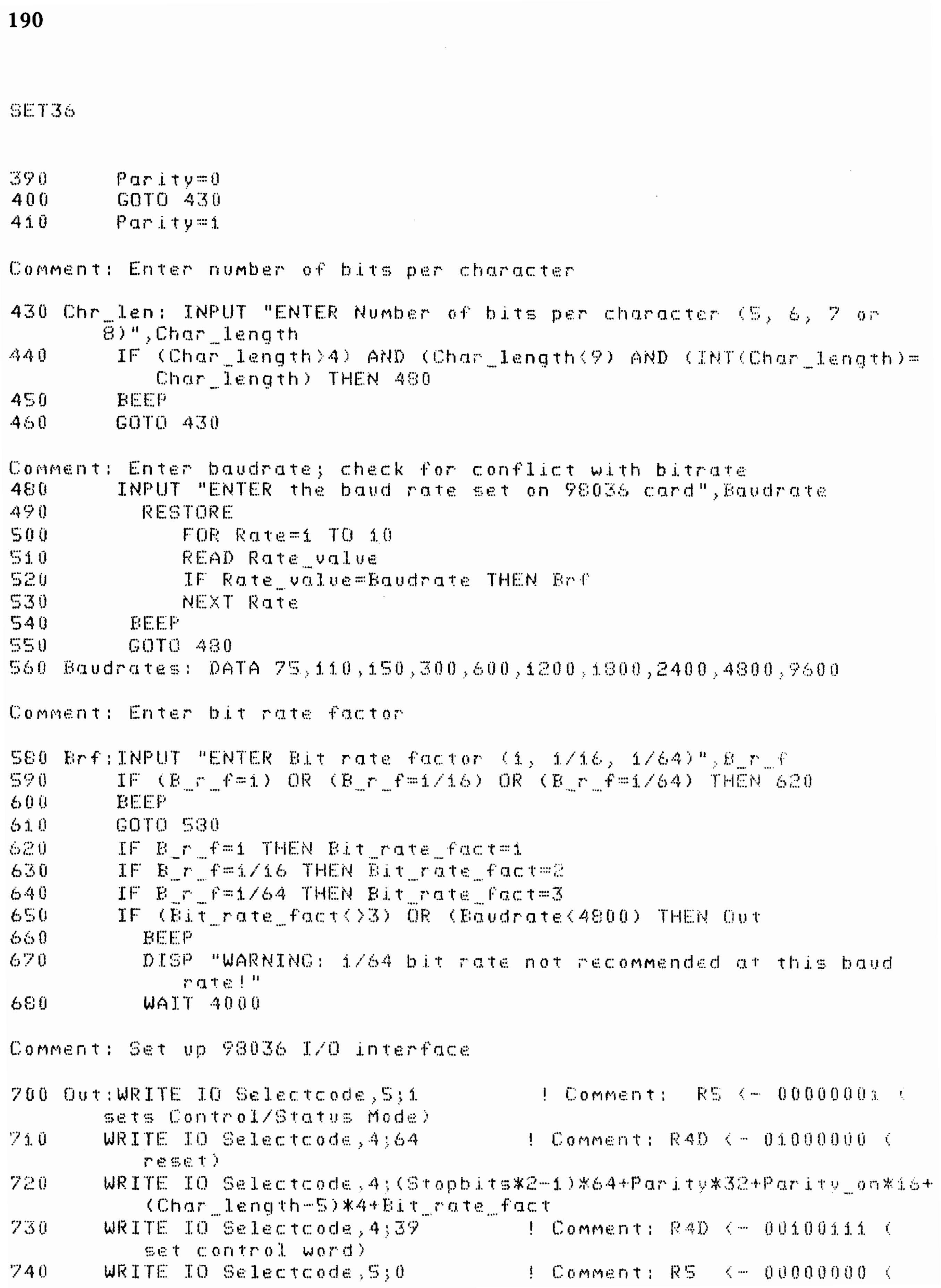
6) The baud rate (this is only necessary to check for compatability with the bit rate factor)

7) The bit rate factor (1,1/16, 1/64)

When this information has been input the program will output the proper series I/O commands to set up the card as specified and reset the interface for interrupts.

# SET36

```
Comment: Subroutine to set up 98036A I/O card

20    SUB Set_up_98036

Comment: Enter select code for 98036

40    INPUT "ENTER the select code of the I/O card",Selectcode
50    IF (Selectcode>0) AND (Selectcode<16) THEN 80
60        BEEP
70        GOTO 40
80      IF IOSTATUS(Selectcode) THEN Stat        ! Comment:  check
           interface
90      BEEP
100     DISP "NO OPERABLE INTERFACE"
110     WAIT 3000
120     GOTO 40
130 Stat:     STATUS Selectcode;Status           ! Comment: check for
        98036
140     IF BINAND(Status,48)=16 THEN 200
150 Error:  BEEP
160   DISP "NO 98036 INTERFACE; PLEASE REENTER Selectcode"
170   WAIT 3000
180   GOTO 40

Comment: Enter stop bits

200   INPUT "ENTER the number of Stop Bits (1, 1.5 or 2)",Stopbits
210   IF (Stopbits=1) OR (Stopbits=1.5) OR (Stopbits=2) THEN 250
220      BEEP
230      GOTO 200

Comment: Is parity to be enabled?

250   INPUT "Enable Parity? (YES/NO)",Response$
260      Parity_on=0
270      Parity=0
280      IF Response$[1;1]="Y" THEN 320
290      IF Response$[1;1]="N" THEN Chr_len
300      BEEP
310      GOTO 250
320      Parity_on=1

Comment: Enter parity type

340   INPUT "Is Parity even? (YES/NO)",Response$
350      IF Response$[1,1]="Y" THEN 410
360      IF Response$[1,1]="N" THEN 390
370      BEEP
380      GOTO 340
```



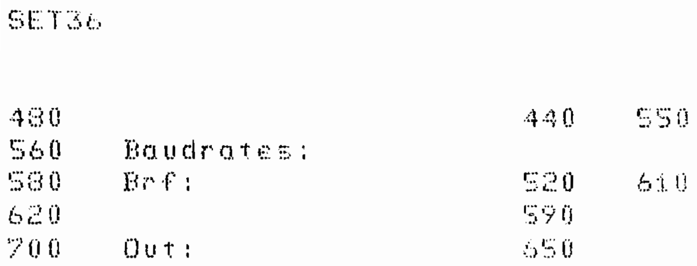
```
SET36


390      Parity=0
400      GOTO 430
410      Parity=1

Comment: Enter number of bits per character

430 Chr_len: INPUT "ENTER Number of bits per character (5, 6, 7 or
       8)",Char_length
440      IF (Char_length>4) AND (Char_length<9) AND (INT(Char_length)=
            Char_length) THEN 480
450      BEEP
460      GOTO 430

Comment: Enter baudrate; check for conflict with bitrate
480      INPUT "ENTER the baud rate set on 98036 card",Baudrate
490        RESTORE
500           FOR Rate=1 TO 10
510           READ Rate_value
520           IF Rate_value=Baudrate THEN Brf
530           NEXT Rate
540       BEEP
550       GOTO 480
560 Baudrates: DATA 75,110,150,300,600,1200,1800,2400,4800,9600

Comment: Enter bit rate factor

580 Brf:INPUT "ENTER Bit rate factor (1, 1/16, 1/64)",B_r_f
590      IF (B_r_f=1) OR (B_r_f=1/16) OR (B_r_f=1/64) THEN 620
600      BEEP
610      GOTO 580
620      IF B_r_f=1 THEN Bit_rate_fact=1
630      IF B_r_f=1/16 THEN Bit_rate_fact=2
640      IF B_r_f=1/64 THEN Bit_rate_fact=3
650      IF (Bit_rate_fact<>3) OR (Baudrate<4800) THEN Out
660        BEEP
670        DISP "WARNING: 1/64 bit rate not recommended at this baud
              rate!"
680        WAIT 4000

Comment: Set up 98036 I/O interface

700 Out:WRITE IO Selectcode,5;1         ! Comment:  R5 <- 00000001 (
       sets Control/Status Mode)
710     WRITE IO Selectcode,4;64        ! Comment: R4D <- 01000000 (
           reset)
720     WRITE IO Selectcode,4;(Stopbits*2-1)*64+Parity*32+Parity_on*16+
           (Char_length-5)*4+Bit_rate_fact
730     WRITE IO Selectcode,4;39        ! Comment: R4D <- 00100111 (
           set control word)
740     WRITE IO Selectcode,5;0         ! Comment: R5  <- 00000000 (
```

SET36

```
         set data mode)
750    READ IO Selectcode,4;Scratch     ! Comment: Cock for interrupt
760  SUBEND
```

# SET36

```
SET36

MAIN PROGRAM

COMMON VARIABLES:

VARIABLES:

USER DEFINED FUNCTIONS:

SUB PROGRAMS:

JUMP TARGETS:


SUB PROGRAM Set_up_98036

COMMON VARIABLES:

VARIABLES:
B_r_f                          580   590   620   630   640
Baudrate                       480   520   650
Bit_rate_fact                  620   630   640   650   720
Char_length                    430   440   720
Parity                         270   390   410   720
Parity_on                      260   320   720
Rate                           500   530
Rate_value                     510   520
Response$                      250   280   290   340   350   360
Scratch                        750
Selectcode                      40    50    80   130   700   710   720   730
                                     740   750
Status                         130   140
Stopbits                       200   210   720

USER DEFINED FUNCTIONS:

SUB PROGRAMS:

JUMP TARGETS:
40                              70   120   130
80                              50
130    Stat:                    80
150    Error:
200                            140   230
250                            210   310
320                            280
340                            380
390                            360
410                            350
430    Chr_len:                290   400   460
```



```
SET36


480                                440    550
560    Baudrates:
580    Brf:                        520    610
620                                590
700    Out:                        650
```

# 98036A Setup-Parameter Version

## Object of Subprogram

This subprogram allows you to set up a 98036A I/O card. It takes as parameters the select code of the 98036A card, the number of stop bits, parity enabled flag, parity type value (odd or even) the number of bits per character and the bit rate factor.

When the subprogram is called with the proper parameters, a series of I/O commands are used to address the proper I/O register and set the factors specified. Any parameters used that do not conform to the parameters required by the 98036A will be flagged so that you can correct the calling statement.

This subprogram checks for a usable interface and for the existence of a 98036A card.

For more detailed information concerning the setup of the 98036A card see the HP System 45B I/O ROM Programming Manual or the 98036A manual.

This version of the subprogram uses a set of parameters specified in the calling statement to set up the 98036A. The PROMPTER version of this subprogram will explicitly ask you what the desired value for each required parameter. As each parameter is entered it is checked for validity. See 98036A SETUP–Prompter Version for more information.

## User Instructions

The 98036A SETUP Subprogram–Parameter version is in DATA file "Set36"

The I/O ROM is necessary to use this subprogram, as is a 98036A card.

This is a subprogram and must be attached to your main program. See the Introduction for instructions for linking files.

### Calling Syntax

CALL Set_up_98036(Selectcode, Stopbits, Parity, Parity_on, Char_length, B_r_f)

### Input parameters

1) Selectcode: The select code of the 98036A (e.g. 7)

2) Stopbits: The number of stop bits (1,1.5 or 2)

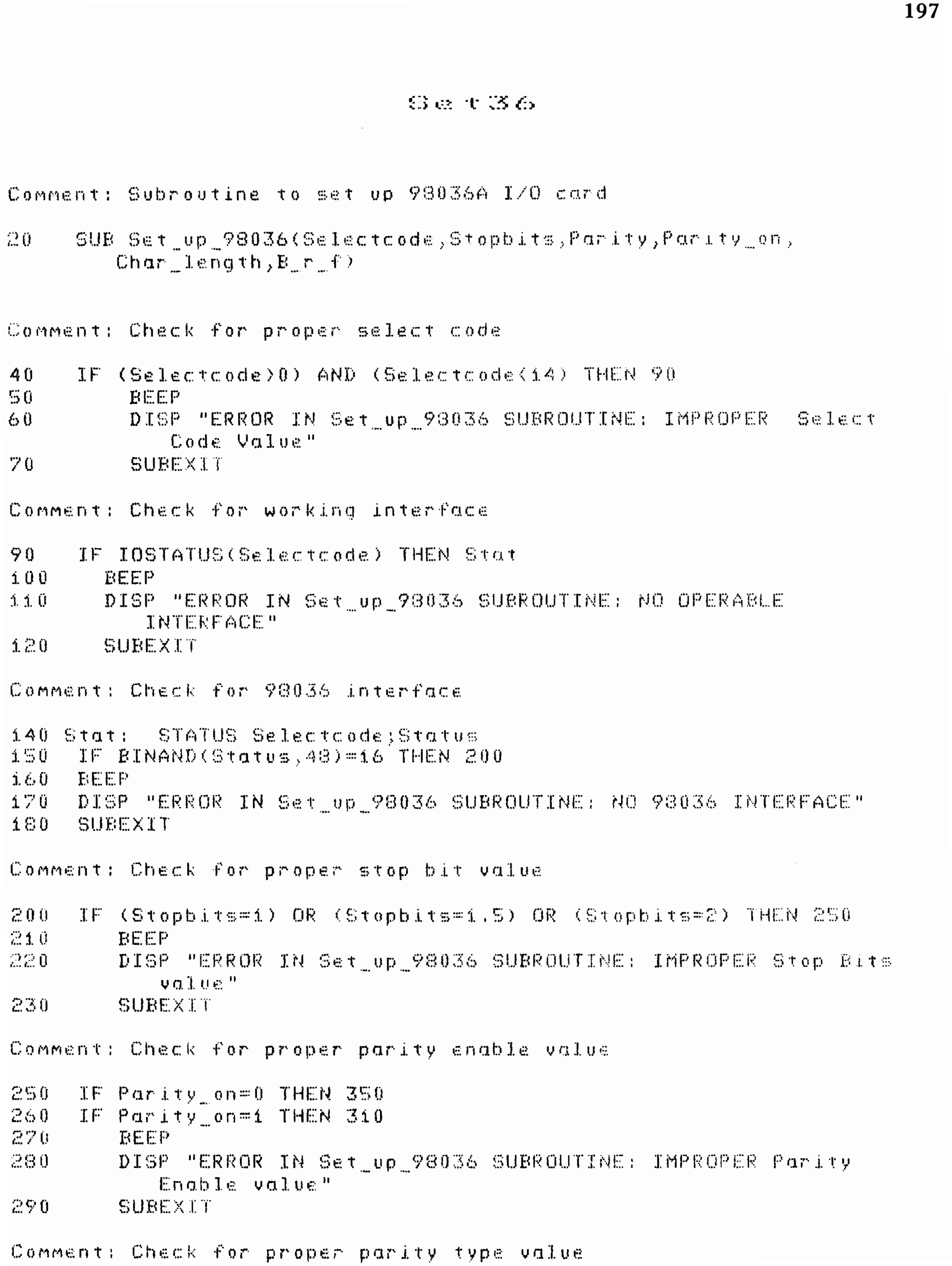
3) Parity: 1 for even parity; 0 for odd parity

4) Parity_on: 1 for parity enabled; 0 for parity NOT enabled

5) Char_length: The number of bits per character (5,6,7 or 8)

7) B_r_f: The bit rate factor (1,1/16,1/64)

When the subprogram is called with the proper parameters, the proper series of I/O commands sets up the card as specified and resets the interface for interrupts.



Set36

Comment: Subroutine to set up 98036A I/O card

```
20    SUB Set_up_98036(Selectcode,Stopbits,Parity,Parity_on,
         Char_length,B_r_f)
```

Comment: Check for proper select code

```
40    IF (Selectcode>0) AND (Selectcode<14) THEN 90
50        BEEP
60        DISP "ERROR IN Set_up_98036 SUBROUTINE: IMPROPER  Select
             Code Value"
70        SUBEXIT
```

Comment: Check for working interface

```
90    IF IOSTATUS(Selectcode) THEN Stat
100     BEEP
110     DISP "ERROR IN Set_up_98036 SUBROUTINE: NO OPERABLE
           INTERFACE"
120     SUBEXIT
```

Comment: Check for 98036 interface

```
140 Stat:  STATUS Selectcode;Status
150   IF BINAND(Status,48)=16 THEN 200
160   BEEP
170   DISP "ERROR IN Set_up_98036 SUBROUTINE: NO 98036 INTERFACE"
180   SUBEXIT
```

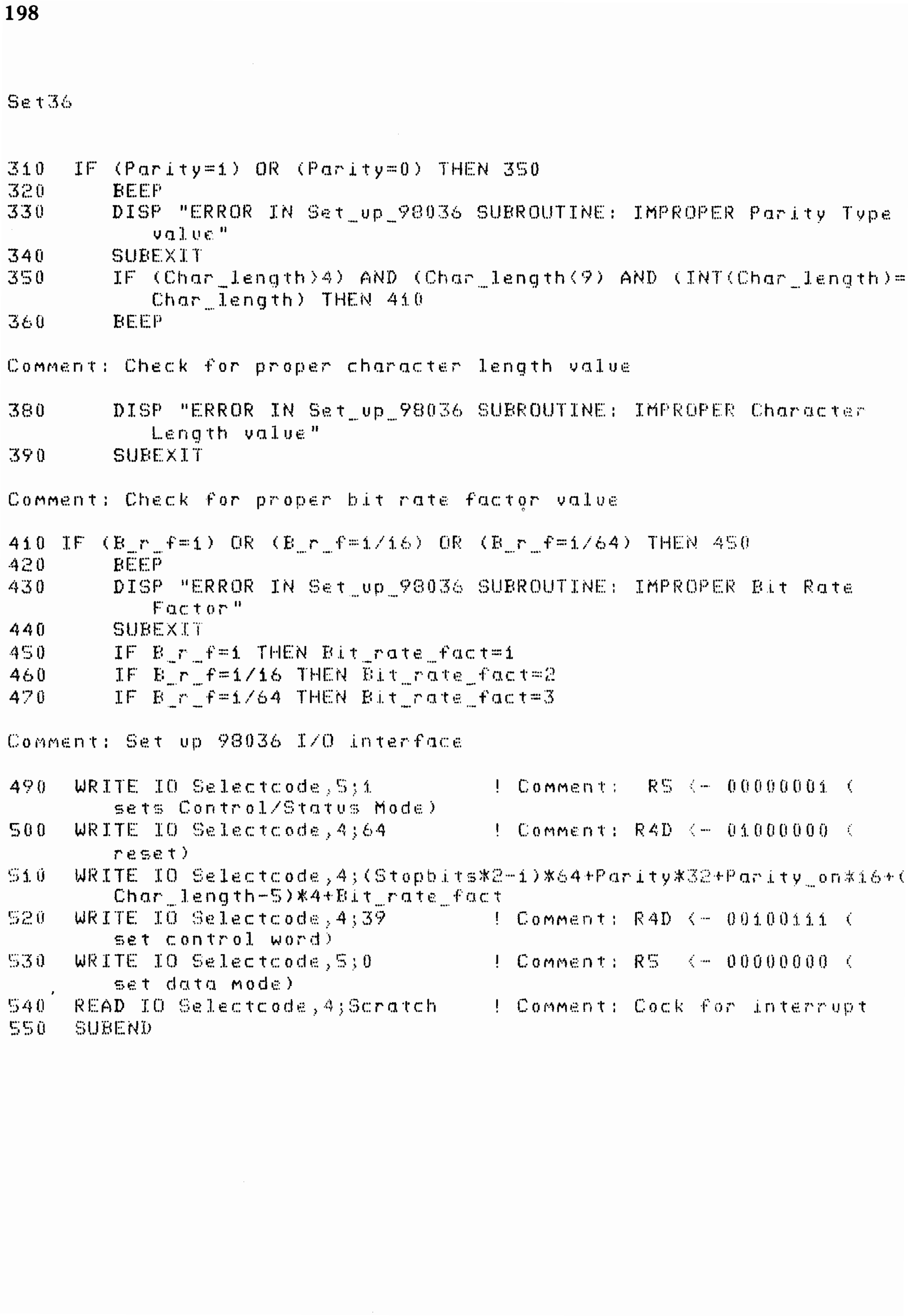
Comment: Check for proper stop bit value

```
200   IF (Stopbits=1) OR (Stopbits=1.5) OR (Stopbits=2) THEN 250
210      BEEP
220      DISP "ERROR IN Set_up_98036 SUBROUTINE: IMPROPER Stop Bits
            value"
230      SUBEXIT
```

Comment: Check for proper parity enable value

```
250   IF Parity_on=0 THEN 350
260   IF Parity_on=1 THEN 310
270      BEEP
280      DISP "ERROR IN Set_up_98036 SUBROUTINE: IMPROPER Parity
            Enable value"
290      SUBEXIT
```

Comment: Check for proper parity type value



```
Set36


310  IF (Parity=1) OR (Parity=0) THEN 350
320      BEEP
330      DISP "ERROR IN Set_up_98036 SUBROUTINE: IMPROPER Parity Type
            value"
340      SUBEXIT
350      IF (Char_length>4) AND (Char_length<9) AND (INT(Char_length)=
            Char_length) THEN 410
360      BEEP

Comment: Check for proper character length value

380      DISP "ERROR IN Set_up_98036 SUBROUTINE: IMPROPER Character
            Length value"
390      SUBEXIT

Comment: Check for proper bit rate factor value

410 IF (B_r_f=1) OR (B_r_f=1/16) OR (B_r_f=1/64) THEN 450
420      BEEP
430      DISP "ERROR IN Set_up_98036 SUBROUTINE: IMPROPER Bit Rate
            Factor"
440      SUBEXIT
450      IF B_r_f=1 THEN Bit_rate_fact=1
460      IF B_r_f=1/16 THEN Bit_rate_fact=2
470      IF B_r_f=1/64 THEN Bit_rate_fact=3

Comment: Set up 98036 I/O interface

490  WRITE IO Selectcode,5;1          ! Comment:  R5 <- 00000001 (
         sets Control/Status Mode)
500  WRITE IO Selectcode,4;64         ! Comment: R4D <- 01000000 (
         reset)
510  WRITE IO Selectcode,4;(Stopbits*2-1)*64+Parity*32+Parity_on*16+(
         Char_length-5)*4+Bit_rate_fact
520  WRITE IO Selectcode,4;39         ! Comment: R4D <- 00100111 (
         set control word)
530  WRITE IO Selectcode,5;0          ! Comment: R5  <- 00000000 (
         set data mode)
540  READ IO Selectcode,4;Scratch     ! Comment: Cock for interrupt
550  SUBEND
```

# Set36

```
Set36

MAIN PROGRAM

COMMON VARIABLES:

VARIABLES:

USER DEFINED FUNCTIONS:

SUB PROGRAMS:

JUMP TARGETS:


SUB PROGRAM Set_up_98036

COMMON VARIABLES:

VARIABLES:
B_r_f                       20    410    450    460    470
Bit_rate_fact              450    460    470    510
Char_length                 20    350    510
Parity                      20    310    510
Parity_on                   20    250    260    510
Scratch                    540
Selectcode                  20     40     90    140    490    500    510    520
                                  530    540
Status                     140    150
Stopbits                    20    200    510

USER DEFINED FUNCTIONS:

SUB PROGRAMS:

JUMP TARGETS:
90                          40
140    Stat:                90
200                        150
250                        200
310                        260
350                        250    310
410                        350
450                        410
```

# Read Status Of I/O Cards

(98032A, 98033A, 98034A, 98035A, 98036A)

## Object of Subprograms

These five subprograms interrogate the interface specified by the select code passed as a parameter. They then print the status of the card in the form of a table with the meaning of the individual bits printed above the value of the bit. Each subprogram can only be used with one type of interface since the meaning of the individual bits varies from interface to interface.

Each of these interfaces (with the exception of the 98034A) has a status word of 8 bits. Bits 4 and 5 (from the right) are the interface identifier bits. The status will be displayed as follows:

**98032A:**

| Priphrl | Interr. | Dir.Mem. | (Interface ID.) | | Inv.In | Inv.Out | Status | Status |
|---|---|---|---|---|---|---|---|---|
| status | enable | Acc.En. | (one | zero | data | data | bit 1 | bit 0 |
| x | x | x | 1 | 0 | x | x | x | x |

**98033A:**

| Priphrl | Interr. | Zero | (Interface ID.) | | Zero | Zero | Zero | Zero |
|---|---|---|---|---|---|---|---|---|
| status | enable | (unused) | (one | zero) | (unused) | (unused) | (unused) | (unused) |
| x | x | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

**98035A:**

| Priphrl | Interr. | Zero | (Interface ID.) | | Zero | Zero | Interrupt | Error |
|---|---|---|---|---|---|---|---|---|
| status | enable | (unused) | (one | zero) | (unused) | (unused) | flag | flag |
| x | x | 0 | 1 | 0 | 0 | 0 | x | x |

**98036A:**

| Priphrl | Interr. | Zero | (Interface ID.) | | Zero | Zero | Receiver | Transmtr |
|---|---|---|---|---|---|---|---|---|
| status | enable | (unused) | (zero | one) | (unused) | (unused) | Intrrpt | Intrrpt |
| x | x | 0 | 0 | 1 | 0 | 0 | x | x |

(Where "x" is either zero or one depending on the status bit.)

The 98034A interface is somewhat different, in that it will return four status words. The display will be similar to that produced by the other subprograms but will be in four lines as follows:

**98034A:**

| Priphrl | Service | Contrller | talker | Listener | System | One | Serial | End of |
|---|---|---|---|---|---|---|---|---|
| status | req True | active | active | active | Ctrlr set | (unused) | Poll set | Record |
| x | x | x | x | x | x | 1 | x | x |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Zero | Zero | Zero | (Interface ID) | | Device | Zero | ERROR |
| (unused) | (unused) | (unused) | (zero | zero) | clear | (unused) | Detected |
| 0 | 0 | 0 | 0 | 0 | x | 0 | x |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| One | One | Zero | (<– – – HP–IB ADDRESS – – –>) | | | | |
| (unused) | (unused) | (unused) | (Most signfcnt bit ----> Least signfcnt bit) | | | | |
| 1 | 1 | 0 | x | x | x | x | x |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| End or Identify | Remote enable | Service request | Atten–tion | Intrfce clear | Not Ready for data | Not Data accepted | Data valid |
| x | x | x | x | x | x | x | x |

When the status has been displayed the subprogram will return to the calling program.

For more information concerning the cards and their status see the I/O ROM Programming Manual or the manuals for the individual cards.

## User Instructions

The READ STATUS OF 98032A subprogram is in DATA file "Stat32"

The READ STATUS OF 98033A subprogram is in DATA file "Stat33"

The READ STATUS OF 98034A subprogram is in DATA file "Stat34"

The READ STATUS OF 98035A subprogram is in DATA file "Stat35"

The READ STATUS OF 98036A subprogram is in DATA file "Stat36"

The I/O ROM is necessary to use these subprograms.

The appropriate I/O card is necessary to use each subprogram.

These are subprograms and must be attached to your main program. See the Introduction for instructions for linking files.

### Calling Syntax

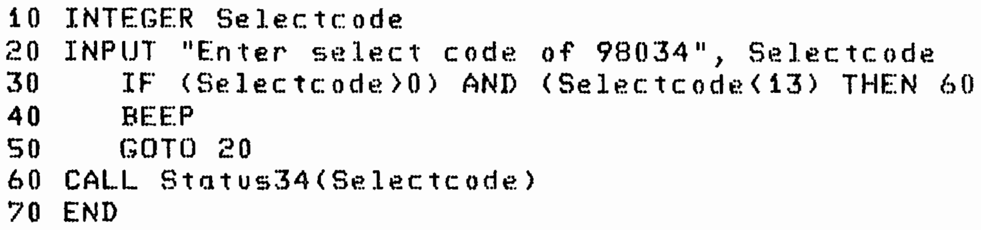
```
CALL Status32(Selectcode)
CALL Status33(Selectcode)
CALL Status34(Selectcode)
CALL Status35(Selectcode)
CALL Status36(Selectcode)
```

### Input parameters

1) Selectcode:an INTEGER value or variable which is the select code of the I/O card to be interrogated

Control will be returned to your main program when the status has been displayed on the CRT.

A simple driver main program can be constructed to use these subprograms. Here is one for the 98034A:



```
10 INTEGER Selectcode
20 INPUT "Enter select code of 98034", Selectcode
30    IF (Selectcode>0) AND (Selectcode<13) THEN 60
40    BEEP
50    GOTO 20
60 CALL Status34(Selectcode)
70 END
```

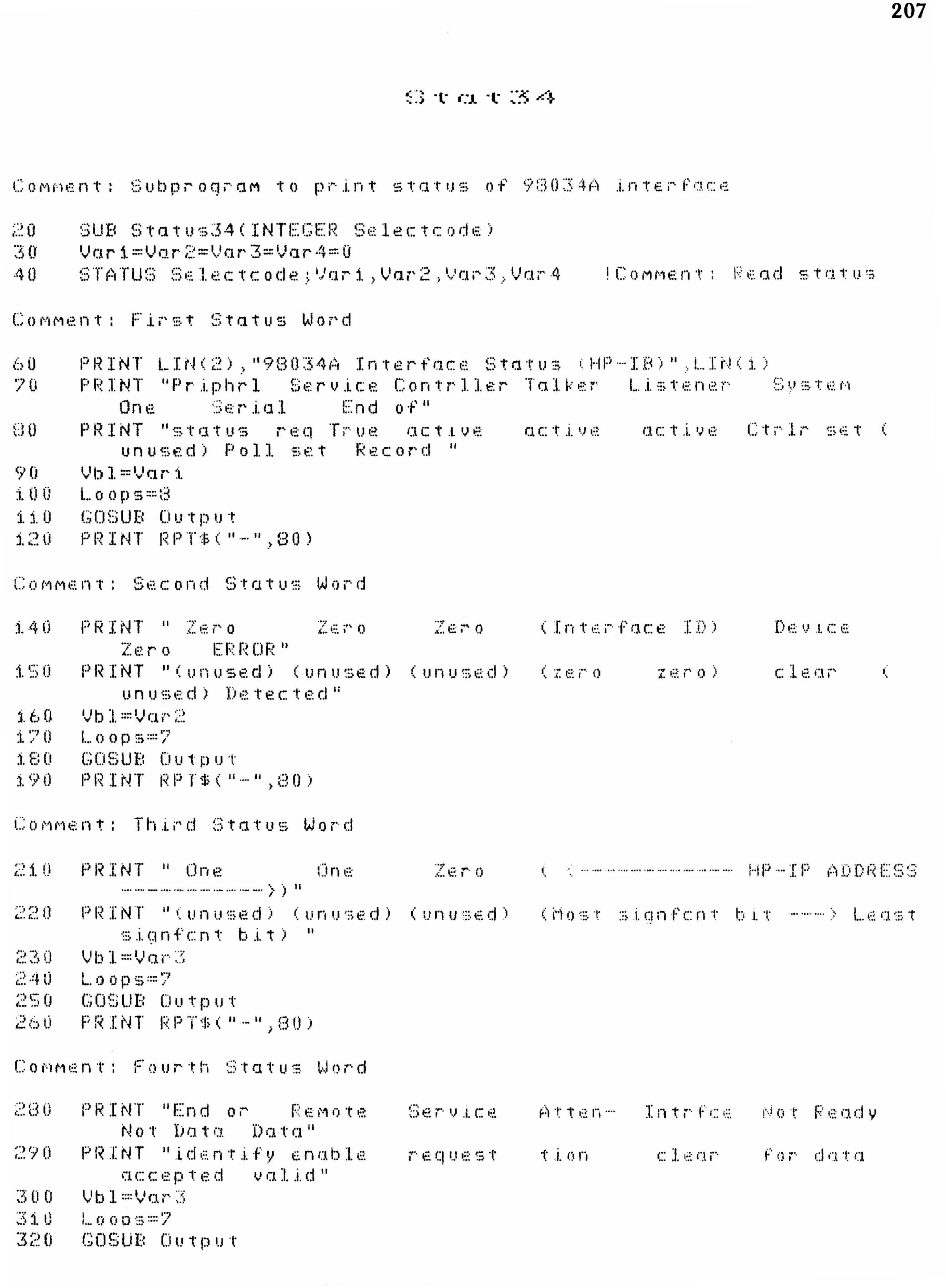
The status will be printed on the current printer select code. To redirect the output, merely execute the PRINTER IS statement (either from the program or from the keyboard) for the desired printer prior to issuing the CALL command.

# Stat32

```
Comment: Subroutine to print status of 98032A interface
20   SUB Status32(INTEGER Selectcode)
30   STATUS Selectcode;Vari
40   PRINT LIN(2),"           98032A Interface Status",LIN(1)
50   PRINT "Priphrl  Interr.  Dir.Mem.  (Interface ID.)  Inv.In
        Inv.Out   Status  Status"
60   PRINT "status   enable   Acc. En.  (one       zero)   data
        data       bit 1   bit 0"
70 Image:   IMAGE #,3X,D,5X
80       FOR I=8 TO 0 STEP -1
90       Bit=BIT(Vari,I)
100      PRINT USING Image;Bit
110      NEXT I
120  PRINT
130  PRINT LIN(1),RPT$("-",80)
140  SUBEND
```

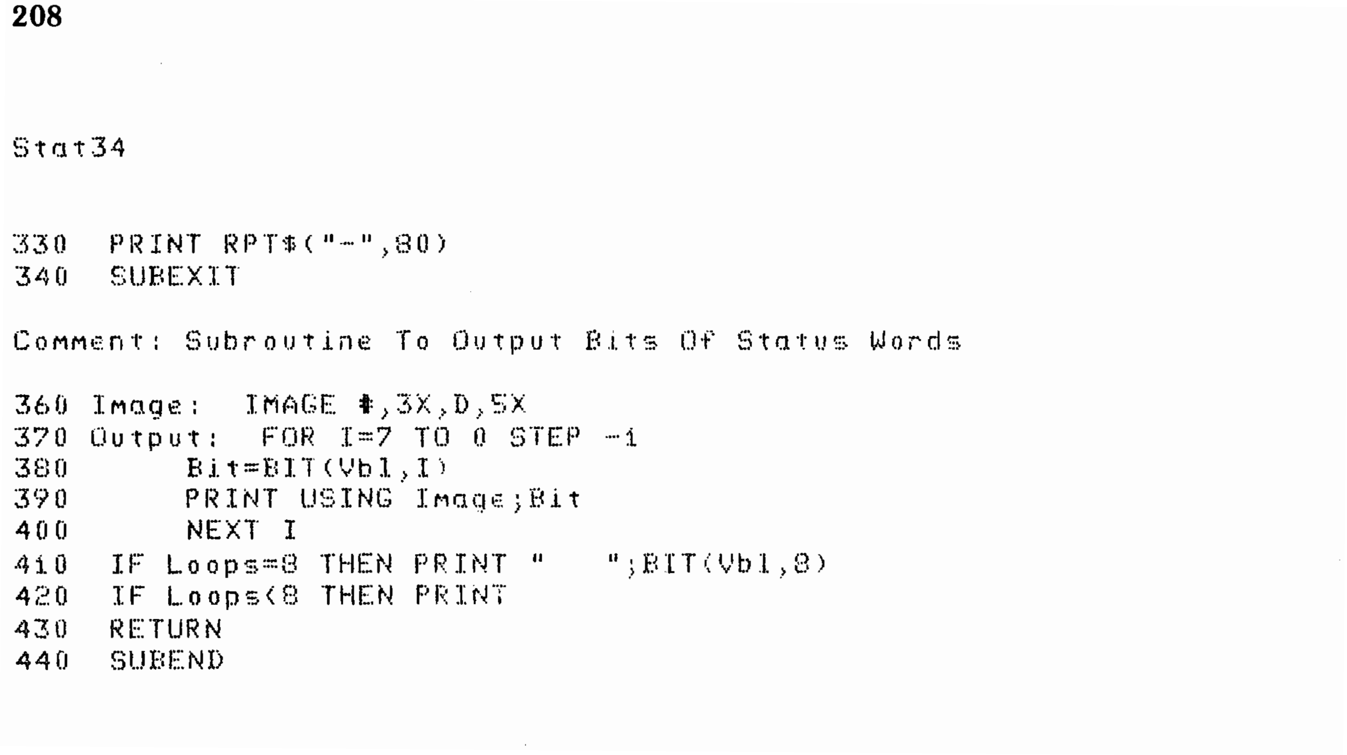
# Stat33

```
Comment: Subroutine to print status of 98032A interface
20   SUB Status33(INTEGER Selectcode)
30   STATUS Selectcode;Vari
40   PRINT LIN(2),"          98033A Interface Status",LIN(1)
50   PRINT "Priphrl  Interr.    Zero    (Interface ID.)   Zero
        Zero    Zero     Zero "
60   PRINT "status   enable   (unused)  (one       zero)  (unused) (
        unused) (unused) (unused)"
70 Image:   IMAGE #,3X,D,5X
80        FOR I=8 TO 0 STEP -1
90        Bit=BIT(Vari,I)
100       PRINT USING Image;Bit
110       NEXT I
120  PRINT
130  PRINT LIN(1),RPT$("-",80)
140  SUBEND
```



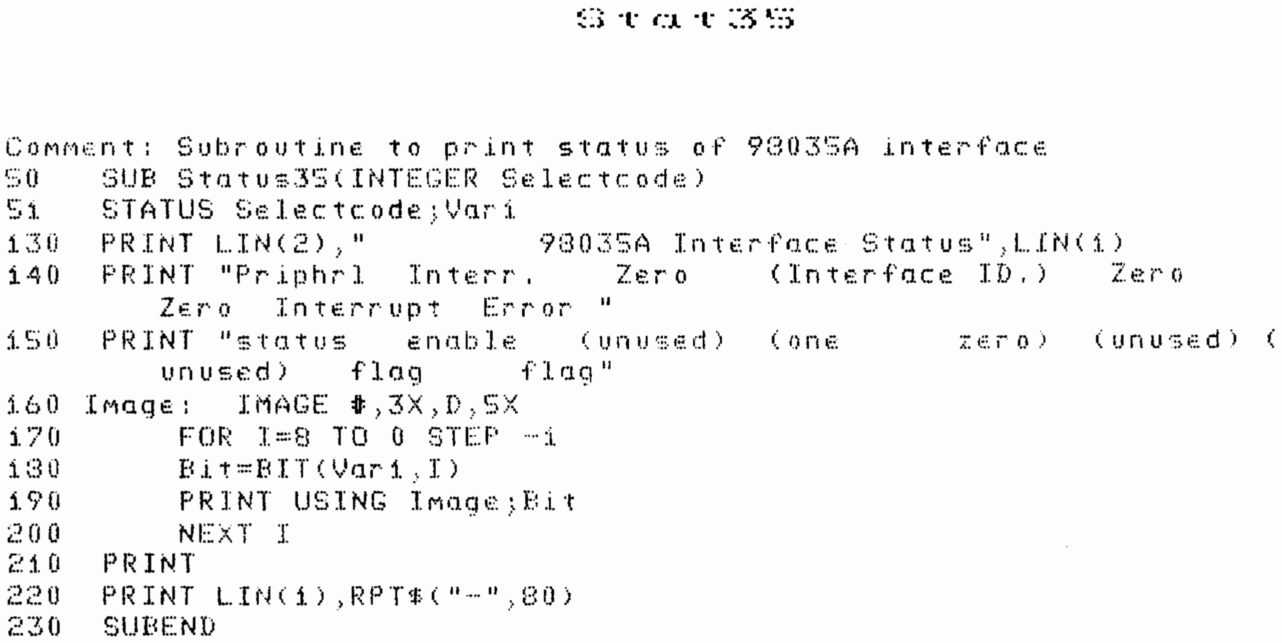
# Stat34

```
Comment: Subprogram to print status of 98034A interface

20    SUB Status34(INTEGER Selectcode)
30    Var1=Var2=Var3=Var4=0
40    STATUS Selectcode;Var1,Var2,Var3,Var4   !Comment: Read status

Comment: First Status Word

60    PRINT LIN(2),"98034A Interface Status (HP-IB)",LIN(1)
70    PRINT "Priphrl  Service Contrller Talker  Listener   System
         One     Serial    End of"
80    PRINT "status  req True  active   active   active  Ctrlr set (
         unused) Poll set  Record "
90    Vbl=Var1
100   Loops=8
110   GOSUB Output
120   PRINT RPT$("-",80)

Comment: Second Status Word

140   PRINT " Zero       Zero      Zero     (Interface ID)     Device
         Zero    ERROR"
150   PRINT "(unused) (unused) (unused)  (zero     zero)    clear    (
         unused) Detected"
160   Vbl=Var2
170   Loops=7
180   GOSUB Output
190   PRINT RPT$("-",80)

Comment: Third Status Word

210   PRINT " One        One       Zero     ( (-------------- HP-IP ADDRESS
         --------------))"
220   PRINT "(unused) (unused) (unused)  (Most signfcnt bit ---) Least
         signfcnt bit) "
230   Vbl=Var3
240   Loops=7
250   GOSUB Output
260   PRINT RPT$("-",80)

Comment: Fourth Status Word

280   PRINT "End or   Remote    Service    Atten-   Intrfce   Not Ready
         Not Data  Data"
290   PRINT "identify enable   request    tion      clear    for data
         accepted  valid"
300   Vbl=Var3
310   Loops=7
320   GOSUB Output
```



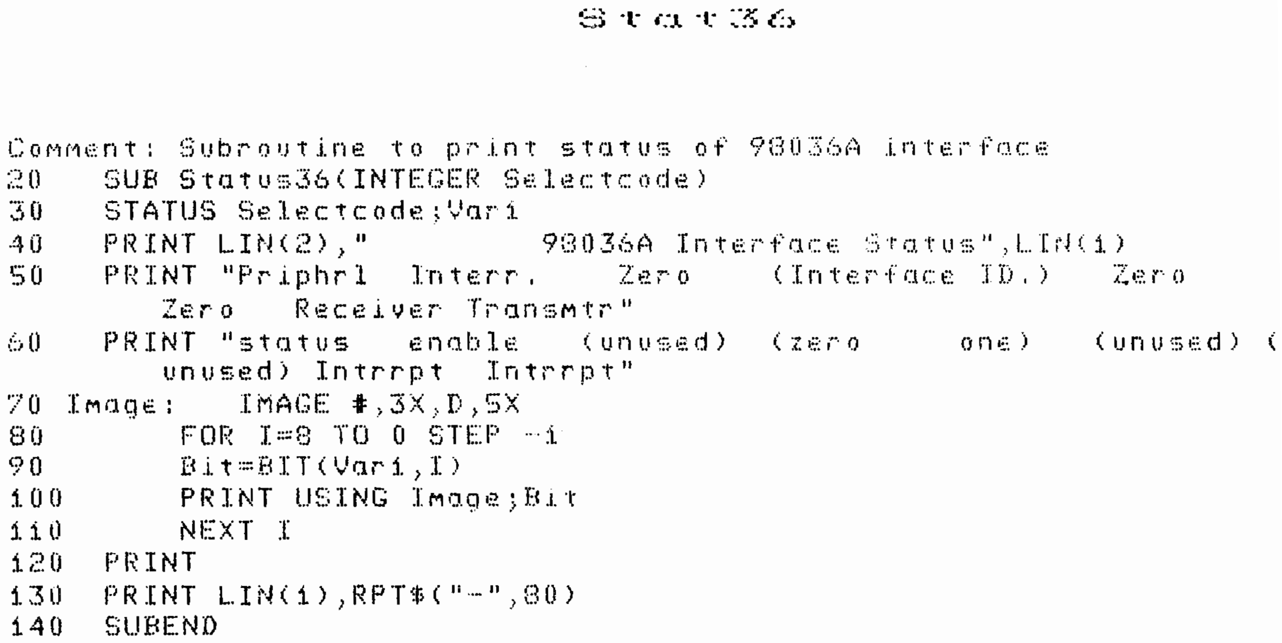
Stat34

```
330  PRINT RPT$("-",80)
340  SUBEXIT

Comment: Subroutine To Output Bits Of Status Words

360 Image:  IMAGE #,3X,D,5X
370 Output:  FOR I=7 TO 0 STEP -1
380      Bit=BIT(Vbl,I)
390      PRINT USING Image;Bit
400      NEXT I
410  IF Loops=8 THEN PRINT "    ";BIT(Vbl,8)
420  IF Loops<8 THEN PRINT
430  RETURN
440  SUBEND
```



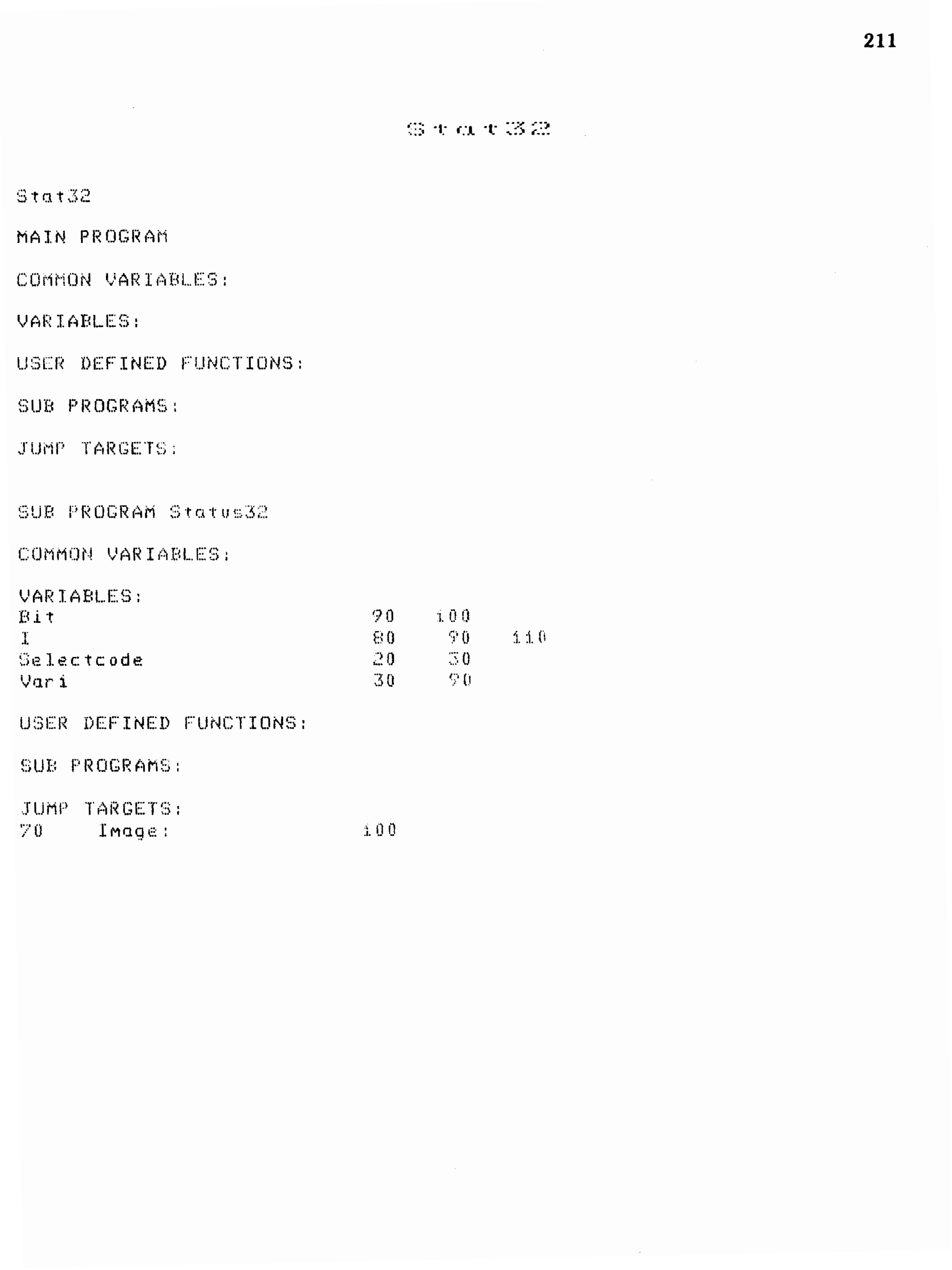
## Stat35

```
Comment: Subroutine to print status of 98035A interface
50   SUB Status35(INTEGER Selectcode)
51   STATUS Selectcode;Var1
130  PRINT LIN(2),"          98035A Interface Status",LIN(1)
140  PRINT "Priphrl  Interr.     Zero     (Interface ID.)    Zero
        Zero  Interrupt  Error  "
150  PRINT "status   enable   (unused)  (one       zero)  (unused) (
        unused)   flag      flag"
160 Image:  IMAGE #,3X,D,5X
170      FOR I=8 TO 0 STEP -1
180      Bit=BIT(Var1,I)
190      PRINT USING Image;Bit
200      NEXT I
210  PRINT
220  PRINT LIN(1),RPT$("-",80)
230  SUBEND
```

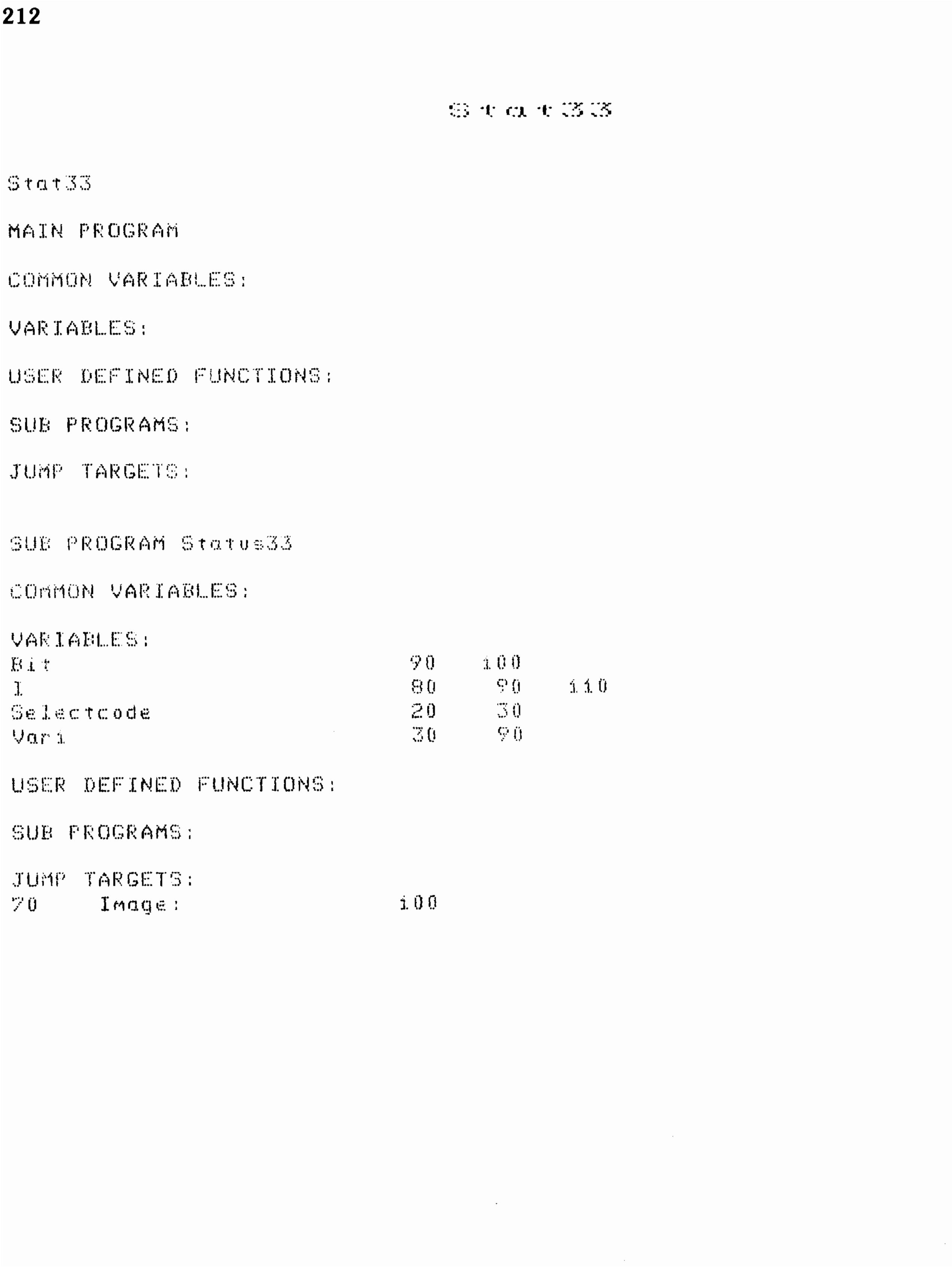


Stat36

```
Comment: Subroutine to print status of 98036A interface
20   SUB Status36(INTEGER Selectcode)
30   STATUS Selectcode;Vari
40   PRINT LIN(2),"          98036A Interface Status",LIN(1)
50   PRINT "Priphrl  Interr.    Zero    (Interface ID.)   Zero
        Zero   Receiver Transmtr"
60   PRINT "status   enable   (unused)  (zero      one)   (unused) (
        unused) Intrrpt  Intrrpt"
70 Image:   IMAGE #,3X,D,5X
80       FOR I=8 TO 0 STEP -1
90       Bit=BIT(Vari,I)
100      PRINT USING Image;Bit
110      NEXT I
120  PRINT
130  PRINT LIN(1),RPT$("-",80)
140  SUBEND
```

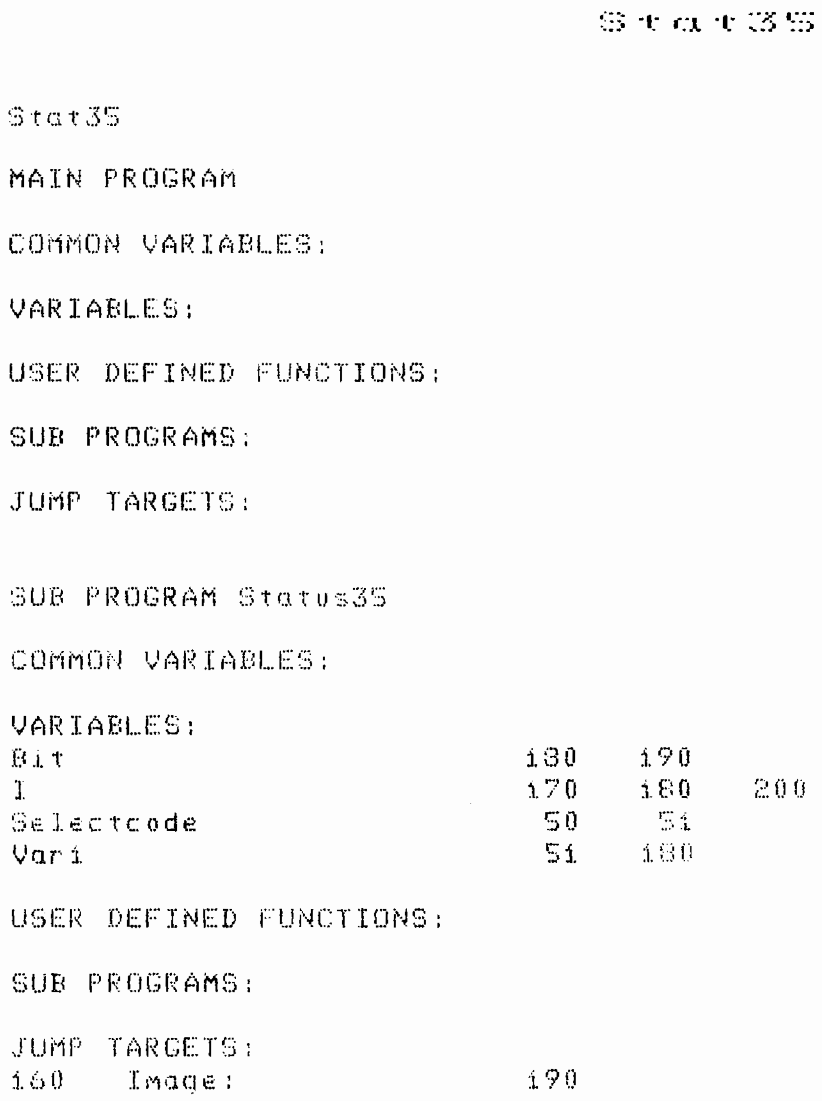


```
                              Stat32

Stat32

MAIN PROGRAM

COMMON VARIABLES:

VARIABLES:

USER DEFINED FUNCTIONS:

SUB PROGRAMS:

JUMP TARGETS:


SUB PROGRAM Status32

COMMON VARIABLES:

VARIABLES:
Bit                           90    100
I                             80     90    110
Selectcode                    20     30
Vari                          30     90

USER DEFINED FUNCTIONS:

SUB PROGRAMS:

JUMP TARGETS:
70     Image:                100
```

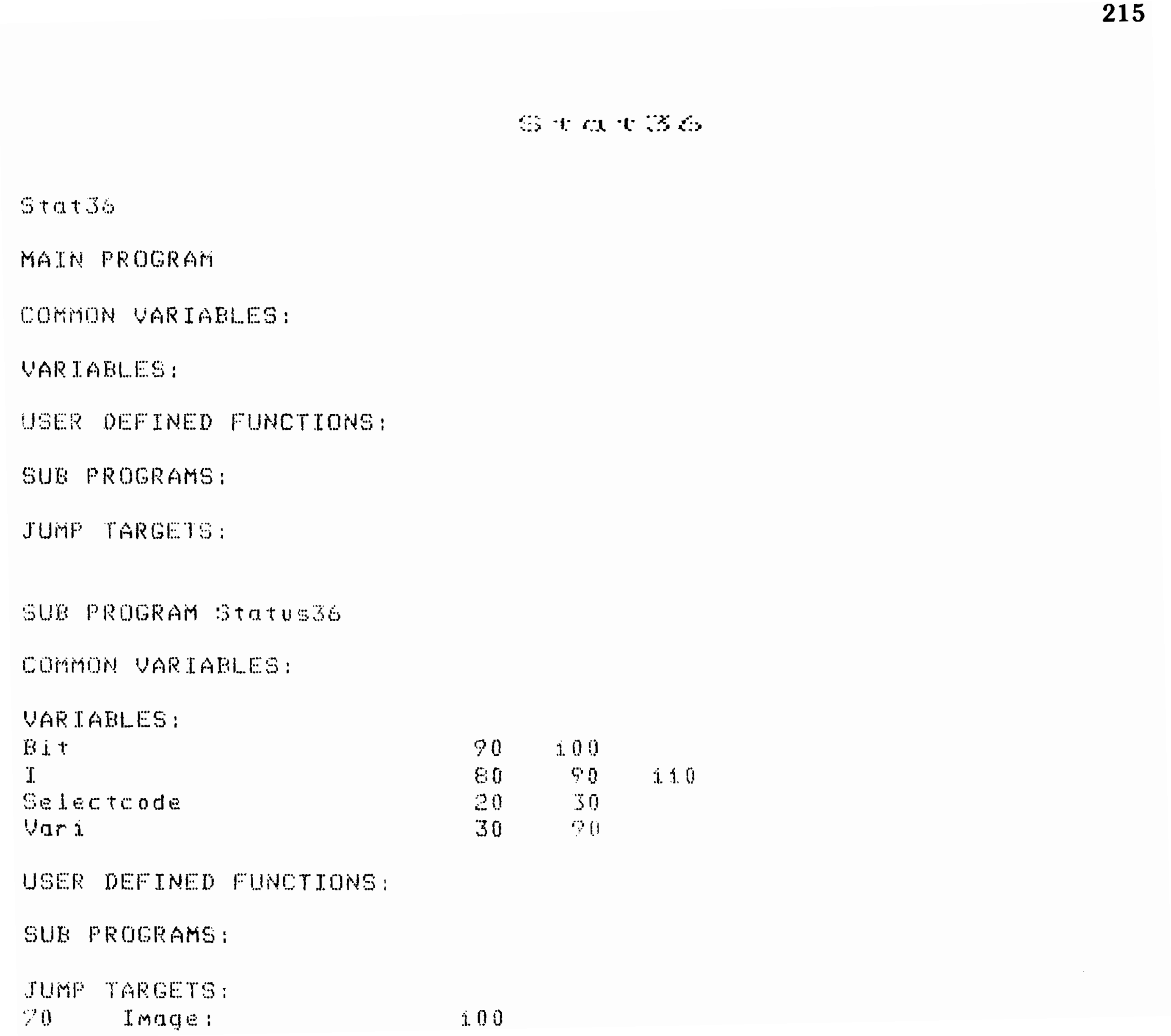


```
                              Stat33

Stat33

MAIN PROGRAM

COMMON VARIABLES:

VARIABLES:

USER DEFINED FUNCTIONS:

SUB PROGRAMS:

JUMP TARGETS:


SUB PROGRAM Status33

COMMON VARIABLES:

VARIABLES:
Bit                            90    100
I                              80     90    110
Selectcode                     20     30
Vari                           30     90

USER DEFINED FUNCTIONS:

SUB PROGRAMS:

JUMP TARGETS:
70     Image:                 100
```

```
                              Stat34

Stat34

MAIN PROGRAM

COMMON VARIABLES:

VARIABLES:

USER DEFINED FUNCTIONS:

SUB PROGRAMS:

JUMP TARGETS:


SUB PROGRAM Status34

COMMON VARIABLES:

VARIABLES:
Bit                      380    390
I                        370    380    400
Loops                    100    170    240    310    410    420
Selectcode                20     40
Var1                      30     40     90
Var2                      30     40    160
Var3                      30     40    230    300
Var4                      30     40
Vbl                       90    160    230    300    380    410

USER DEFINED FUNCTIONS:

SUB PROGRAMS:

JUMP TARGETS:
360   Image:             390
370   Output:            110    180    250    320
```



```
                                  Stat35

Stat35

MAIN PROGRAM

COMMON VARIABLES:

VARIABLES:

USER DEFINED FUNCTIONS:

SUB PROGRAMS:

JUMP TARGETS:


SUB PROGRAM Status35

COMMON VARIABLES:

VARIABLES:
Bit                             180    190
I                               170    180    200
Selectcode                       50     51
Vari                             51    180

USER DEFINED FUNCTIONS:

SUB PROGRAMS:

JUMP TARGETS:
160    Image:                   190
```



# Stat36

```
Stat36

MAIN PROGRAM

COMMON VARIABLES:

VARIABLES:

USER DEFINED FUNCTIONS:

SUB PROGRAMS:

JUMP TARGETS:


SUB PROGRAM Status36

COMMON VARIABLES:

VARIABLES:
Bit                        90    100
I                          80     90    110
Selectcode                 20     30
Vari                       30     70

USER DEFINED FUNCTIONS:

SUB PROGRAMS:

JUMP TARGETS:
70     Image:             100
```

# Read Usart Status Word For 98036A

## Object of Subprogram

This subprogram reads USART status control word of a 98036A interface and prints out the individual bits of the word with the meaning of the bit printed above it on the CRT. The only parameter to this subprogram is the select code of the interface.

The USART status control word will be printed in the following format:

```
 Dataset     Zero      Framing     Overrun    Parity   Transmtr    Receiver    Transmtr
  ready    (unused)     error       error      error     empty       ready       ready
     x          0          x           x          x         x           x           x
```

(Where "x" may be either zero or one depending one the bit value.)

For further information on the USART control status word see the HP System 45B I/O ROM Programming Manual or the 98036A manual.

## User Instructions

The subprogram Stat_usart is in DATA file "USART"

The I/O ROM is necessary to use this subprogram.

This is a subprogram and must be attached to your main program. See the Introduction for instructions for linking files.

### Calling Syntax

CALL Stat_usart(Selectcode)

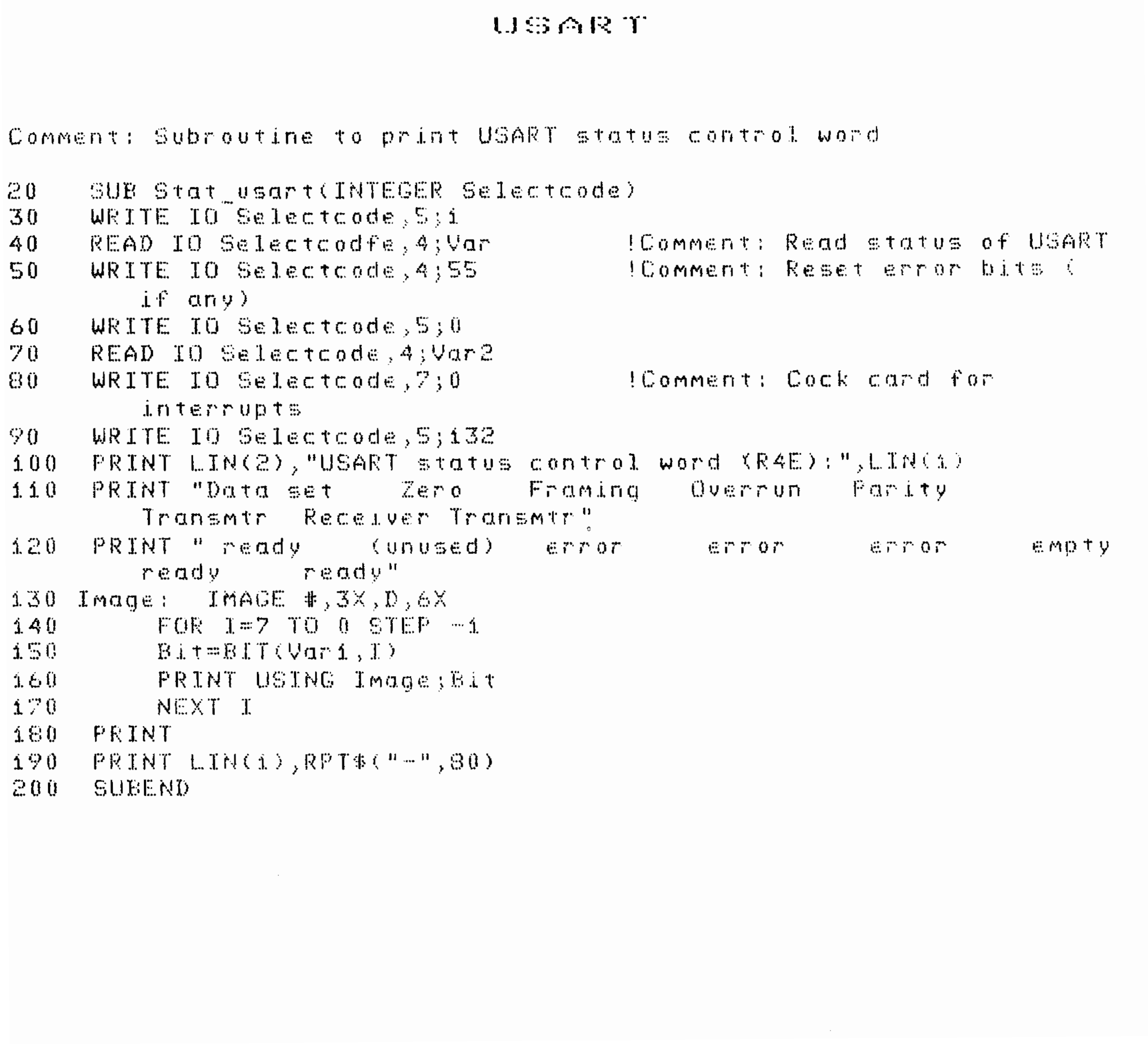
### Input parameters

1) Selectcode:an INTEGER value or variable which is the selectcode of the 98036A card being interrogated.

Control will be returned to the calling program when the status has been displayed on the CRT.

A simple driver for this subprogram can be constructed as follows:

```
10 INTEGER Selectcode
20 INPUT "Enter selectcode of 98036A", Selectcode
30      IF (Selectcode>0) AND (Selectcode<13) THEN 60
40      BEEP
50      GOTO 30
60 CALL Stat_usart(Selectcode)
70 END
```



# USART

```
Comment: Subroutine to print USART status control word

20    SUB Stat_usart(INTEGER Selectcode)
30    WRITE IO Selectcode,5;1
40    READ IO Selectcodfe,4;Var          !Comment: Read status of USART
50    WRITE IO Selectcode,4;55           !Comment: Reset error bits (
         if any)
60    WRITE IO Selectcode,5;0
70    READ IO Selectcode,4;Var2
80    WRITE IO Selectcode,7;0            !Comment: Cock card for
         interrupts
90    WRITE IO Selectcode,5;132
100   PRINT LIN(2),"USART status control word (R4E):",LIN(1)
110   PRINT "Data set     Zero     Framing    Overrun    Parity
         Transmtr  Receiver Transmtr"
120   PRINT " ready     (unused)   error      error      error      empty
         ready     ready"
130 Image:  IMAGE #,3X,D,6X
140       FOR I=7 TO 0 STEP -1
150       Bit=BIT(Var1,I)
160       PRINT USING Image;Bit
170       NEXT I
180   PRINT
190   PRINT LIN(1),RPT$("-",80)
200   SUBEND
```

# USART

```
USART

MAIN PROGRAM

COMMON VARIABLES:

VARIABLES:

USER DEFINED FUNCTIONS:

SUB PROGRAMS:

JUMP TARGETS:


SUB PROGRAM Stat_usart

COMMON VARIABLES:

VARIABLES:
Bit                          150    160
I                            140    150    170
Selectcode                    20     30     50     60     90
Selectcodfe                   40
Var                           40
Var1                         150
Var2                          70

USER DEFINED FUNCTIONS:

SUB PROGRAMS:

JUMP TARGETS:
130    Image:                160
```

# Set Time On 98035A Real Time Clock

## Object of Subprogram

This subprogram allows you to set the date and time of the 98035A real time clock. Each parameter, month, day, hour, minute and second, is entered individually using a clear and simple prompt. Should the clock be option 2 (day before month) the subprogram will take this into account. All parameters are checked for validity.

When all the parameters have been entered the subprogram will pause until you see that the exact time that was entered has arrived. You then push CONT and the real time clock will be set.

## User Instructions

The Set_time subprogram is in DATA file "SetTim".

The I/O ROM is necessary to use this subprogram, as is the 98035A.

This is a subprogram and must attached to your main program. See the Introduction for instructions for linking files.

### Calling Syntax

CALL Set_time

### Data to be entered

1) Option number(1 for American (month before day); 2 for European (day before month))
2) Month (1 through 12)
3) Day (1 through the number of days in the input month)
4) Hour (0 (Midnight) through 23 (11:00 PM))
5) Minute (0 through 59)
6) Second (0 through 59)

When the above information has been input the program will pause waiting for you to press CONT. This should be done at the precise time specified by the input parameters. The program will then send out the appropriate I/O messages to set the 98035A real time clock. The clock is assumed to be at select code 9.

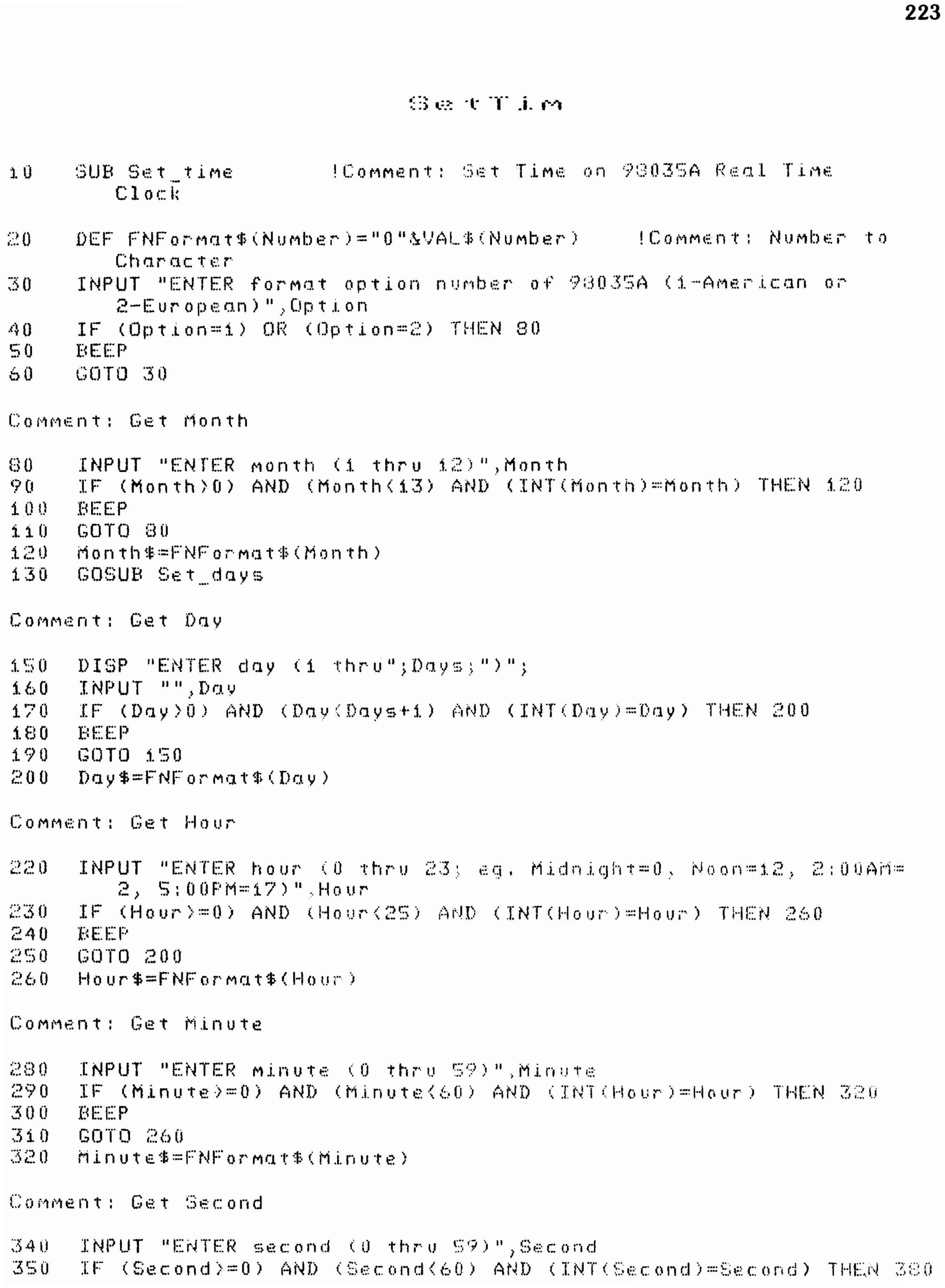
**NOTE**

Do not reset or remove power for 90 seconds after pressing CONT.

A simple driver for this subprogram can be written as follows:

```
10 CALL Set_time
20 END
```

In order to display the time from the 98035A real time clock see the Disp_time subprogram.



SetTim

```
10    SUB Set_time          !Comment: Set Time on 98035A Real Time
         Clock

20    DEF FNFormat$(Number)="0"&VAL$(Number)     !Comment: Number to
         Character
30    INPUT "ENTER format option number of 98035A (1-American or
         2-European)",Option
40    IF (Option=1) OR (Option=2) THEN 80
50    BEEP
60    GOTO 30

Comment: Get Month

80    INPUT "ENTER month (1 thru 12)",Month
90    IF (Month>0) AND (Month<13) AND (INT(Month)=Month) THEN 120
100   BEEP
110   GOTO 80
120   Month$=FNFormat$(Month)
130   GOSUB Set_days

Comment: Get Day

150   DISP "ENTER day (1 thru";Days;")";
160   INPUT "",Day
170   IF (Day>0) AND (Day<Days+1) AND (INT(Day)=Day) THEN 200
180   BEEP
190   GOTO 150
200   Day$=FNFormat$(Day)

Comment: Get Hour

220   INPUT "ENTER hour (0 thru 23; eg. Midnight=0, Noon=12, 2:00AM=
         2, 5:00PM=17)",Hour
230   IF (Hour>=0) AND (Hour<25) AND (INT(Hour)=Hour) THEN 260
240   BEEP
250   GOTO 200
260   Hour$=FNFormat$(Hour)

Comment: Get Minute

280   INPUT "ENTER minute (0 thru 59)",Minute
290   IF (Minute>=0) AND (Minute<60) AND (INT(Hour)=Hour) THEN 320
300   BEEP
310   GOTO 260
320   Minute$=FNFormat$(Minute)

Comment: Get Second

340   INPUT "ENTER second (0 thru 59)",Second
350   IF (Second>=0) AND (Second<60) AND (INT(Second)=Second) THEN 380
```

SetTim

```
360   BEEP
370   GOTO 320
380   Second$=FNFormat$(Second)

Comment: When ready set time

400   IF Option=1 THEN Time$=Month$[LEN(Month$)-1]&":"&Day$[LEN(Day$)-
         1]&":"&Hour$[LEN(Hour$)-1]&":"&Minute$[LEN(Minute$)-1]&":"&
         Second$[LEN(Second$)-1]
410   IF Option=2 THEN Time$=Day$[LEN(Day$)-1]&":"&Month$[LEN(Month$)-
         1]&":"&Hour$[LEN(Hour$)-1]&":"&Minute$[LEN(Minute$)-1]&":"&
         Second$[LEN(Second$)-1]
420   DISP "Time to be set: ";Time$;"       (PRESS CONT when ready to
         set time)"
430   PAUSE
440   OUTPUT 9;"hAlt units & Set";Time$     !Comment: time setting
         instruction
450   DISP "Setting initiated at ";Time$;" DO NOT RESET OR REMOVE
         POWER FOR 90 SEC."
460   SUBEXIT


480 Set_days:  ! Comment: Set the number of days in the month for
       checking
490     DIM D_m(12)        !Comment: Array to hold days in each month
500     RESTORE Days_per_month
510     MAT READ D_m
520     Days=D_m(Month)
530     RETURN
540 Days_per_month:    DATA 0,31,29,31,30,31,30,31,31,30,31,30,31
550   SUBEND
```

# SetTim

```
SetTim

MAIN PROGRAM

COMMON VARIABLES:

VARIABLES:

USER DEFINED FUNCTIONS:

SUB PROGRAMS:

JUMP TARGETS:


SUB PROGRAM Set_time

COMMON VARIABLES:

VARIABLES:
D_m(*)                     490   510   520
Day                        160   170   200
Day$                       200   400   410
Days                       150   170   520
Hour                       220   230   260   290
Hour$                      260   400   410
Minute                     280   290   320
Minute$                    320   400   410
Month                       80    90   120   520
Month$                     120   400   410
Number                      20
Option                      30    40   400   410
Second                     340   350   380
Second$                    380   400   410
Time$                      400   410   420   440   450

USER DEFINED FUNCTIONS:
FNFormat$                   20   120   200   260   320   380

SUB PROGRAMS:

JUMP TARGETS:
30                          60
80                          40   110
120                         90
150                        190
200                        170   250
260                        230   310
320                        290   370
380                        350
480   Set_days:            130
```

```
SetTim

540   Days_per_month:        500
```

# Display Time From 98035A Real Time Clock

## Object of Subprogram

This subprogram displays the time from the 98035A real time clock. It will display the month, name, the day and the time. For example, the date–time may be displayed as follows:

OCT 27          2:55:32 PM

## User Instructions

The subprogram Disp_time is in DATA file ''DspTim''

The I/O ROM is necessary to use this subroutine, as is a 98035A.

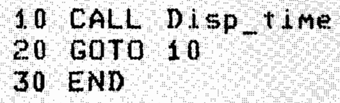
This is a subprogram and must be attached to your main program. See the Introduction for instructions for linking files.

### Calling Syntax

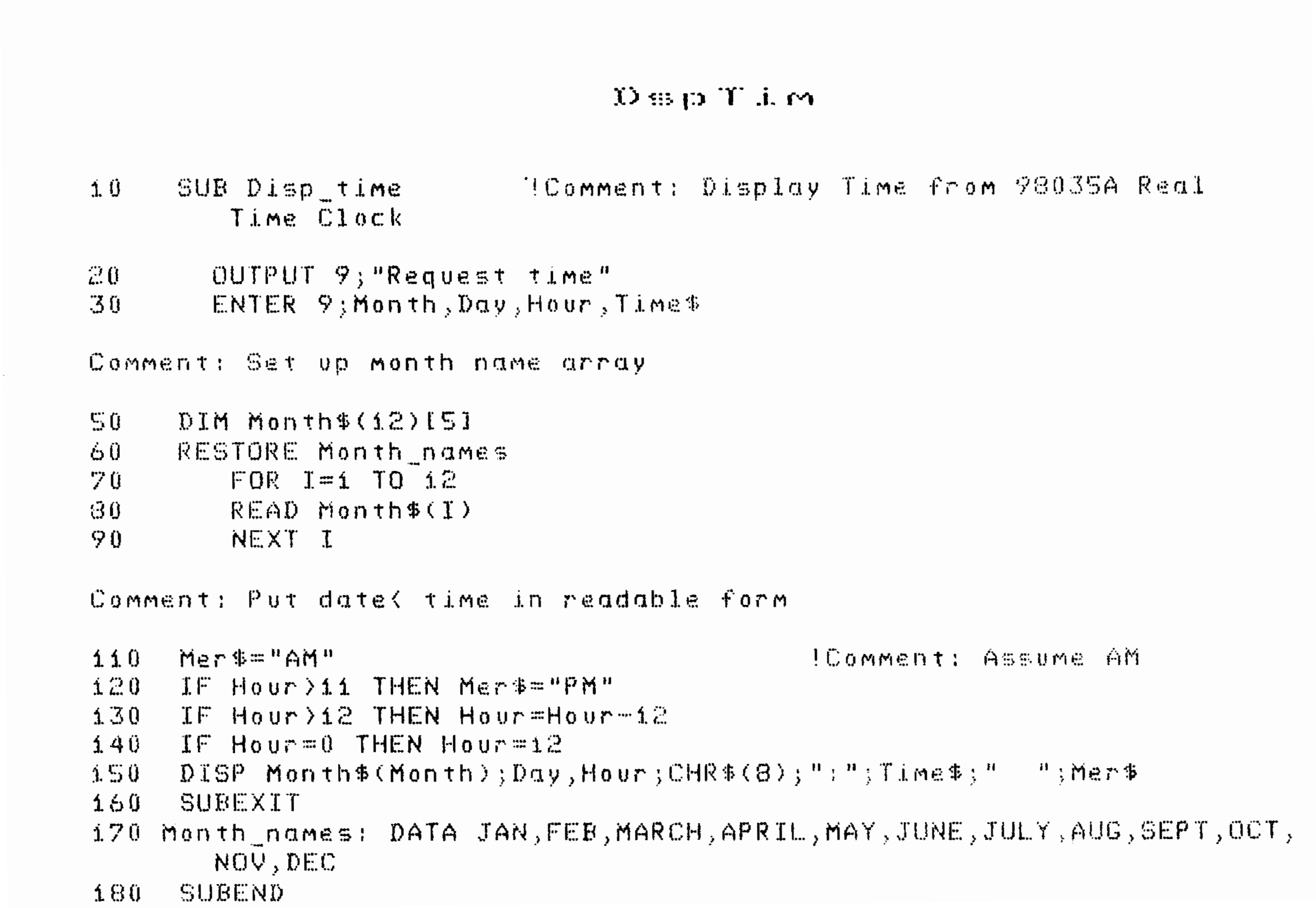
CALL Disp_time

When the subprogram is called the date–time will be displayed, in the display area of the CRT. Control will immediately be returned to the calling program.

A simple driver can be written to use this subprogram as follows:

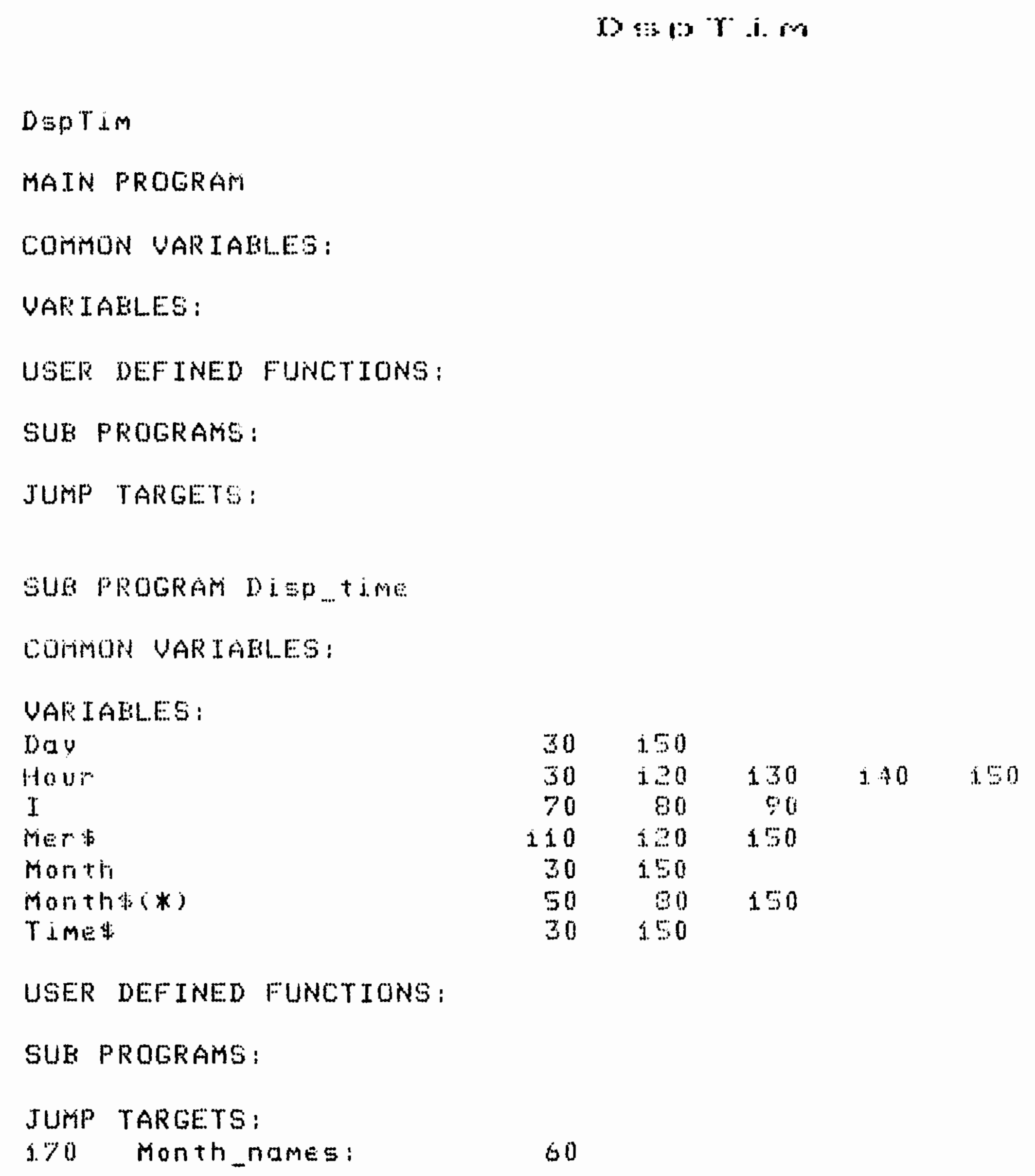


```
10 CALL Disp_time
20 GOTO 10
30 END
```



## DspTim

```
10   SUB Disp_time        !Comment: Display Time from 98035A Real
        Time Clock

20     OUTPUT 9;"Request time"
30     ENTER 9;Month,Day,Hour,Time$

Comment: Set up month name array

50   DIM Month$(12)[5]
60   RESTORE Month_names
70      FOR I=1 TO 12
80      READ Month$(I)
90      NEXT I

Comment: Put date< time in readable form

110  Mer$="AM"                              !Comment: Assume AM
120  IF Hour>11 THEN Mer$="PM"
130  IF Hour>12 THEN Hour=Hour-12
140  IF Hour=0 THEN Hour=12
150  DISP Month$(Month);Day,Hour;CHR$(8);":";Time$;"   ";Mer$
160  SUBEXIT
170 Month_names: DATA JAN,FEB,MARCH,APRIL,MAY,JUNE,JULY,AUG,SEPT,OCT,
       NOV,DEC
180  SUBEND
```



```
                                    DspTim

DspTim

MAIN PROGRAM

COMMON VARIABLES:

VARIABLES:

USER DEFINED FUNCTIONS:

SUB PROGRAMS:

JUMP TARGETS:


SUB PROGRAM Disp_time

COMMON VARIABLES:

VARIABLES:
Day                         30   150
Hour                        30   120   130   140   150
I                           70    80    90
Mer$                       110   120   150
Month                       30   150
Month$(*)                   50    80   150
Time$                       30   150

USER DEFINED FUNCTIONS:

SUB PROGRAMS:

JUMP TARGETS:
170   Month_names:          60
```

# Menu Selection

## Object of Function

This multiple-line function allows you to display a “menu”, a list of choices, on the CRT. You may then position an inverse video cursor over the choice you wish to make. When CONT is pressed the number of the desired item will be returned as the value of the function. For example, suppose that a program reaches a point at which it can take seven different courses of action depending on what the user wants to do at this point.

START OVER

STOP THE PROGRAM

STORE AN ITEM

ENTER AN ITEM

ALTER AN ITEM

LIST ALL ITEMS

DELETE AN ITEM

When these choices are entered into a string array (e.g. Menu$(1)=“START OVER”, Menu$(2)=“STOP THE PROGRAM”, etc.), they can be passed to the FNMenu function. A choice of formats is allowed. The number of rows and columns which are used to print the Menu are passed as parameters, as is a flag for double spacing. A title for the menu may also be passed to the function. There is an option allowing you to specify that the menu is already on the screen and does not need to be rewritten.

For instance, if the Menu$ string array contains the choices listed above and it was desired that the Menu be titled ‘THINGS YOU CAN DO”, the statement

```
150 Choice=FNMenu(1,"THINGS YOU CAN DO",Menu$(*),3,3,2)
```

will cause the following to be printed on the screen:

```
                           THINGS YOU CAN DO
START OVER                ENTER AN ITEM          DELETE AN ITEM

STOP THE PROGRAM          ALTER AN ITEM

STORE AN ITEM             LIST ALL ITEMS
```

The function first notes that the Mode (“1” in the parameter list) is positive. This indicates that the Menu is NOT already on the screen. It will have to be printed. The mode specifies which item the cursor will initially be set to. For instance, if Mode=3, then the cursor will initially be over the third Menu item, “STORE AN ITEM”.

Since the title is not null, “”, it is printed and underlined at the top of the screen. Note that this title might have been passed by assigning it to a string variable (e.g. Title$) and passing the variable as a parameter. If the string passed as the title is null then no title line will be printed. The title is followed by two blank lines.

The third parameter is the string array of choices that can be made. The final three parameters specify the format in which the choices are printed on the screen. The first "3" indicates there are to be 3 rows of menu items. The second "3" specifies there will be 3 columns, which is to say that each item can be 24 characters long: 80 characters per line minus the 4 blanks after each row but the last, divided by 3 columns ((80−4*(3−1)) DIV 3). Any choice longer than 24 characters will be truncated to 24. If longer items were desired then the number of columns should be smaller and the number of rows larger (e.g. 7 for Rows and 1 for Columns will print one choice per line).

The final parameter specifies the type of spacing. The "2" in this example indicates double spacing, allowing a blank row between each row of choices. "1" would have indicated single spacing.

If any items exceed the limits of the parameters they will be truncated. If the length of a choice exceeds the size of a column (80 divided by the number of columns), then it is truncated to column size. If the number of choices exceeds the number of printable Menu items (numbers of rows * number of columns), then only those items that fit will be available on the screen. Choices which are null strings will not be printed on the screen and will not take up any screen space. If there is more than one column the first column will be filled, then the second columm and so on. If there are not enough items to fill the last column, then those Menu positions will be blank and inaccessible by the cursor.

When the Menu has been printed the inverse video cursor will be over the mode specified item. The arrows in the display section of the keyboard will move the position of the cursor in the direction of the arrow. The cursor will "wrap around" (e.g. if you are pushing the up−arrow and the cursor is at the top of the screen, the cursor will move to the bottom row of the next column to the right).

The cursor can be moved as many times as desired. When the CONT key is pressed, the number of the item over which the cursor is currently positioned will be returned as the value of the function. For example, if the cursor is over "ALTER AN ITEM" when CONT is pressed the the value 5 will be returned for FNMenu since "ALTER AN ITEM" is Menu$(4). Menu$(0) is the first item. This value might be used by following the program line in which FNMenu is invoked by an ON...GOSUB... or ON...GOTO... statement. For example:

```
.

.

150 Choice=FNMenu(1,Title$,Menu$(*),7,1,1)

160 ON Choice GOSUB Begin,Stop,Enter,Alter,Delete,Store,List

.

.
```

Now that the Menu is printed on the CRT the function FNMenu may be called again with the Mode set to a negative value (e.g. −1) and the Menu would not have to be printed again. All the other parameters must be the same as when the function was initially invoked. If any of the items have been changed, or any change made to the information displayed on the CRT then the negative Mode should not be used. The cursor will be positioned according to the absolute value of the mode.

## User Instructions

The function FNMenu is in DATA file "MENU".

A 9845B (not a 9845A) is necessary to run this function due to the use of ON KBD statements, selective addressing of the CRT and the use of the ONKBDG binary program for the 9845B.

This is a function and must be attached to a program or subprogram for use. In addition, it requires the ONKBDG binary to be in the 9845B memory in order to work. This binary allows the use of the ON KBD statement in a subprogram or function. The binary should be loaded prior to linking the function to your program. The ONKBDG binary is to be loaded by typing:

**LOAD BIN "ONKBDG"**

and pressing EXECUTE. Once the binary is loaded then you may link on the function FNMenu by following the steps indicated in the Introduction.

If the linked files are STOREd as PROG files then the binary program will be stored with it. The LOAD used to load this file will load the binary too. However, if this file is SAVEd then the binary will NOT be included. This means that you will have to follow the procedure desribed above to LOAD the binary file before GETting the combined file.

### Calling syntax

```
150 X=FNMenu(Mode,Title$,Menu$(*),Rows,Cols,Spaces)
```

### Function parameters

1) Mode:REAL value or REAL valued variable; if value is negative then function assumes that Menu is already on the CRT; if value is positive then CRT is cleared and the Menu specified is printed on the CRT; initial cursor position is over Menu$(MIN(1,ABS(Mode)))
2) Title$:STRING or STRING variable which is the title to be printed above the Menu (maximum length: 80 characters); will appear underlined
3) Menu$(*):one dimension STRING array; each element of the array is taken to be a Menu choice.
4) Rows:INTEGER or integer valued real or variable specifying the number of rows printed in the menu (maximum 20 if no title;17 if there is a title)
5) Cols:INTEGER or integer valued real or variable specifying the number of columns printed in the menu (maximum 16)
6) Spaces:INTEGER or integer valued real or variable indicating single or double spacing; if Spaces<2 then single spacing used; if Spaces>=2 then double spacing used.

### Returned value

1) The value returned is the index into Menu$(*) specifying the Menu item indicated by the cursor at the time when CONT is pressed.

# MENU

```
10 Fnmenu:    !
20    DEF FNMenu(Mode,Title$,Menu$(*),Rows,Cols,Spaces)
30    ! MODE = POSITIVE ... WRITE MENU$ TO THE  SCREEN, SET CURSOR AT
       MENU$(MODE).
40    !      = NEGATIVE ... MENU$ IS ASSUMED TO BE ON THE SCREEN, SET
       CURSOR AR
50    !                         MENU$(ABS(MODE)).
60    !      = 0 ... MODE IS SET TO 1
70    ! TITLE$ ... IF NOT "" THEN TITLE$ IS PRINTED CENTERED AND
       UNDERLINED ON
80    !                TOP OF THE SCREEN AND MENU ITEMS START 3 LINES DOWN.
90    !            ... IF "" THEN NO TITLE IS PRINTED AND MENU ITEMS START
       AT THE TOP
100   !                OF THE SCREEN.
110   ! MENU$(*) ... CONTAINS THE MENU ITEMS, ANY ELEMENT = "" IS NOT
       CONSIDERED
120   !                  TO BE A MENU ITEM.
130   ! ROWS ... THE NUMBER OF ROWS IN WHICH TO PRINT MENU$.
140   ! COLS ... THE NUMBER OF COLUMNS IN WHICH TO PRINT MENU$.
150   ! SPACES <= 1 ... SINGLE SPACE THE MENU ITEMS.
160   !        >= 2 ... DOUBLE SPACE THE MENU ITEMS.
170   ! FUNCTION RETURN VALUE ... THE OPTION BASE 1 INDEX OF MENU$ THAT
       CONTAINS
180   !                               THE SELECTED MENU ITEM.
190   ! NOTE: ALL PARAMETERS ARE ADJUSTED IF NECESSARY TO ENSURE THAT
       THE FUNCTION
200   !          WILL ALWAYS WORK (I.E. THE MAXIMUM MENU ITEMS IS SET TO
       ROWS*COLS,
210   !          MAXIMUM ROWS AND COLS ARE SET TO WHAT WILL FIT ON THE
       SCREEN, IF
220   !          MENU$(ABS(MODE)) IS NOT A MENU ITEM THEN THE CURSOR IS
       SET TO THE
230   !          NEAREST VALID MENU ITEM, IF ANY ELEMENT OF MENU$ IS TOO
       LONG FOR
240   !          THE COLUMN SPECIFICATION IT WILL BE TRUNCATED.
250     ON KBD GOSUB Knull ,ALL
260     DIM K$[80],D$[80]
270     S=MAX(1,MIN(2,INT(Spaces)))
280     T=3*(LEN(Title$)>0)
290     R=MAX(1,MIN((20-T) DIV S,INT(Rows)))
300     C=MAX(1,MIN(16,INT(Cols)))
310     M=Mode
320     Ys=(80-4*(C-1)) DIV C
330     IF NOT M THEN M=1
340     DIM M$(ROW(Menu$)-1)[80]
350     DEF FNA$(X,Y,X$)=CHR$(27)&"&a"&VAL$(X)&"y"&VAL$(Y)&"C"&X$
360 K:    IMAGE #,K
370     Curx=0
380     Maxx=0
390     IF M>0 THEN PRINT USING K;CHR$(27),"H",CHR$(27),"J"
```

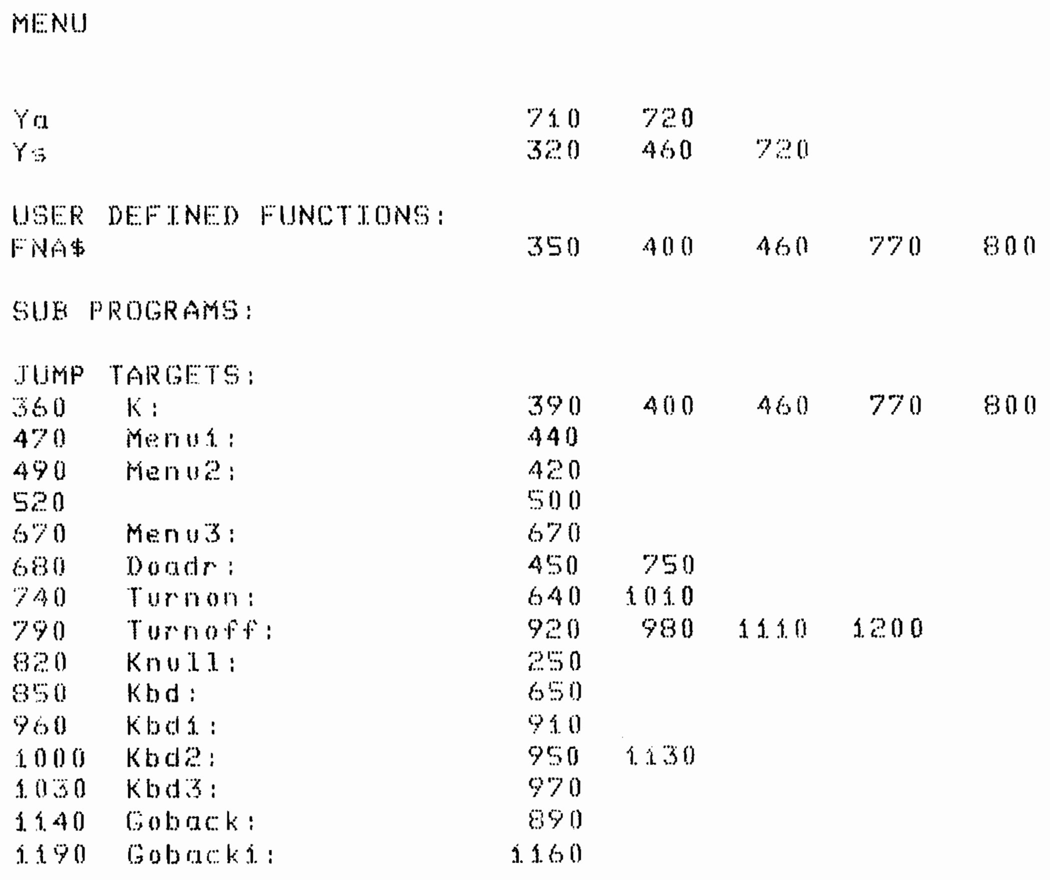
MENU

```
400    IF T*(M>0) THEN PRINT USING K;FNA$(0,39-LEN(Title$) DIV 2,CHR$(
          132)&Title$&CHR$(128))
410    FOR A=0 TO ROW(Menu$)-1
420      IF NOT LEN(Menu$(A)) THEN Menu2
430      M$(Curx)=Menu$(A)[1;MIN(LEN(Menu$(A)),80)]
440      IF M<0 THEN Menu1
450      GOSUB Doadr
460      PRINT USING K;FNA$(X,Y,M$(Curx)[1;MIN(Ys,LEN(M$(Curx)))]&CHR$(
            128))
470  Menu1:     !
480      Curx=Maxx=Maxx+1
490  Menu2:     !
500      IF R*C=Curx THEN 520
510    NEXT A
520    Curx=-1
530    M=MIN(ABS(M),R*C)-1
540    FOR A=0 TO M
550      IF LEN(Menu$(A)) THEN Curx=Curx+1
560    NEXT A
570    IF Curx=-1 THEN Curx=0
580    IF C*R>1 THEN D$="Use the "
590    IF R>1 THEN D$=D$&CHR$(247)&CHR$(224)
600    IF C>1 THEN D$=D$&CHR$(248)&CHR$(240)
610    IF C*R>1 THEN D$=D$&" arrows to select an item, "
620    D$=D$&"use CONT to execute the selected item."
630    DISP D$
640    GOSUB Turnon
650    ON KBD GOSUB Kbd ,ALL
660    BEEP
670  Menu3:   GOTO Menu3
680  Doadr:  !
690    Xa=Curx MOD R
700    X=Xa*S+T
710    Ya=Curx DIV R
720    Y=4*Ya+Ys*Ya
730    RETURN
740  Turnon:  !
750    GOSUB Doadr
760    IF NOT Maxx THEN RETURN
770    PRINT USING K;FNA$(X,Y,CHR$(129))
780    RETURN
790  Turnoff:  !
800    PRINT USING K;FNA$(X,Y,CHR$(128))
810    RETURN
820  Knull:  !
830    K$=KBD$
840    RETURN
850  Kbd:  !
860    K$=KBD$
870    IF K$[1;1]<CHR$(255) THEN RETURN
```

```
880   K=NUM(K$[2;1])
890   IF K=19 THEN Goback
900   IF Maxx=1 THEN RETURN
910   IF NOT ((K)=24)*(K(=25)) THEN Kbd1
920   GOSUB Turnoff
930   IF K=24 THEN Curx=(Curx-1) MOD Maxx
940   IF K=25 THEN Curx=(Curx+1) MOD Maxx
950   GOTO Kbd2
960 Kbd1:  !
970   IF K MOD 128<>28 THEN Kbd3
980   GOSUB Turnoff
990   Curx=(K>128)*(Maxx-1)
1000 Kbd2: !
1010  GOSUB Turnon
1020  RETURN
1030 Kbd3: !
1040  IF (K<22)+(K>23) THEN RETURN
1050  Ss=1
1060  IF K=22 THEN Ss=-1
1070  Cx=Curx+Ss*R
1080  IF Cx>=Maxx THEN Cx=(Xa+1) MOD MIN(Maxx,R)
1090  IF Cx<0 THEN Cx=Maxx DIV R*R+(Xa-1) MOD MIN(Maxx,R)
1100  IF Cx>=Maxx THEN Cx=Cx-R*(1+(Cx-Maxx) DIV R)
1110  GOSUB Turnoff
1120  Curx=Cx
1130  GOTO Kbd2
1140 Goback: !
1150  FOR C=0 TO ROW(Menu$)-1
1160    IF M$(Curx)=Menu$(C) THEN Goback1
1170  NEXT C
1180  C=-1
1190 Goback1: !
1200  GOSUB Turnoff
1210  DISP ""
1220  RETURN C+1
1230  FNEND
```

# MENU

```
MENU

MAIN PROGRAM

COMMON VARIABLES:

VARIABLES:

USER DEFINED FUNCTIONS:

SUB PROGRAMS:

JUMP TARGETS:
10    Fnmenu:


FNMenu

COMMON VARIABLES:

VARIABLES:
A                          410   420   430   510   540   550   560
C                          300   320   500   530   580   600   610  1150
                                1160  1170  1180  1220
Cols                        20   300
Curx                       370   430   460   480   500   520   550   570
                                 690   710   930   940   990  1070  1120
                                1160
Cx                        1070  1080  1090  1100  1120
D$                         260   580   590   600   610   620   630
K                          880   890   910   930   940   970   990  1040
                                1060
K$                         260   830   860   870   880
M                          310   330   390   400   440   530   540
M$(*)                      340   430   460  1160
Maxx                       380   480   760   900   930   940   990  1080
                                1090  1100
Menu$(*)                    20   340   410   420   430   550  1150  1160
Mode                        20   310
R                          290   500   530   580   590   610   690   710
                                1070  1080  1090  1100
Rows                        20   290
S                          270   290   700
Spaces                      20   270
Ss                        1050  1060  1070
T                          280   290   400   700
Title$                      20   280   400
X                          350   460   700   770   800
X$                         350
Xa                         690   700  1080  1090
Y                          350   460   720   770   800
```



```
MENU


Ya                                   710    720
Ys                                   320    460    720

USER DEFINED FUNCTIONS:
FNA$                                 350    400    460    770    800

SUB PROGRAMS:

JUMP TARGETS:
360    K:                            390    400    460    770    800
470    Menu1:                        440
490    Menu2:                        420
520                                  500
670    Menu3:                        670
680    Doadr:                        450    750
740    Turnon:                       640   1010
790    Turnoff:                      920    980   1110   1200
820    Knull:                        250
850    Kbd:                          650
960    Kbd1:                         910
1000   Kbd2:                         950   1130
1030   Kbd3:                         970
1140   Goback:                       890
1190   Goback1:                     1160
```

HEWLETT
PACKARD

Part No. 09845-10151

Printed in U.S.A.
1st Edition, April 1979