Tektronix
4050 Series

Applications Library
Program Documentation

# Applications Library

Applications Library
Applications Library
Applications Library
Applications Library
Applications Library
Applications Library
Applications Library
Applications Library
Applications Library
Applications Library
Applications Library
Applications Library

PROGRAMMING AIDS T2

Prepared by the 4050 Series Applications Library. The 4050 Series Applications Library is maintained as a service for our customers by the Information Display Division of Tektronix, Inc., Group 451, P.O. Box 500, Beaverton, Oregon 97077 U.S.A.

Tektronix®
COMMITTED TO EXCELLENCE

# PROGRAMMING AIDS T2

# 062-5972-01

# DOCUMENTATION

Applications Library
Group 451
Tektronix, Inc.
P.O. Box 500
Beaverton, Oregon 97007

062-5972-00

6/81

Tektronix®
COMMITTED TO EXCELLENCE

# DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

| TITLE | PART NUMBER |
|---|---|
| PROGRAMMING AIDS T2 | 062-5972-01 |
| ORIGINAL DATE: June, 1981 / REVISION DATE: | |


ABSTRACT

PROGRAMMING AIDS T2 is a tape collection of 15 programs PLUS the Programming Tips handbook to aid you in creating or dissecting a 4050 BASIC program. Employ these routines to follow a program's structure, track variables or change them, enhance graphs. Take a look at the abstract describing the novel program which converts bases (Hexadecimal Operations). The Programming Tips handbook has been a best seller and is sure to expand and streamline your 4050 operations.

The individual abstracts describe the programs.

Tape file 1 contains the directory. Press AUTOLOAD and select your program.

Be sure to read the documentation before running a program.


| Title/ Previous Abstract # | Tape File # | Documentation Page # |
|---|---|---|
| Directory | 1 | |
| Flowcharter 51/00-8005/1 | 2 | 1 |
| Automatic Tape Directory 51/00-8022/0 | 3 | 7 |
| Tape File Header Expander 51/00-8039/0 | 4 | 12 |
| List Program's Statement Types 51/00-8001/0 | 5 | 18 |
| Sort & List Program Variables 51/00-8003/0 | 6 | 23 |
| Change & List Program Variables 51/00-8028/0 | 7 | 28 |

| TITLE | PART NUMBER |
|---|---|
| PROGRAMMING AIDS T2 | 062-5972-01 |

| TITLE | PART NUMBER |
|---|---|
| PROGRAMMING AIDS T2 | 062-5972-01 |

## TRANSFERRING FILES TO A NEW TAPE

PLOT 50 General Utilities Vol. 1 (TEKTRONIX Part #4050A08) contains a program to transfer any type of 4050 files (program/data/text) quickly and easily along with the header names; however, it requires a 4924 Tape Drive.

### Transferring ASCII or BINARY PROGRAMS without a transfer program

Step 1. Do a TLIST of the MASTER program tape.

Step 2. Record which files go with which program (they are all named) and the size of each file.

Step 3. MARK your new tape to accept the respective files for that program, e.g.,

```
FIND 0
MARK 1,20000
FIND 2
MARK 1,4000
etc.
```

Step 4. Insert the MASTER tape.

```
FIND a file
OLD for ASCII or CALL "BOLD" for BINARY
```

Step 5. Insertthe new tape

```
FIND the file to receive the file in memory
SAVE for ASCII or CALL "BSAVE" for BINARY
```

REPEAT Steps 4 and 5 until all files comprising that program are transferred to the new tape. Note: This procedure will not retain the file header names.

### Transferring ASCII or BINARY DATA to a new tape

The 4051R06 Editor ROM could be used to transfer ASCII DATA files.

4050 Applications Library program "Binary Data File Duplicator" will transfer BINARY DATA files without any peripheral.

4050 Applications Library program "Tape Duplication" will transfer ASCII or BINARY DATA or PROGRAM files, but requires a 4924 Tape Drive.

Both of these programs are contained on the 4050 Applications Library UTILITIES T1 tape (TEKTRONIX Part # 062-5974-01), and UTILITIES D1 disk (TEKTRONIX Part #062-5975-01).

Tektronix®
COMMITTED TO EXCELLENCE

# DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

TITLE

Flowcharter

| ORIGINAL DATE | REVISION DATE |
|---|---|
| November, 1976 | September, 1977 |

EQUIPMENT AND OPTIONS REQUIRED

16K

AUTHOR Han Klinkspoor, Tektronix, Inc.
Rev. by: Leland C. Sheppard, Sheppard Software
Sunnyvale, CA

PERIPHERALS

ABSTRACT

Files: 1 ASCII Program

Statements: 405

This program will flowchart any 4050 BASIC program from a tape file. It does the job in the following way:

In the first pass, a map of the branches is made to enable "look ahead" in the second pass. In the second pass, program lines are processed one at a time. The line number is stripped off and the branch table is examined to draw incoming or outgoing branches, if any. As each entry in the branch table is processed, the page number on which that reference occurred is plugged back into the branch table for subsequent printing. As the program is charted, the current page number and the starting and ending statement numbers shown on that page are printed on the bottom of the page.

It includes page number references in the branch table and the range of statement numbers shown on a page are printed with the page number at the bottom.

On a 16K machine, the program will chart programs with up to 170 branches. On 24K or 32K machines, it will chart programs with 700 or more branches.

Restrictions: Limit of 4 character statement numbers to allow the program to run on a 16K machine. This may be modified to 5. Maximum of 20 FOR/NEXT loops unless modified to increase the limit. Page limit is 99 but may be modified.

TITLE

Flowcharter

## METHODOLOGY

This is a rewrite of 51/00-8005/0; the purpose of the rewrite was to increase the speed of the program while reducing its appetite for memory. Some compromises were made to allow the program to be run on a 16K machine; please see the section on restrictions. The overall flow of the program was not modified from the original.

This program will flowchart any 405 BASIC program from a tape file. It does the job in the following way:

In the first pass, a map of the branches is made to enable "look ahead" in the second pass.

In the second pass, program lines are processed one at a time. The line number is stripped off and the branch table is examined to draw incoming or outgoing branches, if any.

As each entry in the branch table is processed in pass 2 the page # on which that reference occurred is plugged back into the branch table for subsequent printing. As the program is charted the current page # and the starting and ending statement numbers that are shown on that page are printed at the bottom of the page.

The output from the printing of the branch table will look as follows:

| BR# | FROM | PAGE | TO | PAGE | | BR# | FROM | PAGE | TO | PAGE | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 100 | 2 | | 34 | 1700 | 12 | 4000 | 22 | SUBR.CALL |
| 2 | 8 | 1 | 7000 | 28 | SUBR.CALL | 35 | 1720 | 12 | 1910 | 14 | |
| 3 | 17 | 1 | 17 | 1 | | 36 | 1750 | 12 | 1790 | 13 | |
| 4 | 32 | 1 | 8000 | 31 | | 37 | 1760 | 13 | 1780 | 13 | |
| 5 | 40 | 2 | 1000 | 7 | | 38 | 1770 | 13 | 1790 | 13 | |
| 6 | 280 | 3 | 8000 | 31 | SUBR.CALL | 39 | 1700 | 13 | 5000 | 25 | SUBR.CALL |
| 7 | 390 | 3 | 1000 | 7 | INTERRUPT | 40 | 1010 | 13 | 1060 | 14 | |

The detailed description of the FLOWCHARTERs function is as follows:

In the first pass all branches ("from" and "to" statement numbers) are stored in strings D$ ("from") and E$ ("to"). Strings J$ and K$ are used to store flags indicating sub-routine calls or interrupt service requests (to differentiate

TITLE

FLOWCHART ER

them from normal branches).

Each element in D$ and E$ is 4 bytes; each in J$ and K$ is 2 bytes. The flags are contained in one of the 2; the 2nd will be used for page numbers during pass 2.

An example: (assume the first statement encountered in the program to be charted is "1 GO TO 100")

```
D$ positions 1-4: "   1"  ("from" or origin)
E$ positions 1-4: " 100"  ("to" or destination)
J$ positions 1-2: "  "    (no flags, not GOSUB)
K$ positions 1-2: "  "    (no flags, not ON)
```

No page numbers have been placed in J$ or K$ since we are still in pass 1 (they may not be printable anyway since they are stored in binary. More about that later).

Another example: assume the following:

```
D$ positions 41-44: " 510"
E$ positions 41-44: "3000"
J$ positions 21-22: "- "
K$ positions 21-22: "  "
```

This would mean that branch #11 was a GOSUB in statement 510 to statement 3000 ("510 GOSUB 3000").

In the second pass program lines are processed one at a time. The statement number is stripped off and saved (if first or perhaps last on a page). The branch table is examined for incoming branches. The Basic Verb is checked and if it is different from the previous one a box is drawn around command(s) on the screen (this can result in several statements of a similar type in one box). The statement is then checked for outgoing branches, FOR/NEXT connections to be drawn, etc. At the point where the incoming and outgoing branch checks are made the current page number is stored in J$ (for incoming) or K$ (for outgoing) as a 1 byte binary counter. This technique was used as a memory saving device so that the FLOWCHARTER could be run on a 16K 4051. The functions used to convert to and from binary are ASC and

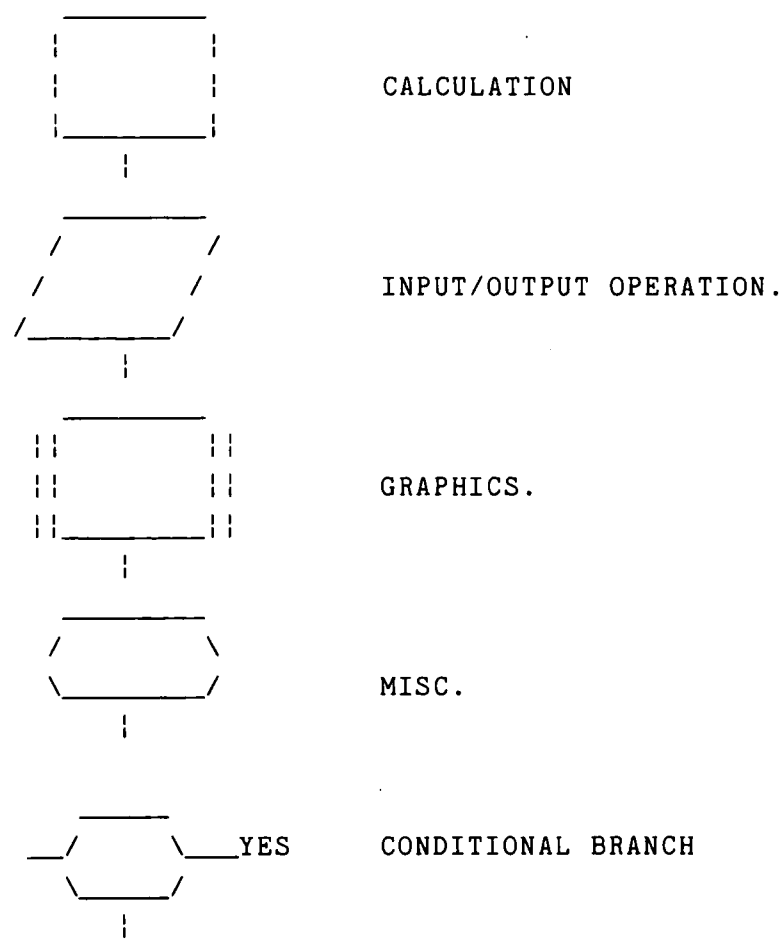
TITLE

FLOWCHARTER

CHR. Paging is done by the program after drawing unsatisfied loops to continue over the page boundary. The page number is printed in the lower left corner along with the first and last statement numbers which appeared on that page.

Operation of the program is as follows: Load the program and type RUN. After a short delay during initialization the program will display the flowchart symbols used (the program does it graphically; it is reproduced here using print characters for illustration only).

F L O W C H A R T  S Y M B O L S



TITLE

FLOWCHARTER

SUBROUTINE CALL.

(The graphics and subroutine call boxes are actually double boxes which could not readily be reproduced here.)

After a short pause the screen will page and a request for the file number of the program to be charted will appear:

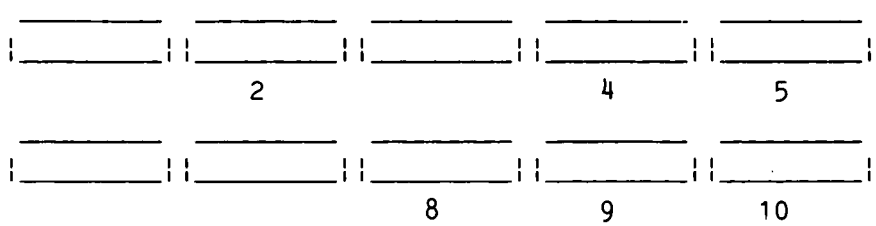
Specify File # to be flowcharted:

Respond with your program file #. The screen will clear.
At the end of pass 1 the following message will appear:

Flowchart between line numbers:

Respond with "1,9999" for the entire file or a range (less than that) of statement numbers of your choice.

The function keys available are as follows:




Key 2: Print the branch table (automatic on range 1-9999; use key when flowcharting part of program)

Key 4: Suspend execution (1 instruction loop where screen is left as is; useful for studying a page)

TITLE

FLOWCHARTER

Key 5: Run (Resume execution after Suspend; picks up where it left off)

Key 8: Display Symbols (same as starting display)

Key 9: End (terminate the run immediately)

Key 10: Restart Pass 2 (line number request is reissued so a new range can be specified if desired)

Memory Requirements: 9K for program; will run on a 16K machine and chart programs with up to 170 branches; on 24K or 32K machines it will chart programs with 700 or more branches.

Peripherals Required: None. 4631 Hardcopy optional.

Restrictions:

1. statement numbers 1-9999 allowed; modify N and alter the statements which initialize G$ and H$ to allow 5 character statement numbers (the limit of 4 was used to allow the program to run on a 16K machine).
2. Maximum of 20 FOR/NEXT loops can be handled in one program without modification; increase F and F$ and the respective initialization and search loops to increase this limit.
3. the page limit is 99 without modification to the image used to print the branch table; the page limit is 127 for the one byte binary counters.
4. when a partial range is specified for the 2nd pass the page number references in the branch table will be incomplete and will print (during branch table print) as page # 32; the flowcharter cannot distinguish between a blank (decimal 32) and a real page number (since J$ and K$ were initialized to blanks and never changed).

# APPLICATIONS LIBRARY PROGRAM

| TITLE | | |
|---|---|---|
| AUTOMATIC TAPE DIRECTORY | | |
| ORIGINAL DATE<br>January 1978 | REVISION DATE | MEMORY REQUIREMENT<br>8K |
| AUTHOR<br>E.A. Bleiweiss | University of New Mexico<br>Civil Engineering Research | PERIPHERALS<br>None |

ABSTRACT

Statements: 150

Files: 1 ASCII Program

The program will list in order the contents of a magnetic tape cartridge by file number, size, type (ASCII or BINARY) and contents (title of Program). It will then load and run a selected program if requested.

The program requires the first 2 files on the tape. File 1 is for the program, file 2 for the data (Table of Contents).

When used for the first time the program will scan each file starting with file 3, then read and list the first line of each file. The first line of each program must be a REM statement and contain the title in braces { }. Up to 48 characters may be used in the title.

The maximum title storage capability in file 2 is 43 files. If more are required file 2 may be marked larger and the appropriate lines of code changed in the program.

NOTE: Requires a data file.

TITLE

AUTOMATIC TAPE DIRECTORY

## OPERATING INSTRUCTIONS

The first line of each program on tape must be a REM statement and contain its title in braces { }. Whatever is between the braces (to a maximum of 48 characters) will be printed out as the contents, all else will be ignored.

Procedure for first use on a new tape:

1. Load program into memory
2. Change line 350 to show title of tape
3. FIND 0
4. MARK 2,5000
5. FIND 1
6. SAVE
7. F1=1
8. RUN 280

Procedure for later use:

1. AUTOLOAD or
2. FIND 1
3. OLD
4. RUN

## PROGRAM LIMITATIONS

The maximum title storage capability in file #2 is 43 files with titles of maximum length (48 characters). If the user requires a larger storage capability he may wish to mark file #2 larger than 5120 and change line 410 to reflect the new size.

TITLE

AUTOMATIC TAPE DIRECTORY

EXAMPLE

Procedure for new tape after program has been loaded.

```
FIN0
MARK 2,5000
FIN1
SAVE
F1=1
RUN 280
MAKE SURE THE TAPE SAFTEY IS OFF,
THEN ENTER THE PRESENT DATE: 16 AUG 78
```

TITLE

AUTOMATIC TAPE DIRECTORY

EXAMPLE

Output of tape that has many programs (before update).

```
***4051 APPLICATIONS LIBRARY PROGRAMS***

FILE    SIZE    TYPE    CONTENTS
----    ----    ----    --------
 1      5120    ASCII   AUTOMATIC TAPE DIRECTORY/17JAN78
 2      5120    BINARY  DIRECTORY DATA
 3      5120    ASCII   2-LINE LABEL PROGRAM
 4      5120    ASCII   MODIFIED 2-LINE LABEL PROGRAM
 5      768     LAST

CURRENT AS OF: DECEMBER 11, 1978

DO YOU WANT AN UPDATE:
```

TITLE

AUTOMATIC TAPE DIRECTORY

EXAMPLE

Output of same tape (after update).

```
***4051 APPLICATIONS LIBRARY PROGRAMS***

FILE    SIZE     TYPE     CONTENTS
----    ----     ----     --------
 1      5120     ASCII    AUTOMATIC TAPE DIRECTORY/17JAN78
 2      5120     BINARY   DIRECTORY DATA
 3      5120     ASCII    2-LINE LABEL PROGRAM
 4      5120     ASCII    MODIFIED 2-LINE LABEL PROGRAM
 5      1280     ASCII    SOFTWARE CHARACTER GENERATOR
 6      1792     ASCII    SOFTWARE CHARACTER GENERATOR
 7      2304     ASCII    DATA /
 8      8192     ASCII    DRAW
 9      2048     BINARY   DATA /
10      1792     ASCII    DATA /
11      10752    ASCII    DRAW
12      4608     ASCII    DASHED LINES
13      768      ASCII    DATA GRAPHING
14      22016    ASCII    DATA GRAPHING
15      1792     NEW
16      768      LAST

CURRENT AS OF: DECEMBER 18, 1978

DO YOU WANT TO RUN A PROGRAM:
```

Tektronix
COMMITTED TO EXCELLENCE

# DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

| TITLE | |
|---|---|
| Tape File Header Expander | EQUIPMENT AND OPTIONS REQUIRED |
| ORIGINAL DATE: November, 1979 / REVISION DATE: | |
| AUTHOR: Randal C. Bowling, Tennessee Valley Authority, Chattanooga, TN | PERIPHERALS |

ABSTRACT

Statements: 198
Files: 1 ASCII Program

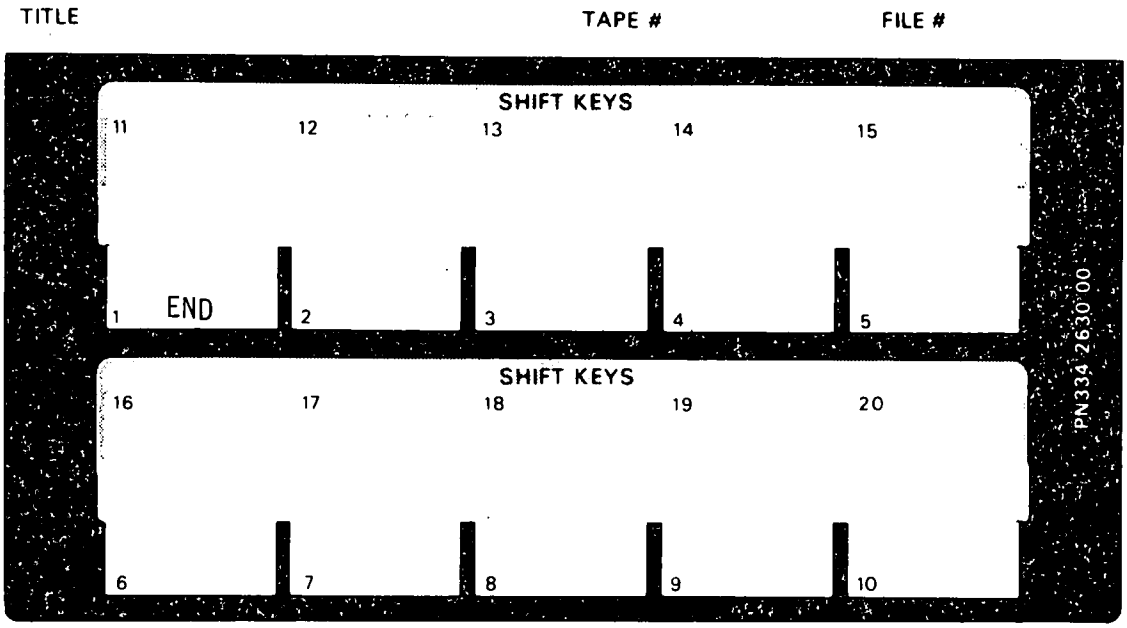
This program permits you to annotate the standard file header as well as add information in the remaining 256-byte header block not occupied by the standard header.

A descriptively annotated file allows quick identification during the normal TLIS. The expanded header would only be displayed using the expanded TLIS portion of this program.

The annotations of the standard header or its expansion does not affect the data contained in any tape file. The file may contain a program or data in ASCII or binary. Programs may be secret or open.

TITLE

Tape File Header Expander




During program execution, user-definable key 1 may be used to end the program.

If the 'BREAK' key is pressed instead, the system parameters may be left in an altered state.

TITLE

Tape File Header Expander

OPERATION

Load this program into memory.

Insert the tape containing the file(s) to be changed into your 4050 tape drive. Key in RUN and follow instructions. The program is tutorial.

E X P A N D E D  T L I S T

**Please enter the number of the operation you wish to perform.**

**1 Display an EXPANDED TLIST**

**2 Expand a File Header**

**3 Annotate the Standard Header**

**4 Normal TLIST**

**5 END**

Annotation of the Standard Header changes various bytes on the existing file header to reflect pertinent information. This information will be revealed in a normal TLIST.

The expanded file header prints text to that portion of the header block following the standard header. This information will be displayed when program 1, above, is chosen.

Any integer greater than 4 will end the program.

The program may be ended at any time by pressing User-Definable Key 1.

Either of these two methods returns the system to power-up status. Ending the program by using the 'BREAK' key may leave system parameters altered.

All user inputs are validated. All yes/no queries specifically require a Y or N entry. Either header input is checked for length. The user may enter headers shorter than those required, but not longer.

All files numbers input must be positive integers less than 256 since 255 is the maximum number of files that may be recorded on a single tape cartridge.

| TITLE | ABSTRACT NUMBER |
|---|---|
| Tape File Header Expander | 51/00-8039/U |

## PROGRAM LIMITATIONS

Certain bytes of the existing header may not be changed. They are 1 through 4, 9, 17, and 35 to 42. Furthermore, numerical data may not be inserted into bytes 15-34. If the original header contains an "S" in byte 27, it may not be changed. The program checks for illegal entries and corrects them or requests new input.

The expanded file header is limited to 33 characters to permit the expanded TLIST to keep both the standard header and the expanded header on one 4050 display line.

## METHODOLOGY

The standard file header is composed of 44 bytes. However, a 256-byte header block is reserved for the header. This results in a large unused portion of the header block. The "Expand a File Header" portion of the program takes advantage of this to include additional information within the header.

Anything may be included in the unused header space.

The "Annotate the Standard Header" inserts descriptive information into unused bytes within the existing header. The program checks to be sure only valid data is included, and that certain bytes are not changed within the standard header. The necessary CR and control-S (DC3) are appended to the standard header revision.

## OBSERVATIONS

If a program is saved or data printed to an annotated file header, bytes 10-27 will be rewritten by the 4050 to standard notation, i.e., ASCII PROGRAM, ASCII DATA, etc. Thus, this program will have to be used again on those files to reflect the desired annotations.

| TITLE | ABSTRACT NUMBER |
|---|---|
| Tape File Header Expander | 51/00-8039/0 |

EXAMPLES

```
 1    DO NOT USE- BAD TAPE AREA    4096
 2     TVA          P RANDY BOWLING  6656
 3        A         P                768
 4    B   A G    S  P                1024
 5    A   A A    O  P                1536
 6    T   A L    F  P                1280
 7    T   A A    T  P                1280
 8    L   A C    W  P                2048
 9    E   A T    A  P                768
10        A I    R  P                768
11    S   A C    E  P                768
12    T   A A    |  P                6144
13    A   A |    |  P                768
14    R   A |    |  P                768
15    |   B |    |  D                29184
16    |   B |    |  D                30464
17   \|/  A\|/  \|/ P                768
18        L                         768
```

Fig. 1. Normal TLIS showing annotated header.

```
 1 DO NOT USE- BAD TAPE AREA      4096
 2  TVA          P RANDY BOWLING  6656 E X P A N D E D   T L I S T
 3     A         P                 768 BATTLE- STAR GALACTICA MENU
 4 B   A G    S  P                1024 CIRCULAR SCANNER
 5 A   A A    O  P                1536 EXPANDING FIELD SCANNER
 6 T   A L    F  P                1280
 7 T   A A    T  P                1280 OSCILLATING SCANNER
 8 L   A C    W  P                2048 RADAR SCANNER
 9 E   A T    A  P                 768 WINDSHIELD WIPER SCANNER
10    A I    R  P                 768 POSTS DOILY
11 S  A C    E  P                 768 STAR FIELD
12 T  A A    |  P                6144 SPACE TAG
13 A  A |    |  P                 768 RANDOM PATTERNS
14 R  A |    |  P                 768 TRANSFORM #1
15 |  B |    |  D               29184
16 |  B |    |  D               30464
17\|/ A\|/ \|/ P                 768 TRANSFORM #2
18    L                          768
```

Fig. 2. Expanded TLIS showing annotated and expanded header.

| TITLE | ABSTRACT NUMBER |
|---|---|
| Tape File Header Expander | 51/00-8039/0 |

```
 1 DO NOT USE- BAD TAPE AREA          4096
 2  TVASCII    PROGRAM    WLING      6656 E X P A N D E D   T L I S T
 3     A          P                   768 BATTLE- STAR GALACTICA MENU
 4 B   A G    S P                    1024 CIRCULAR SCANNER
 5 A   A A    O P                    1536 EXPANDING FIELD SCANNER
 6 T   A L    F P                    1280 DEFENSE VECTOR SCANNER
 7 T   A A    T P                    1280 OSCILLATING SCANNER
 8 L   A C    W P                    2048 RADAR SCANNER
 9 E   A T    A P                     768 WINDSHIELD WIPER SCANNER
10    A I    R P                      768 POSTS DOILY
11 S   A C    E P                     768 STAR FIELD
12 T   A A    | P                    6144 SPACE TAG
13 A   A |    | P                     768 RANDOM PATTERNS
14 R   A |    | P                     768 TRANSFORM #1
15 |   B |    | D                  29184
16 |   B |    | D                  30464
17\|/  A\|/  \|/P                    768 TRANSFORM #2
18     L                             768
```

Fig. 3. The program on file 2 was corrected and then saved. Note the 4051 changed the header to standard notation on bytes 10-27.

Tektronix®
COMMITTED TO EXCELLENCE

# DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

| TITLE | |
|---|---|
| List Program's Statement Types | EQUIPMENT AND OPTIONS REQUIRED |
| ORIGINAL DATE: October, 1976 / REVISION DATE: | 8K |
| AUTHOR: Brian Diehm, Tektronix, Inc., Wilsonville, OR | PERIPHERALS |

ABSTRACT

Files: 1 ASCII Program

Statements: 113

This program reads a tape file containing a BASIC ASCII program and prints an alphabetized list of all the statement types used in that program, together with the number of occurrences of each statement type.

The list is sorted in decreasing order by number of occurrences and reprinted.

The program first asks the user which tape file contains the program to be analyzed. Then, after reading the file, the alphabetized list of statement types is printed, with the count of the occurrences of each type. This is followed by a total of the number of statements in the analyzed program. Provision is made to allow processing of several files, combining the results into one list.

| TITLE | |
|---|---|
| LIST PROGRAM'S STATEMENT TYPES | |

HARDWARE REQUIREMENTS

Any size 4050 will support this program. The built-in tape drive of the 4050 is utilized.

PROGRAM LIMITATIONS

The tape file that is analyzed with this program must be valid BASIC program code. It should have been stored on the tape by utilizing the SAVE statement. Any size program may be analyzed.

METHODOLOGY

All statement type names are considered 12 character entities, 50 of which may be stored in a 600 character array. Trailing spaces are added to short names to make them 12 characters.

The input file is read line by line and each line is analyzed to determine the statement type. When this is determined, it is compared with the list of statement types already encountered. If this is not new, the appropriate count is incremented, if it has not been encountered before, it is added to the list in the appropriate place.

When the file is completely read, the list is printed, first in alphabetical order by statement types and then in descending numeric order by number of statement occurrences.

OPERATING HINTS

Note that the program differentiates between LET statements. LET statements that do not contain the LET keyword are listed as LET(IMPLIED).

If you wish to list the file as it is being analyzed, insert the following line into the code:

```
295 PRINT L$
```

OPERATING INSTRUCTIONS

When first run, the program will print the following message:

```
SORT AND LIST STATEMENT TYPES
Enter File Number:  ?
```

TITLE

LIST PROGRAM'S STATEMENT TYPES

Enter the number of the file on which is stored the program you wish to analyze. Follow with RETURN.

Next the program asks:

Is this a continuation of a previous run?:

Type Y or N, followed by RETURN. If you respond N, the program's lists of already found statement types is cleared before proceeding. If Y is the response, it is assumed that the results of this run should be combined with the results of the previous run of the program. If the program has not been run since loading, you must respond N.

After reading the file, the alphabetized list is printed. For example, when the program is run on itself, the following list is printed:

```
STATEMENT TYPES USED (ALPHABETICALLY)

DIM             1
END             1
FIND            2
FOR             5
GO TO           4
GOSUB           1
IF              19
INPUT           4
LET(IMPLIED)    41
NEXT            5
ON              1
PAGE            3
PRINT           15
REM             9
RETURN          1

TOTAL NUMBER OF STATEMENTS:  112
Press RETURN to continue.
```

TITLE

LIST PROGRAM'S STATEMENT TYPES

```
        STATEMENT TYPES USED (USE ORDER):

        LET(IMPLIED)   41
        IF             19
        PRINT          15
        REM            9
        FOR            5
        NEXT           5
        GO TO          4
        INPUT          4
        PAGE           3
        FIND           2
        DIM            1
        END            1
        GOSUB          1
        ON             1
        RETURN         1

        TOTAL NUMBER OF STATEMENTS:  112

DATA STRUCTURES:

Memory Allocation for Variables:  2131 bytes

Memory Allocation for Program:  3030 bytes
```

**TITLE**

LIST PROGRAM'S STATEMENT TYPES

**VARIABLE MAP**

| VARIABLE | TAPE FILE | USAGE |
|---|---|---|
| F | | Work Variable |
| I | | Work Variable |
| J | | Work Variable |
| M1(50) | | Array containing the current number of occurrences of each statement type in use order. |
| N | | Current number of Statement Types found |
| N1(50) | | Array containing the current number of occurrences of each Statement Type in alphabetical order. |
| A$(12) | | Work String |
| B$(12) | | Work String |
| L$(72) | | String containing current statement being analyzed |
| M$(600) | | String containing Statement Types in use order. |
| N$(600) | | String containing Statement Types. |

Tektronix®
COMMITTED TO EXCELLENCE

# DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

| TITLE | |
|---|---|
| Sort & List Program Variables | |
| ORIGINAL DATE: November, 1976 / REVISION DATE: | EQUIPMENT AND OPTIONS REQUIRED: 8K |
| AUTHOR: Dan Taylor, Tektronix, Inc., Wilsonville, OR | PERIPHERALS: |

ABSTRACT

Files: 1 ASCII Program

Statements: 118

This program reads a BASIC ASCII program from tape and produces an alphabetized table of the variables used in that program. The BASIC program may be stored on multiple tape files as long as the files are sequential on tape. No operator intervention is required between first and last files.

TITLE

SORT & LIST PROGRAM VARIABLES

## Program Limitations

The tape file (s) to be analyzed must be valid 405 BASIC programs. The programs should have been stored on tape with the SAVE command.

## Methodology

The imput files are read line by line and searched for variable names. When a variable name is found, it is compared with the names already found, to eliminate duplication. If the name is not already stored, it is inserted in its proper alphabetical location. When the files are completely read, the names are printed:

TITLE

SORT & LIST PROGRAM VARIABLES

Here is a sample of the table produced:

VARIABLES:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| A | AØ | A1 | A2 | A3 | A4 | A5 | | | |
| | | B1 | B2 | B3 | B4 | B5 | B6 | B7 | B8 |
| C | CØ | C1 | C2 | C3 | C4 | C5 | C6 | | |
| D | | | | | | | | | |
| | | F1 | F2 | F3 | F4 | F5 | F6 | F7 | F8 |
| I | | | | | | | | | |
| J | | | | | | | | | |
| N | | | | | | | | | |
| | | | | S3 | | | | | |
| | | | | T3 | | | | | |

If string variables had been used they would appear in the last column of the table.

The blank spaces in the table are the variables which are not used at all in your program.

TITLE

SORT & LIST PROGRAM VARIABLES

Operating Instructions

After the program is in the machine type RUN:

SORT AND LIST PROGRAM VARIABLES

NOTE: YOUR PROGRAM MAY BE ON SEVERAL TAPE FILES.

Begin with File Number = 14

End with File Number = 19

FILE # 14

FILE # 15

FILE # 16

FILE # 17

FILE # 18

FILE # 19

When the last file has been processed (19 in this example) the program pages the screen and prints the table of the variables used. (see Methodology)

TITLE

SORT & LIST PROGRAM VARIABLES

Variables

A$ (1) - the character previous to the one being analyzed

B$ (1) - the character being analyzed

C$ (1) - the character following the one in B$

F - the number of the tape file currently being read

F1 - beginning file number

F2 - ending file number

J - character position in L$ of A$

JØ - character position in L$ of B$

J1 - character position in L$ of C$

J2 - scratch variable

K$ (3) - the first 3 characters of a line of code which follow the first space. (eg K$ might be REM or PRI, etc.)

L$ (74) - string to hold a line of input from the file

N - the number of different variables found

N$ (572) - holds the variable names

T$ (1) - used when printing out the table scratch

V$ (2) - the name of a variable (eg A1)

W$ (2) - scratch

Tektronix®
COMMITTED TO EXCELLENCE

# DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

| TITLE | |
|---|---|
| Change & List Program Variables | |
| ORIGINAL DATE / REVISION DATE | EQUIPMENT AND OPTIONS REQUIRED |
| | 8K |
| AUTHOR S. Schicktanz<br>Technical University<br>Muich, Germany | PERIPHERALS |

ABSTRACT

Files: 1 ASCII Program

Statements: 143

The program allows listing or changing the names of the variables of an ASCII program from tape. Listing the program is also available.

When changing variable names, input is tested for validity and correct type, errors being indicated by an appropriate message. The changed program can be output to the original tape file or another.

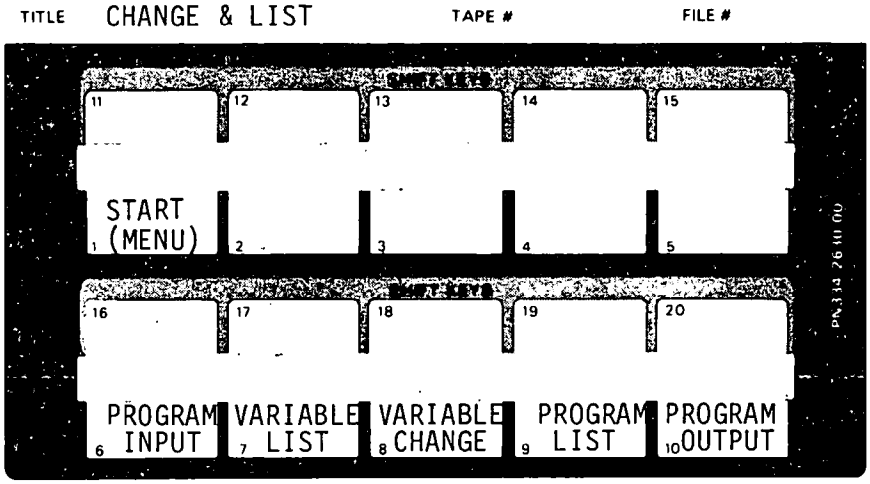
The program can be used either with the menu or the User-Definable Keys. The user is prompted for necessary input information by use of the POInter statement. It is not necessary to terminate input with the RETURN key-

User-Definable Keys provide:

Menu
Program input
Variable list
Variable change
Program list
Program Output

TITLE

CHANGE & LIST PROGRAM VARIABLES




| | | |
|---|---|---|
| UDK #1 | START (MENU) | Initializes program, lists menu on screen |
| UDK #6 | PROGRAM INPUT | Requests file number of program to be changed |
| UDK #7 | VARIABLE LIST | Lists program variables |
| UDK #8 | VARIABLE CHANGE | Activates the variable change routine |
| UDK #9 | PROGRAM LIST | List program |
| UDK #10 | PROGRAM OUTPUT | Saves program into original file |

TITLE

CHANGE & LIST PROGRAM VARIABLES

OPERATING INSTRUCTIONS

Press AUTOLOAD if program is the first file on tape.

If not the first file on tape:

FIND #

OLD

RUN or press User-Definable Key #1

The program clears the screen and prints the following:

```
File Nr.
```

No provision is made to make sure that the selected file contains a program or that its contents fit into the available memory space. If the program is to long, program execution is aborted, and the error message INVALID FUNCTION PARAMETER IN LINE 250... appears. If input is completed, a list of choices for program activities is given:

```
File Nr. 1
V-list variables
P-  "  program
C-change variables
```

Input of the appropriate character causes the desired activity to start -- the RETURN-key must not be pressed!

For activities "V" and "P" nothing else must be done - just be a little patient if listing variables, as a lot of searching must be done.

When changing variable names "C", the user is prompted to give the names.

```
Variable to be changed:
```

The input is tested for validity as a name and matching of type, but not for occurrence in the program. An error is indicated by an appropriate message, and the program asks for new input.

TITLE

CHANGE & LIST PROGRAM VARIABLES

To terminate the "change variable"-mode, just press the RETURN-key as input for the variable to be changed.

-- CAUTION: the RETURN-key must not be pressed in any other case except to give a program file number, or to make a no-character entry either to input a one-character variable or to end the "change variable"-mode.

Having finished the first activity, the program offers two new choices:

```
O-output new program
N-new start
```

"O" causes the program worked on to be written on tape in its changed version to the file that it came from.

If "N" is specified, the working space is cleared and the program asks you for another program file number. If "O" has not been given before, the tape stays in its original state, any changes being lost.

To terminate work select "N" as entry to the menu and specify 0 as file number; the tape will be rewound to be ready for use again.

The alternate way to work with the program is by use of the User-Definable Keys #6 to #10. They give access to all of the five basic activities of the program.

To save output to a file other than the original

FINd the desired file,

MARk 1, LEN(P$) if necessary,

FINd the file again,

PRInt @33: P$ and

CLOse

TITLE

CHANGE & LIST PROGRAM VARIABLES

INTERNAL DATA STORAGE

| Variable | Used to Store | Type |
|---|---|---|
| F | Keyboard input file number | Simple |
| H$ | Special characters | String |
| I | For next loop | Simple |
| J | ASCII character 48-57 | Simple |
| K | Pointer | Simple |
| K$ | Input program file | String |
| M | Input program 1 = continue, 0 = Stop EOF | Simple |
| P | Memory | Simple |
| P$ | Input program loaded | String |
| S$ | ASCII character 13 "CR" | String |
| V | Length of input variable | Simple |
| V$ | Input variable searched for | String |
| W$ | Input variable substitute | String |
| X | Set pointer position | Simple |
| Y | Set pointer position to input character | Simple |

Tektronix®
COMMITTED TO EXCELLENCE

# DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

| TITLE | | |
|---|---|---|
| Variable Name Changer | | |
| ORIGINAL DATE | REVISION DATE | EQUIPMENT AND OPTIONS REQUIRED |
| September, 1980 | | 8K |
| AUTHOR Mallory M. Green<br>U.S. Dept. of HUD<br>Washington, D. C. | | PERIPHERALS<br>4907 File Manager |

ABSTRACT

Files: 1 Program

Statements: 167

Variable Name Changer allows a programmer to change any number of variable names in his program with ease. The program to be changed must be an ASCII file on tape in the 4050 internal tape drive. The revised program is written as an ASCII file on the 4907 disk, leaving the original program intact on tape.

Variable names in REMark statements are not changed.

Variable Name Changer prompts the user for a list of current variables to be changed and for their new names; it then prompts for the file number of the program to be revised. The file is read a statement at a time. If it is not a REMark or IMAge statement, it is scanned character by character for variables. When a variable to be changed is discovered, it is replaced with the new variable name. The process continues until the whole program has been read from tape and written to disk.

TITLE

VARIABLE NAME CHANGER

Methodology

The VARIABLE NAME CHANGER program first builds an index of variables to be changed that includes the new variable names. Next a record is read. If this record is a Remark or Image statement it is written directly out to disk. If the record is not a Remark or Image statement it is scanned character by character for variables. When a variable to be changed is discovered, it is replaced with the new variable name. This process continues until the whole program has been read from tape and written to disk.

TITLE

VARIABLE NAME CHANGER

Operating Instructions

The program to be read must be stored on the internal tape cartridge in ASCII format. This is done using the SAVE command. When the program is started, the user is asked for a list of variables to be changed. Once this list of variables is complete, the user is then asked to enter the file # where the program to be revised is stored. Then the user is asked for a disk file name to write the new ASCII program copy. After this file name is supplied, execution begins and the new program copy is written to disk. The new program can be loaded into memory from disk with the command: OLD "file name", "ASCII".

TITLE

VARIABLE NAME CHANGER

Example Run

```
ENTER VARIABLE NAME CHANGES DESIRED

PRESS 'RETURN' FOR 'OLD NAME' TO EXIT

OLD VARIABLE NAME                    NEW VARIABLE NAME
*****************                    *****************
OLD = A$                             NEW = X$
OLD = I$                             NEW = P$
OLD = M$                             NEW = A$
OLD = A                              NEW = Y
OLD = G                              NEW = N
OLD = G0                             NEW = N0
OLD = G1                             NEW = P1
OLD = G2                             NEW = P2
OLD = G3                             NEW = P3
OLD = G4                             NEW = I4
OLD = G5                             NEW = P4
OLD = G6                             NEW = P5
OLD = G7                             NEW = P6
OLD = S                              NEW = U
OLD = U                              NEW = X0
OLD = Y                              NEW = Y0
OLD =

ENTER INPUT TAPE FILE # OR '0' TO STOP? 5

ENTER OUTPUT ASCII DISK FILE NAME? @B2/MAIN
```

TITLE

VARIABLE NAME CHANGER

Program Remarks Outline

PROGRAM VARIABLES MODIFIER '$VARMOD' 7/17/79

| M# | LINE# | MODULE NAME | MODULES CALLED |
|---|---|---|---|
| 1 | 100 | FLOW CONTROL | 2 3 4 8 9 |
| 2 | 300 | DIM | |
| 3 | 400 | INITIALIZE | |
| 4 | 600 | SPECIFY VARIABLE CHANGES | 5 |
| 5 | 900 | UPDATA CHANGE MATRICE | 6 6 |
| 6 | 1100 | CONVERT Z$ TO Z1,Z2 | |
| 7 | 1400 | CONVERT Z1,Z2 TO Z$ | |
| 8 | 1600 | SET UP FILES | |
| 9 | 2000 | SCAN LINE L$ & CHANGE TO LINE M$ | 10 11 |
| 10 | 2900 | ESTABLISH J2,J3,A$,B$,C$ GIVEN J1 | |
| 11 | 3000 | CHANGE M$ WHEN NEEDED | 6 7 |

TITLE

VARIABLE NAME CHANGER

Program Variables

| | |
|---|---|
| A1(26,12) | A-Z Index Pointer |
| A2(26,12) | $-9 Index Pointer |
| A$(1) | Preceeding Character |
| B$(1) | Character being scanned |
| C$(1) | Next Character |
| D$ | Scratch |
| F$ | Disk output file name |
| F | Tape input file # |
| I$ | Scratch |
| I | Scratch |
| J$ | Scratch |
| J | Scratch |
| J1 | Position of A$ in data line |
| J2 | Position of B$ in data line |
| J3 | Position of C$ in data line |
| K$ | Scratch |
| L$ | Input line |
| M$ | Output line |
| N$ | New variable |
| O$ | Old variable |
| S | Spacing difference between L$ and M$ |
| Y1 | Index pointer |
| Y2 | Index pointer |
| Z1 | Index pointer |
| Z2 | Index pointer |
| Z$ | Scratch |

Tektronix®
COMMITTED TO EXCELLENCE

# DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

| TITLE | |
|---|---|
| Program Module Cross-Reference & Map | |
| ORIGINAL DATE: January, 1978 / REVISION DATE: | EQUIPMENT AND OPTIONS REQUIRED: 16K |
| AUTHOR: Captain S. K. Sanford, Aberdeen Proving Ground, MD | PERIPHERALS: Optional - 4051R06 Editor, 4924 Tape Drive |


ABSTRACT

Files: 2 ASCII Program

Statements: 276

The Cross-Reference program requires that the user create two files of calling and called subprogram names using the 4051R06 Editor ROM or a simple BASIC program. The first file must be sorted in calling program sequence (alphabetically), while the second, which is identical, must be sorted in called program sequence.

The program reads the created data files from the 4050 or a 4924 Tape Drive one at a time and produces a listing of the calling programs with their called programs, then a listing of the called programs with their calling programs. The pages of output are numbered alphabetically from "a" to "zz", and may be automatically copied by the 4631 Hard Copy Unit.

The Modeul Map program requires a file on tape showing the calling and called module names, and their interrelationships (an example is included). The program searches the tape for all occurrences of a calling program and records the called program modules. In order for a program to appear on the module map, it must be called by another program, with the exception of the "MAIN" program (input by the user), and an optional "BLOCK DATA" program for use with FORTRAN program systems.

The map appears as multiple pages on the 4050 and may be automatically copied. Each page is headed by one program. The programs called by this first program are displayed in blocks linked by arrows to the first block. From each called program block is an arrowhead and the number of the page on which that program appears.

TITLE

PROGRAM MODULE CROSS REFERENCE & MAP

## CROSS REFERENCE PROGRAM

### Description

The user first creates two files of calling and called subprogram names under the Editor, or with a simple BASIC program. The first file is sorted in calling program sequence (alphabetically), while the second, which is identical, is sorted in called program sequence. The X-ref program then reads these files, one at a time, and produces a listing of the calling programs with their called programs, then a listing of the called programs with their calling programs. The pages of output are numbered alphabetically from "a" to "zz".

### Data Tape Structure

| | |
|---|---|
| File Number | 1 (May be changed: See variable FØ) |
| Type | ASCII |
| Format | Each record is 12 characters long.<br>Characters 1-6: Calling program name.<br>Characters 7-12: Called program name.<br>Sorted in alphabetic sequence by calling program name. |
| Unit | 33 (May be changed: See variable UØ) |
| Marking Instructions | File is for READ-ONLY. |
| File Number | 2 (May be changed: Always FØ + 1) |
| Type | ASCII |
| Format | Same as first file, except sorted in alphabetic sequence by called program name. |
| Unit | 33 (May be changed: See variable UØ) |
| Marking Instructions | File is for READ-ONLY. |

TITLE

PROGRAM MODULE CROSS REFERENCE & MAP

Internal Data Storage

| Variable | Used to Store | Type |
|---|---|---|
| UØ | Data unit address | Simple |
| FØ | First data file number | Simple |
| A$ | Input data string | String (12) |
| B$ | Program name buffer | String (6) |
| C$ | Program name input buffer | String (6) |
| D$ | Program name input buffer | String (6) |
| R$ | Input response string | String (1) |
| PØ | Index for page labeling | Simple |
| P1 | Index for page labeling | Simple |
| Z9 | SRQ input code | Simple |
| F$ | Part of page label | String (72) |
| R1 | Autocopy switch | Simple |
| I | For-loop index | Simple |
| E$ | Page label | String (72) |

TITLE

PROGRAM MODULE CROSS REFERENCE & MAP

## Methods

Operating parameters are solicited. Note that UØ must be changed manually, as well as FØ, to use a different input unit or file. The first prepared data file is read. The calling program name is compared with the stored (previous) calling program name. If unlike, the "MODULE ... CALLS" message is printed, as well as the called program name. The calling program name is stored for future comparison. If the calling program names match, only the called program name is listed.

Only 28 lines per page are printed, as controlled by the for-loop. When this loop is exited, the page label is computed from indices PØ and P1, and printed. A "WAIT" or "AUTOCOPY" is then executed.

If end of file is encountered, Z9 is set to 12 and the second file is read. The procedure for this file is the same, except that the called program name is read first and compared with the stored name. On end of file for this file, the program concludes. A serious tape error will cause abnormal program termination.

## Program Loading

"OLD" the program normally from tape. If you desire to use an external tape drive for data input, set UØ (line 120) to the address of that unit. If you do not want data from files 1 and 2, change FØ (line 140) to the first data file number. The second data file must immediately follow the first on the same unit (FØ + 1).

## Program Execution

"RUN" normally. Select or reject "AUTOCOPY" (automatic page copying to a 4631 hard copier) option. If UØ = 33 (default), replace the program tape with the data tape. The program will automatically list the crossreference table. When through, replace the program tape in the internal tape drive (if UØ = 33). The program will automatically load and run the program on file 1 of the program tape, which is presumed to be a directory program.

TITLE

PROGRAM MODULE CROSS REFERENCE & MAP

Examples

```
                    *** MODULE CROSSREFERENCE ***

>AUTOCOPY? (Y/N): N
please insert DATA tape - press RETURN when ready:
```

TITLE

PROGRAM MODULE CROSS REFERENCE & MAP

*** MODULE CROSSREFERENCE ***

```
MODULE ADDALI CALLS MODULE(S):
                    BEPUT
                    NEWV
                    PGPAGC
                    RESTOV
                    SAVEV
                    UPALI
                    VACCES

MODULE ADDSUB CALLS MODULE(S):
                    CONSOL
                    NXTGEN
                    RESTOV
                    SAVEV
                    SUBCH
                    UPALI
                    VACCES

MODULE ADDSUP CALLS MODULE(S):
                    CONSOL
                    NXTGEN
                    RESTOV
                    SAVEV
                    SUBCH
                    UPALI
                    VACCES

MODULE BELLS  CALLS MODULE(S):
                    DRBELL
```

a

TITLE

PROGRAM MODULE CROSS REFERENCE & MAP

*** MODULE CROSSREFERENCE ***

```
MODULE BEPUT   CALLS MODULE(S):
                     COMPRS

MODULE CAVEAT  CALLS MODULE(S):
                     BELLS

MODULE CDBLCP  CALLS MODULE(S):
                     BELLS

MODULE CNGEN   CALLS MODULE(S):
                     BELLS

MODULE CONSOL  CALLS MODULE(S):
                     NXTGEN
                     SAVEV

MODULE CPRIO   CALLS MODULE(S):
                     RESTOV
                     SAVEV
```

b

TITLE

PROGRAM MODULE CROSS REFERENCE & MAP

## MODULE MAP

### Description

The program requires a file on tape showing the calling and called module names, and their interrationships. This tape may be inserted into the internal or external tape dirve (see variable UØ).

The program searches the tape for all occurances of a calling program, and records the called program modules. This search begins with the name of the "MAIN" program. The map progresses through the called programs showing which programs they, in turn, call. In order for a program to appear on the module map, it must be called by another program, with the exception of the "MAIN" program and an optional "BLOCK DATA" program for use with FORTRAN program systems.

The map appears as multiple pages on the 4051 console, and may be automatically copied by the 4631 Hard Copy Unit. Each page is headed by one program, and is numbered. The programs called by this first program are then displayed in blocks linked by arrows to the first block. From each called program block is an arrowhead and the number of the page on which that program appears.

The program provides tape diagnostic messages for the 4924 drive.

### Data Tape Structure

| | |
|---|---|
| File Number | 1 (May be changed: See variable FØ) |
| Type | ASCII |
| Format | Each entry consists of a six-column name of a calling program followed by a six-column name of a called program. This may be created by another program of the user's design, or preferrably, with the 4051R06 Editor ROM. This list need not appear in any special order. |
| Unit | 33 (May be changed: See variable UØ) |
| Markgin Instructions | This file is READ-ONLY. It should be marked appropriately when created by the user. |

TITLE

PROGRAM MODULE CROSS REFERENCE & MAP

Internal Data Storage

| Variable | Used to Store | Type |
|---|---|---|
| UØ | Input device number | Simple |
| FØ | Input file number | Simple |
| A$ | Main program name | String (72) |
| C$ | Input data string | String (12) |
| D$ | Calling program name | String (6) |
| E$ | Input calling program name | String (6) |
| F$ | Input called program name | String (6) |
| G$ | Calling program name table | String (1200) |
| H$ | Called program name table | String (300) |
| R$ | Utility | String (1) |
| BØ | Block data program indicator | Simple |
| B$ | Utility blank buffer (for padding) | String (72) |
| J$ | Utility string | String (72) |
| K$ | Utility string | String (72) |
| L$ | Utility string | String (72) |
| PØ | Page "NUMBER" ASCII index | Simple |
| P1 | Page "NUMBER" ASCII index | Simple |
| P2 | Page "NUMBER" ASCII index | Simple |
| IØ | Page "NUMBER" position switch | Simple |
| DØ | Calling program table index | Simple |
| D1 | Calling program table counter | Simple |
| I$ | Utility string | String (72) |
| Z9 | SRQ return code | Simple |
| GØ | Calling program search index | Simple |
| XØ | Utility graphics coordinate | Simple |
| YØ | Utility graphics coordinate | Simple |
| I | Utility | Simple |
| J | Utility | Simple |
| K | Utility | Simple |
| JØ | Utility | Simple |
| X1 | Utility graphics coordinate | Simple |
| Y1 | Utility graphics coordinate | Simple |
| G1 | Calling program table index | Simple |

TITLE

PROGRAM MODULE CROSS REFERENCE & MAP

## Methods

The initialization phase of the program consists of setting initial variable values, initializing I/O units, and soliciting setup information from the user.

Once initialized, the program searches the tape for occurances of the first calling program in the table (in this case, the "MAIN" program). When encountered, the names of its subroutines are recorded in both the calling and the called tables. When END-OF-FILE is reached, the page headed by the search calling program is prepared. A decision is made on the spacing of the blocks on the page, depending on the number of subroutines. Off-page numbers are computed based on the position of the subroutines in the calling program table. The program then prints the map, copies it if desired, and continues wite the next program name listed in the calling program table. Each map may consist of one or more pages, but "PAGE NUMBERS" refer to the entire map of one calling program, not each individual sheet.

The program terminates when all calling programs in the calling program table have been displayed.

## Program Loading

"FIND" and "OLD" normally. If the internal tape drive is to be used for data, remove the program tape and insert the data tape when solicited by the program. If an external tape drive is used, insert the data tape into the external tape drive before loading the program.

## Program Execution

Follow directions as they appear on the screen. The program must run from the beginning to its conclusion, and cannot be easily interrupted and re-started.

TITLE

PROGRAM MODULE CROSS REFERENCE & MAP

Examples

```
                    *** MODULE MAP ***

>ENTER MAIN PROGRAM NAME: ADDALI
>IS THERE A BLOCK DATA SUBPROGRAM? (Y/N): Y
>AUTOCOPY? (Y/N): N

please insert DATA tape - press RETURN when ready:
```

TITLE

PROGRAM MODULE CROSS REFERENCE & MAP

Sample Input File

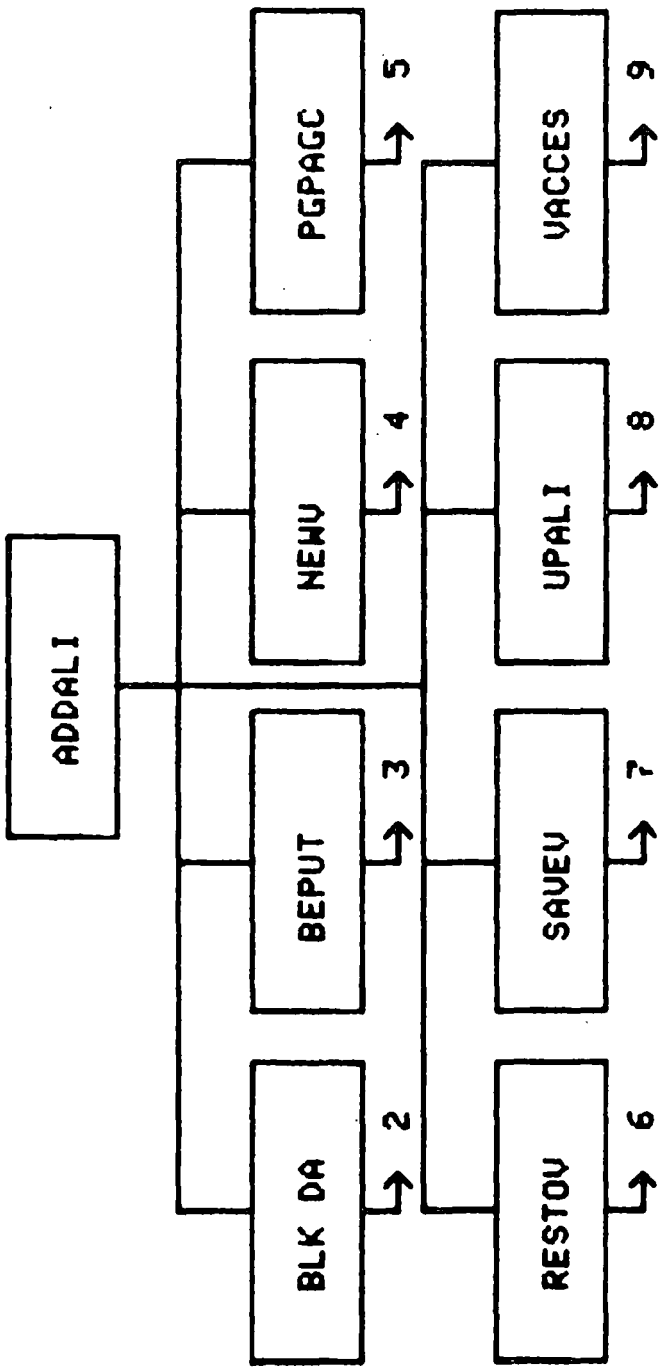
```
:ADDALIBEPUT
:ADDALINEWV
:ADDALIPGPAGC
:ADDALIRESTOV
:ADDALISAVEV
:ADDALIUPALI
:ADDALIVACCES
:ADDSUBCONSOL
:ADDSUBNXTGEN
:ADDSUBRESTOV
:ADDSUBSAVEV
:ADDSUBSUBCH
:ADDSUBUPALI
:ADDSUBVACCES
:ADDSUPCONSOL
:ADDSUPNXTGEN
:ADDSUPRESTOV
:ADDSUPSAVEV
:ADDSUPSUBCH
:ADDSUPUPALI
:ADDSUPVACCES
:BELLS DRBELL
:BEPUT COMPRS
:CAVEATBELLS
:CDBLCPBELLS
:CNGEN BELLS
:CONSOLNXTGEN
:CONSOLSAVEV
:CPRIO RESTOV
:CPRIO SAVEV
```

TITLE

PROGRAM MODULE CROSS REFERENCE & MAP

CHART #1

*** MODULE MAP ***




(a)

Tektronix®
COMMITTED TO EXCELLENCE

# DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

| TITLE | |
|---|---|
| Dash, Dot, Dash-Dot Routine | |
| ORIGINAL DATE: October, 1976 / REVISION DATE: | EQUIPMENT AND OPTIONS REQUIRED: 8K |
| AUTHOR: Nick Fkiaras, Tektronix, Inc., Wilsonville, OR | PERIPHERALS: Optional - 4662 Plotter |

ABSTRACT

Files: 1 ASCII Program

Statements: 151

The program draws a solid, dotted, dashed, or dot-dashed line between any two points, X1, Y1, and X2, Y2 . . . Xn, Yn, regardless of the window and viewport used. User inputs X and Y coordinate points, viewport, window, line type and output device.

User-Definable Keys enable the user to:

Enter data and draw

Redraw with a different line type

TITLE

DASH, DOT, DASH-DOT LINE ROUTINE

TYPICAL SEQUENCE OF OPERATION

To plot a sequence of points X(1),Y(1);X(2),Y(2);...;X(N),Y(N)

1. SET Q=GPIB address of plotting device (=32 for 4051 display)
2. SET C=0
3. SET F2=line type code:
   =2 for dot line
   =3 for dash line
   =4 for dash-dot
4. Dimension D(2),E(2),H(2)
5. SET H(1)=Length of intermediary moves in GDU's
6. SET H(2)=Length of dash
7. SET V1,V2,V3,V4 equal to your viewport parameters
8. SET W1,W2,W3,W4 equal to your window parameters
9. MOVE @Q;X(1),Y(1)
10. FOR I=2 TO N
11. X1=X(I-1)
12. X2=X(I)
13. Y1=Y(I-1)
14. Y2=Y(I)
15. GOSUB 100
16. NEXT I
17. END

TITLE

DASH, DOT, DASH-DOT LINE ROUTINE

```
VARIABLES
 X1,Y1 AND X2,Y2-COORDINATES OF THE TWO POINTS DEFINING A
                 LINE SEGMENT

 F2-LOGIC: =1 FOR SOLID LINE TYPE
           =2 FOR DOT LINE TYPE
           =3 FOR DASH LINE TYPE
           =4 FOR DASH-DOT LINE TYPE

 V1,V2,V3,V4-VIEWPORT PARAMETERS

 W1,W2,W3,W4-WINDOW PARAMETERS

 Q-GPIB ADDRESS OF PLOTTING DEVICE (=32 FOR SCREEN)

 K-LOGIC: >0 FOR DRAW
          <0 FOR MOVE

 K1-LOGIC (USED IN DASH-DOT): =2 WHEN A DOT IS PRINTED
                              =0 FOR SECOND DRAW

 L-LOGIC: =1 USE VECTOR LENGTH FOR DRAW
          =2 USE DARK VECTOR LENGTH

 H-LENGTHS FOR DASHES AND INTERMEDIARY MOVES IN GDU's
   H(1)=LENTGH OF INTERMEDIARY MOVE
   H(2)=LENGTH OF DASH

 D-X INCREMENTS IN GDU's:
   D(1)=X INCREMENT FOR MOVES
   D(2)=X INCREMENT FOR DRAWS
```

TITLE

DASH, DOT, DASH-DOT LINE ROUTINE

```
E-Y INREMENTS IN GDU's:
  E(1)=Y INCREMENT FOR MOVES
  E(2)=Y INCREMENT FOR DRAWS

F1-LOGIC: =0 IF ENTIRE LINE SEGMENT HAS NOT BEEN PLOTTED
          =1 IF ENTIRE LINE SEGMENT HAS BEEN PLOTTED

C=LENGTH OF DASH LEFT OVER AFTER THE ENTIRE LINE SEGMENT
  HAS BEEN PLOTTED

A,B,R,F3-INTERMEDIATE VALUES
```

TITLE

DASH, DOT, DASH-DOT LINE ROUTINE

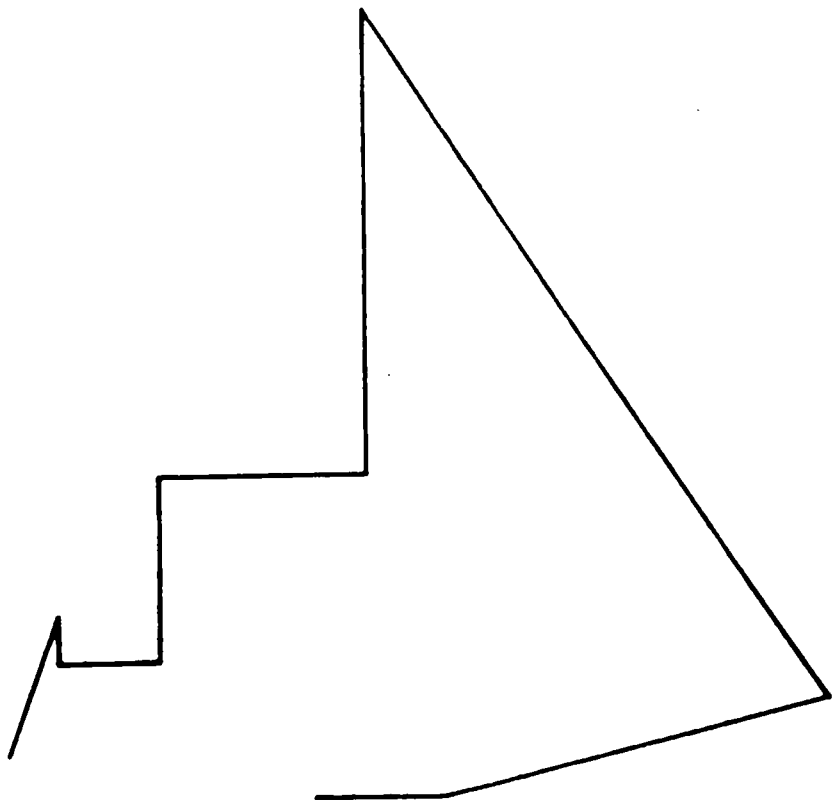
EXAMPLES

The plots on the next pages are the result of running the sample program included with the routine with some data.

```
LINE TYPE:
1  SOLID
2  DOT
3  DASH
4  DASH-DOT
YOUR CHOICE: 1
DEVICE CODE:
1  PLOTTER
32 4051 DISPLAY

YOUR CHOICE:32
```

TITLE

DASH, DOT, DASH-DOT LINE ROUTINE

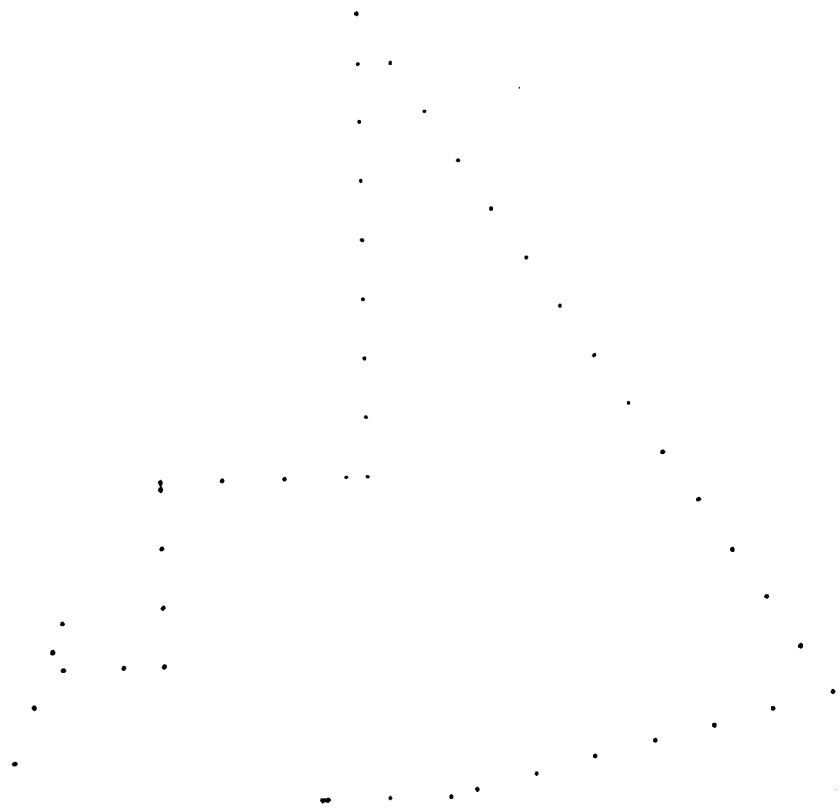


| TITLE | PART NUMBER |
| --- | --- |
| DASH, DOT, DASH-DOT LINE ROUTINE | |

```
LINE TYPE:
1  SOLID
2  DOT
3  DASH
4  DASH-DOT
YOUR CHOICE: 2
DEVICE CODE:
1  PLOTTER
32 4051 DISPLAY

YOUR CHOICE:32
```

TITLE

DASH, DOT, DASH-DOT LINE ROUTINE



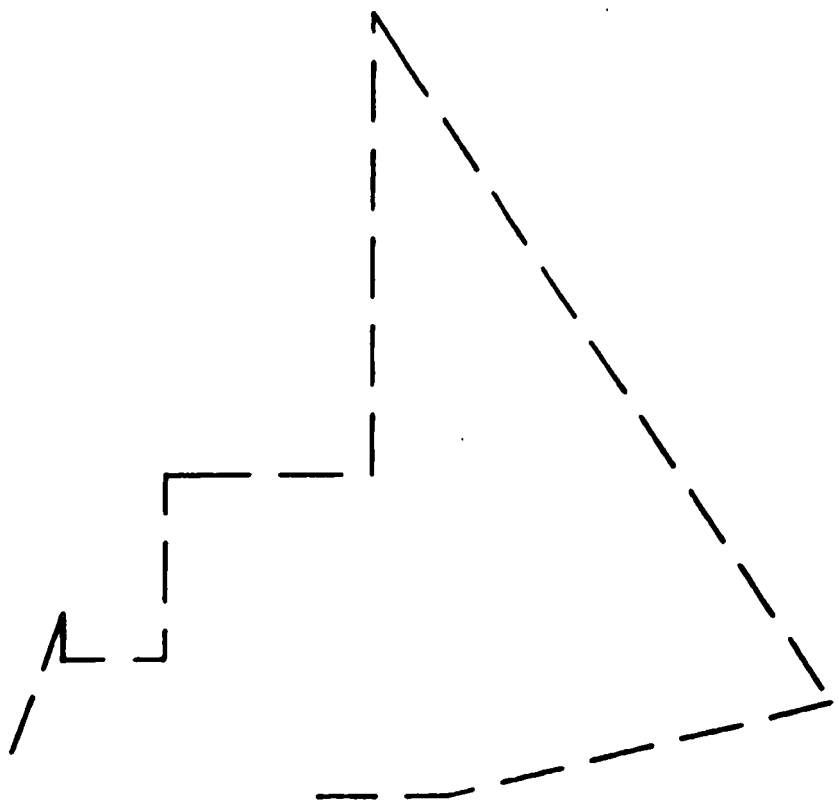
TITLE

DASH, DOT, DASH-DOT LINE ROUTINE

```
LINE TYPE:
1  SOLID
2  DOT
3  DASH
4  DASH-DOT
YOUR CHOICE: 3
DEVICE CODE:
1  PLOTTER
32 4051 DISPLAY

YOUR CHOICE:32
```

TITLE

DASH, DOT, DASH-DOT LINE ROUTINE



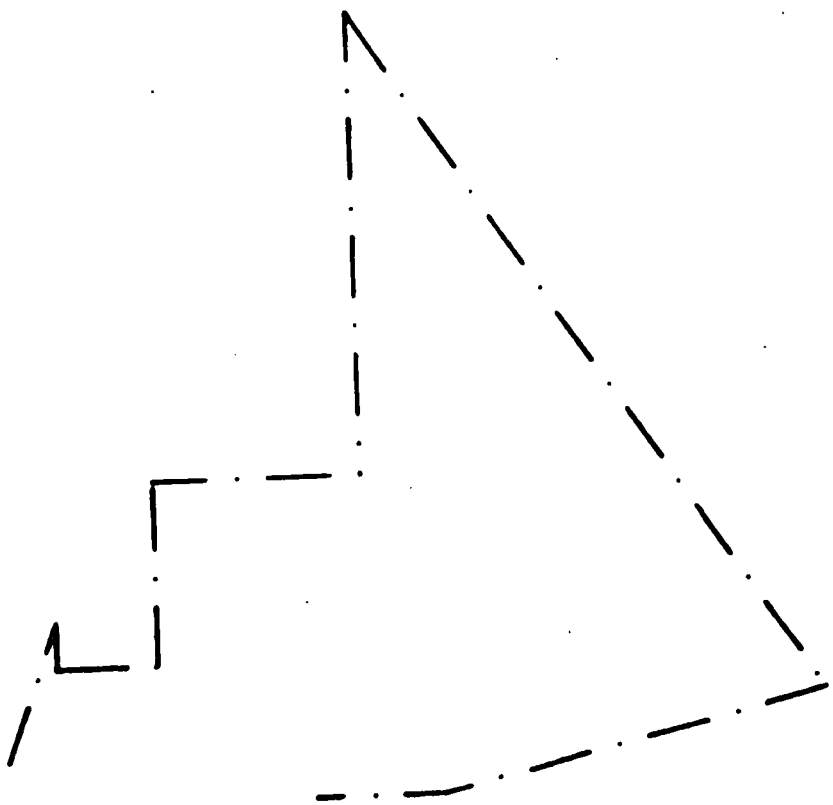
TITLE

DASH, DOT, DASH-DOT LINE ROUTINE

```
LINE TYPE:
1  SOLID
2  DOT
3  DASH
4  DASH-DOT
YOUR CHOICE: 4
DEVICE CODE:
1  PLOTTER
32 4051 DISPLAY

YOUR CHOICE:32
```

TITLE

DASH, DOT, DASH-DOT LINE ROUTINE



Tektronix
COMMITTED TO EXCELLENCE

# DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

| TITLE | |
|---|---|
| Neat Tics and Axis Labeling | |
| ORIGINAL DATE: October, 1976 / REVISION DATE: | EQUIPMENT AND OPTIONS REQUIRED: 8K |
| AUTHOR: Dan Taylor & Kathy Thurman<br>Tektronix, Inc.<br>Wilsonville, OR | PERIPHERALS |

ABSTRACT

Files: 1 ASCII Program

Statements: 78

This program is a subroutine designed to be used with a user program. The subroutine prepares the screen for a user's graph by:

1. Calculating "neat" tick lengths
2. Setting the WINDOW
3. Setting the VIEWPORT
4. Drawing an axis and labeling the tic marks. The axis is drawn through user data value 0 (or data min if 0 is not in the WINDOW). Tic labels always appear to the left and bottom of the screen.

Tic marks on the axis are presumed to be evenly spaced (not logarithmic).

Labels are printed in scientific notation for either axis if any label on that axis = 10.

Requires minimum and maximum data values and number of tic intervals desired on each axis.

The viewport allows room on the screen for a title to be printed above the graph.

**TITLE**

NEAT TICS AND AXIS LABELING

HARDWARE REQUIREMENTS

4051 8K memory - 2.3K for this program

PROGRAM LIMITATIONS

Tic marks on the axis are presumed to be evenly spaced (not logarithmic). Labels are printed in scientific notation for either axis if any label on that axis $\geq$ 10.

METHODOLOGY

1. The user specifies the old minimum and maximum data values and the number of tic intervals desired on each axis.
2. Given the old min, max and number of tic intervals, calculate the "neat" tic interval. The tic interval is always (1, 2 or 5) times 10 to an integer power (plus, minus or zero).
3. Given the tic interval, calculate

   new max > old max

   new min < old min
4. The new min and max are both integer multiples of the tic interval. This means that the new min, new max and Ø all fall on tic marks.
5. Number of tic intervals produced = (new max - new min)/tic interval length
   (Number of intervals/$\sqrt{2.5}$) < # produced < ($\sqrt{2.5}$*Number of intervals + 2)
   In practice the number produced is usually quite close to the number requested.
6. The axis is drawn through user data value Ø (or data min if Ø is not in the window).

OPERATING HINTS

VIEWPORT is set by the routine as 15, 130, 10, 95. The right and upper parameters (130, 95) can be decreased if desired, but the left and lower parameters (15, 10) must not be decreased due to label location. The view port allows room on the screen for the user to print a title above the graph on the first line of the screen.

If line 1270 is changed to:

```
AXIS G(3), G(7), G(1), G(5)
```

then the axis will always be drawn in the lower left corner of the screen.

TITLE

NEAT TICS AND AXIS LABELING

REFERENCES

Lewart, C.R., Algorithm 463, Algorithm SCALE 1, SCALE 2 and SCALE 3 for Determination of Scales on Computer Generated Plots, Comm. ACM, Vol. 16, #10, (Oct. 1973) pp. 639-640.

MAN/MACHINE INTERFACE

This algorithm is in the form of a subroutine accessible from a user program.

OPERATING INSTRUCTIONS

Before calling this subroutine the user must store:

DIM G(8)
X data minimum G(1)
X data maximum G(2)
Y data minimum G(5)
Y data maximum G(6)
Number of tic intervals desired, X axis G(3)
Number of tic intervals desired, Y axis G(7)

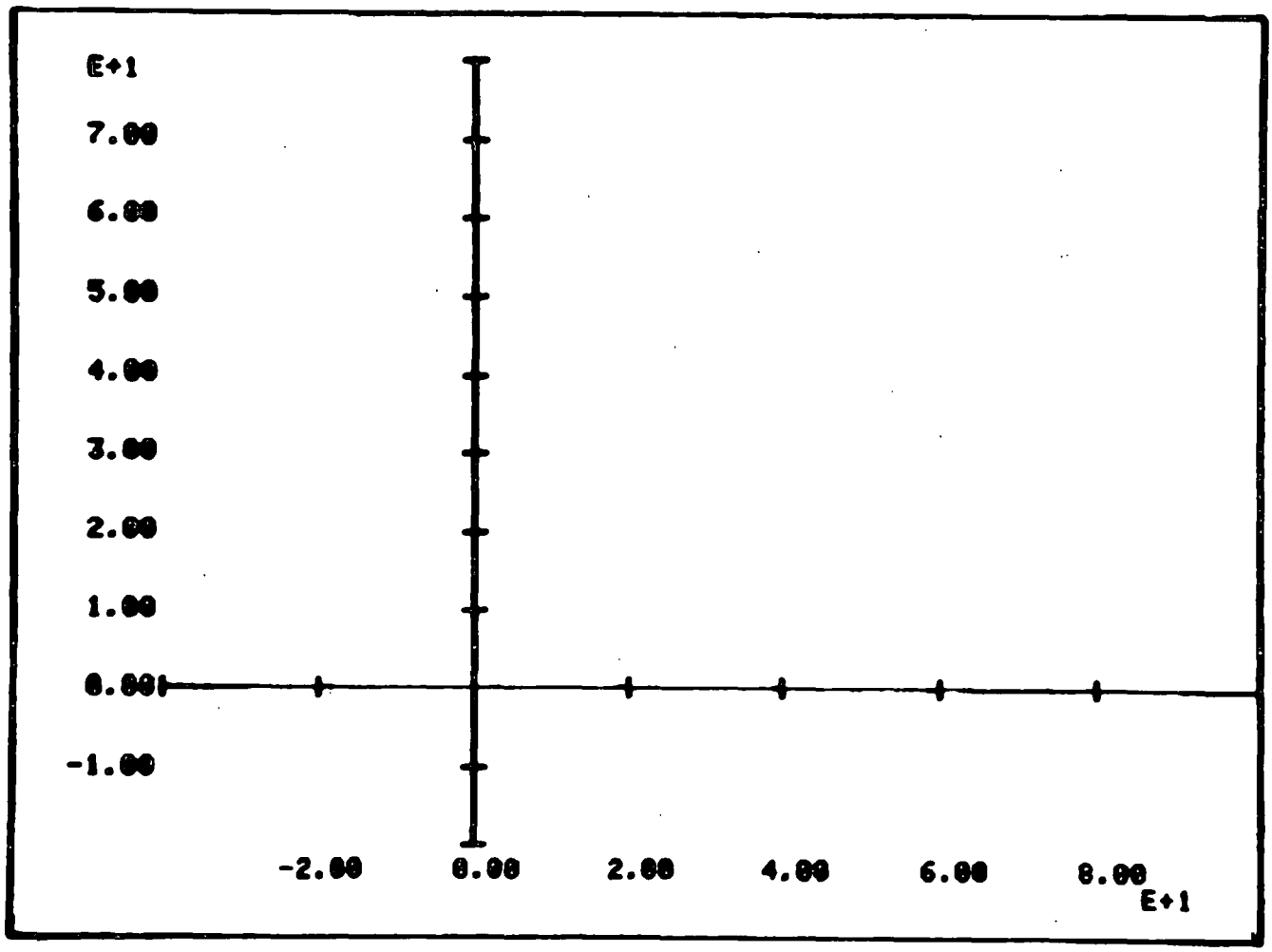
Note: The min, max values in G are changed by this program. When this subroutine is finished plotting, WINDOW has been sent to

WINDOW G(1), G(2), G(5), G(6)

Be sure to call this subroutine BEFORE plotting your data!

TITLE

NEAT TICS AND AXIS LABELING



| TITLE | |
|---|---|
| NEAT TICS AND AXIS LABELING | |

VARIABLE MAP

| VARIABLE | TAPE FILE | USAGE |
|---|---|---|
| A$ | | Move label outside viewport |
| B$ | | Move E notation outside viewport |
| G(1) | | Minimum data value, X axis |
| G(2) | | Maximum data value, X axis |
| G(3) | | Tic interval, number intervals desired, X axis |
| G(4) | | Scratch - label X axis |
| G(5) | | Minimum data value Y axis |
| G(6) | | Maximum data value Y axis |
| G(7) | | Tic interval, number intervals desired, Y axis |
| G(8) | | Scratch, label Y axis |
| V | | Scratch |
| X | | Scratch |
| X1 | | Scratch |
| X2 | | Scratch |
| Y | | Adjust label |
| Y1 | | E value |
| Y2 | | Scratch |

Tektronix
COMMITTED TO EXCELLENCE

# DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

| TITLE | |
|---|---|
| Linear Axis Labeling Routine | EQUIPMENT AND OPTIONS REQUIRED |
| ORIGINAL DATE: October, 1976 / REVISION DATE: | 8K |
| AUTHOR: Steven Den Beste, Tektronix, Inc., Wilsonville, OR | PERIPHERALS |

ABSTRACT

Files: 1 ASCII Program

Statements: 196

This program is a subroutine designed to be used with a user program. The subroutine generates an L-shaped axis, with tics and labels, covering any plot range, and places it anywhere on the screen. It requires 10 input variables and passes back 8 of them to describe the plot exactly.

All labels are four characters, including a decimal point and a sign (if negative).

For orientation, a grid of points is generated within the plottable areas. A point is placed at the intersection of any two intersections and at the intersection of any tic and zero, if zero is within range. (This is optional.)

An example program is included.

TITLE

LINEAR AXIS LABELING ROUTINE

## MAN/MACHINE INTERFACE

This routine is configured as a subroutine. 10 specific variables are preset by the user, the subroutine is called, and returns after generating the axis.

The preset variables are:

| | | | | | |
|---|---|---|---|---|---|
| V1 | – | MIN | X | VIEWPORT | (in GDU's |
| V2 | – | MAX | X | VIEWPORT | |
| V3 | – | MIN | Y | VIEWPORT | |
| V4 | – | MAX | Y | VIEWPORT | |

(These specify exactly where on the screen the plot lies)

| | | | | | |
|---|---|---|---|---|---|
| W1 | – | MIN | X | WINDOW | (in user data units) |
| W2 | – | MAX | X | WINDOW | |
| W3 | – | MIN | Y | WINDOW | |
| W4 | – | MAX | Y | WINDOW | |

(These specify the range of the plot)

| | | |
|---|---|---|
| T1 | – | MINIMUM NUMBER OF TICS IN X DIRECTION |
| T2 | – | MINIMUM NUMBER OF TICS IN Y DIRECTION |

## METHODOLOGY

This subroutine will steal some of the allotted viewport for the axis and labels. The coordinates of the actual plottable area within the axis is returned in V1, V2, V3, and V4.

The triplets W1, W2, T1, and W3, W4, T2 are processed separately. The low and high limits and number of tics are adjusted so that tic intervals are 1*10↑N, 2*10↑N, 2.5*10↑N or 5*10↑N. The adjusted window values are passed back in W1, W2, W3, and W4. T1 and T2 are not changed.

## PROGRAM LIMITATIONS

All labels are 4 characters, including a sign (if negative) and a decimal point. If the significant digits of the label are not within those 4 characters, the labels will look the same.

## OPERATING HINTS

This routine will set the VIEWPORT and WINDOW to describe the plottable area before returning. If it is necessary to change the WINDOW or VIEWPORT, they can be reset by:

or

```
WINDOW W1, W2, W3, W4
VIEWPORT V1, V2, V3, V4
```

The internal grid of points can be eliminated by deleting lines 6150 to 6300.

This subroutine is non-destructive. It can be called several times, with different sets of parameters, without reloading.

TITLE

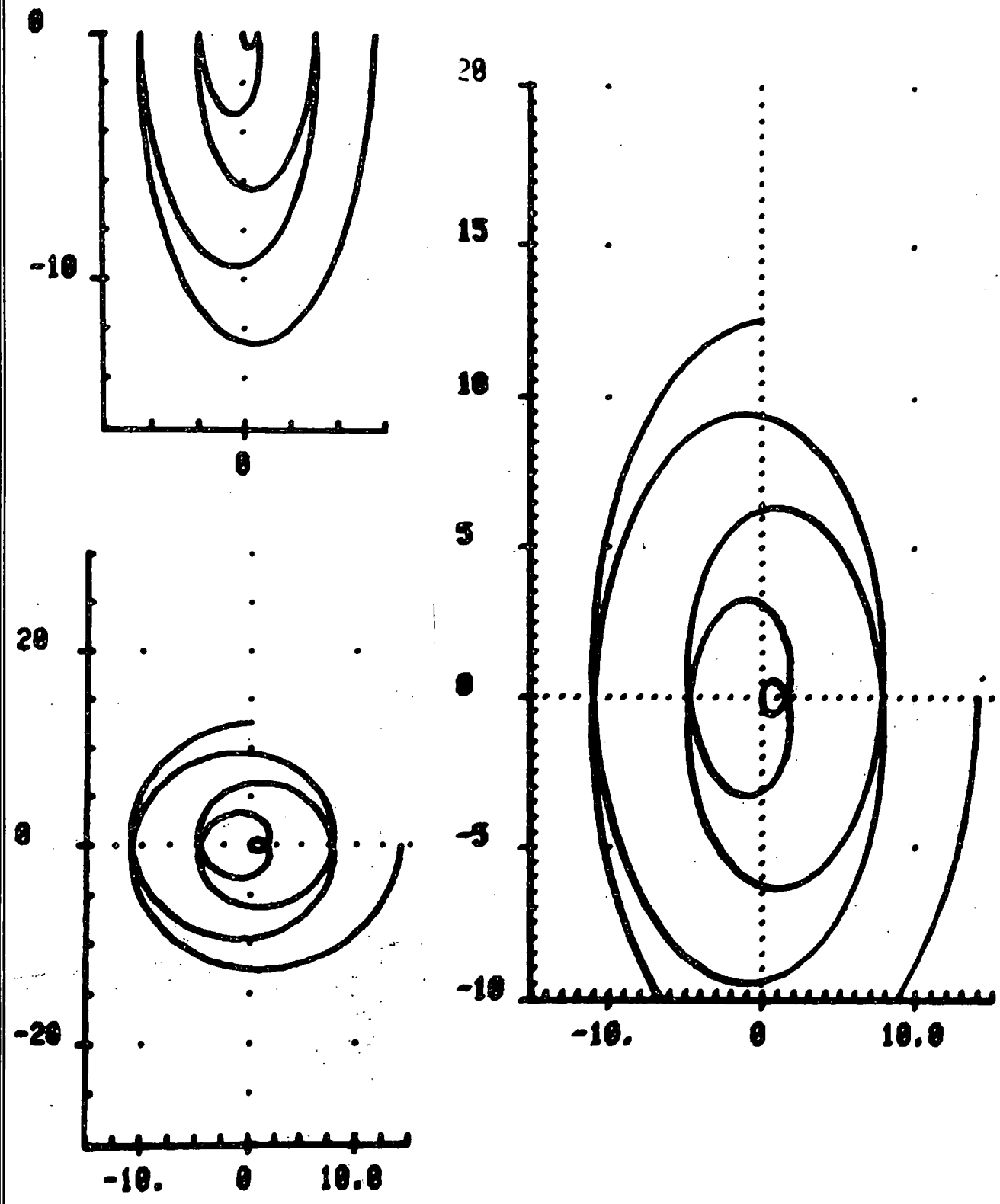
LINEAR AXIS LABELING ROUTINE

VARIABLE MAP

| VARIABLE | TAPE FILE | USAGE |
|---|---|---|
| T3 (1) | | Increment in X direction |
| T3 (2) | | Increment in Y direction |
| T3 (5) | | Factor for X direction |
| T3 (7) | | Factor for Y direction |
| T3 (10) | | Factor (passed back from sub 6340) |
| T3 (12) | | Range of plot (passed to sub at 6340) |
| T3 (13) | | Number of tics (passed to sub 6340) |
| T3 (14) | | Value of increment (passed back from sub 6340) |
| T3 (15) | | Min WINDOW (Passed to sub 6340) |
| T3 (16) | | Max WINDOW (passed to sub 6340) |
| T3 (20) | | Tics per label (passed to and from sub at 6630) |
| T3 (21) | | Current label value (within for-loop) |
| T3 (23) | | Value of increment (passed to sub at 6630) |
| T3 (24) | | Working variable |
| T3 (25) | | Working variable |
| T3 (26) | | Window value of lowest X label |
| T3 (27) | | Window value of highest X label |
| T3 (28) | | Increment of labels X |
| T3 (29) | | Window value of lowest Y label |
| T3 (30) | | Window value of highest Y label |
| T3 (31) | | Increment of labels Y |
| T4 | | FOR-LOOP VARIABLE |
| T5 | | FOR-LOOP VARIABLE |
| X | | USED TO GENERATE LABELS |
| Y | | USED TO GENERATE LABELS |

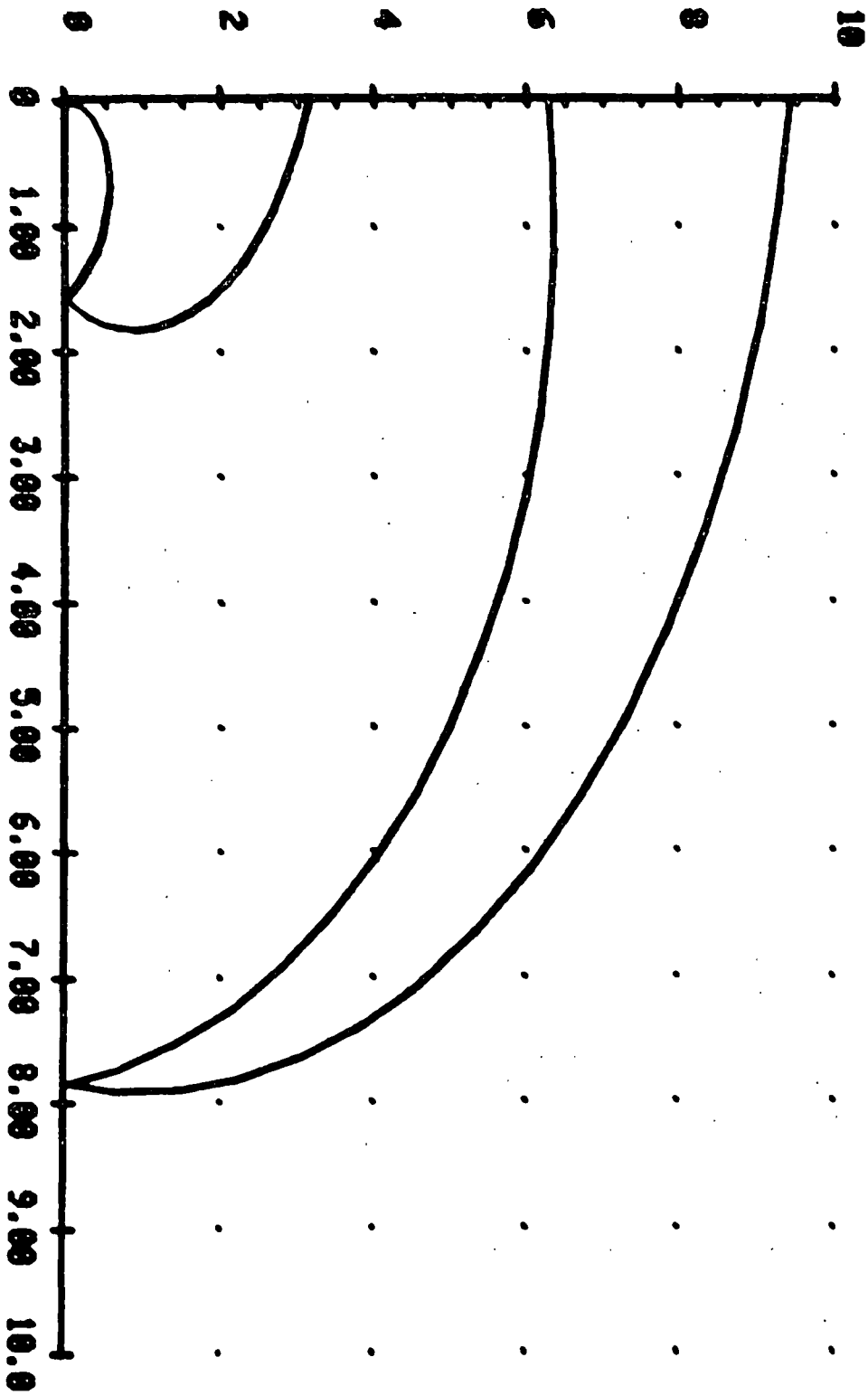
TITLE

LINEAR AXIS LABELING ROUTINE



TITLE

LINEAR AXIS LABELING ROUTINE



Tektronix
COMMITTED TO EXCELLENCE

# DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

| TITLE | | |
|---|---|---|
| Hexadecimal Program | | |
| ORIGINAL DATE | REVISION DATE | EQUIPMENT AND OPTIONS REQUIRED<br>16K |
| AUTHOR Marv Abe<br>Tektronix, Inc.<br>Wilsonville, OR | | PERIPHERALS |

ABSTRACT

Files: 1 ASCII Program

Statements: 360

This routine allows the user to perform miscellaneous hexadecimal functions using the 4050. Each one of the routines is called through the User-Definable Keys.

Conversion routines are:

Decimal Values to Hex Representation

Hex Values to Decimal Representation

RAD40 to ASCII*

ASCII to RAD40 Representation*

Hex # as Bit Pattern

Arithmetic functions are:

Hexadecimal Subtraction

Hexadecimal Addition

Both functions are provided in cumulative form and add and subtract from some constant value.

The different routines prompt the user for the required input and most always terminate on a carriage return.

---

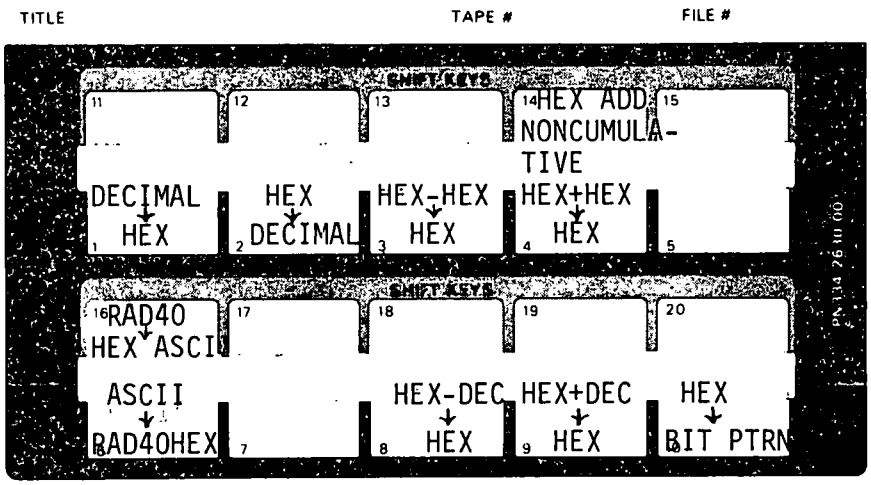
*For TEKTRONIX 4081 System

TITLE

HEXADECIMAL PROGRAM

OPERATING INSTRUCTIONS:

Enter file from tape and type RUN. The program is controlled by the function keys.




| KEY | FUNCTION |
|---|---|
| 1 | Conversion from decimal to hex |
| 2 | Conversion from hex to decimal |
| 3 | Hex subtraction routine |
| 4 | Hex addition routine |
| 6 | Convert ASCII input to RAD40 representation |
| 8 | Decimal subtraction of hex |
| 9 | Decimal addition of hex |
| 10 | Convert input hex to output bit pattern |
| 14 | Hex addition to relative base |
| 16 | Hex RAD40 representation to ASCII |

TITLE

HEXADECIMAL PROGRAM

B9

T9

T3

T2

T1

J1

VARIABLES:

A$
B$
C$
D$
F$
N$
T$

A C I J M N T

TITLE

HEXADECIMAL PROGRAM

```
DECIMAL to HEXADECIMAL conversion a <cr> will terminate

Decimal value ?1
              HEX eqiv: 1
Decimal value ?16
              HEX eqiv: 10
Decimal value ?32
              HEX eqiv: 20
Decimal value ?256
              HEX eqiv: 100
Decimal value ?511
              HEX eqiv: 1FF
Decimal value ?512
              HEX eqiv: 200
Decimal value ?32767
              HEX eqiv: 7FFF
Decimal value ?65000
              HEX eqiv: FDE8
Decimal value ?
```

Tektronix
COMMITTED TO EXCELLENCE

# DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

| TITLE | | |
|---|---|---|
| Disk Directory | | |
| ORIGINAL DATE | REVISION DATE | EQUIPMENT AND OPTIONS REQUIRED |
| May, 1980 | | 8K |
| AUTHOR | Comalco Aluminium Ltd. | PERIPHERALS |
| Nick Ogbourne | George Town, Tasmania | 4907 File Manager |

ABSTRACT

Files: 1 Program

Statements: 193

Disk Directory maintains a directory of up to 50 disk programs and controls access to and execution of those programs.

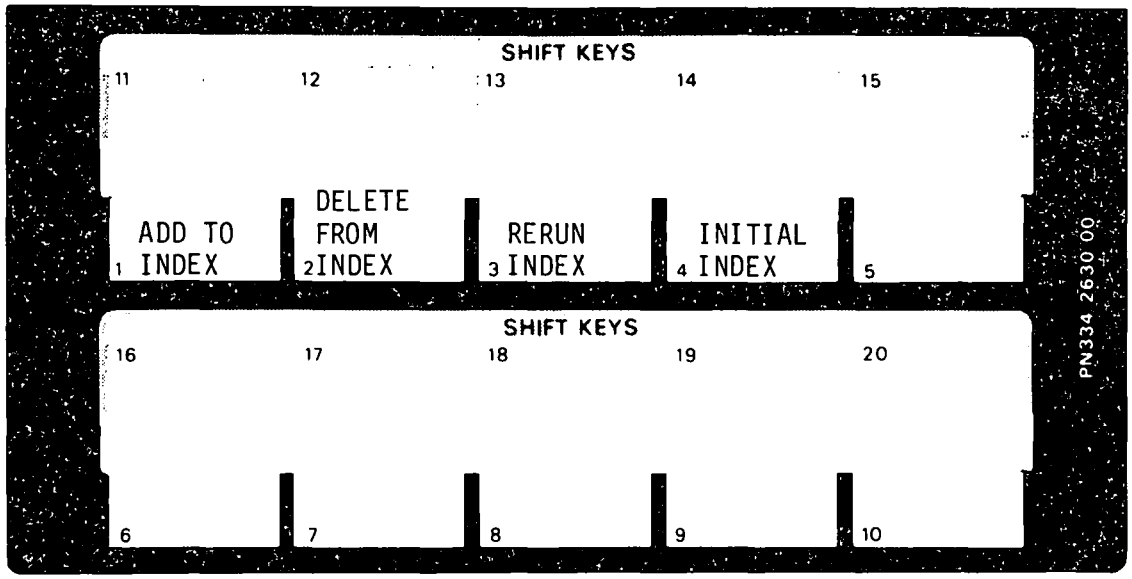
Disk Directory creates and maintains an index file. This index file includes the file identifier, program # (sequenced in the order entered), and user-input information (up to 44 characters) about the file. Programs may be added or deleted through the User-Definable Keys.

Disk Directory reads the index file, prints a directory of the files (multipage if necessary) and prompts for the program of your choice. It will warn you if a selected file is a binary data file. Any other type of file it will attempt to load.

TITLE

Disk Directory

TITLE Disk Directory TAPE # FILE #




ADD TO INDEX - Prompts for filename and information and adds to the INDEX file.

DELETE FROM INDEX - Deletes the requested filename from the index and re-sequences everything, if necessary.

RERUN INDEX - RUNs the program selection routine.

INITIAL INDEX - Initializes the INDEX file if none exists.

TITLE

Disk Directory

PRELIMINARY OPERATING INSTRUCTIONS

Transfer Disk Directory from PROGRAMMING AIDS T2 tape to your new (formatted) disk:

Step 1. Insert PROGRAMMING AIDS T2 tape in your 4050 system.

```
FIND 15
OLD
```

Step 2. Insert your new disk in your 4907 File Manager and MOUNT.

```
SAVE "@INDEX/PROGRAM"
```

NOTE: If your disk unit is not Ø, change statement 220.

OPERATING INSTRUCTIONS

Load and execute the program from disk:

```
OLD "@INDEX/PROGRAM"
RUN
```

Creating an Index

If an INDEX file has not been created, the message:

```
NO INDEX EXISTS!
```

will be printed.

Press User-Definable Key 4 to create the INDEX file.

If an INDEX filehas been created, the message

```
INDEX FILE @INDEX/INDEX ALREADY EXISTS!
```

will be printed.

TITLE

Disk Directory

Adding to an Index

When an INDEX is empty, the message:

INDEX EMPTY!

will be printed when the program is RUN.

Press User-Definable Key 2 to input program names and information.

CAUTION: The program name must meet FILE IDENTIFIER requirements, i.e., it must be no longer than 10 characters, must be totally alphanumeric--no spaces, no other characters--and the first character must be alpha. If it doesn't meet these requirements, the program will fail. To re-start, press UDK #2 again.

```
Enter program name = TESTPROG
There currently is no file called
TESTPROG
Is that OK: Y
Enter program description (Max 54 characters)
A program to test a 468/4050 system
```

TITLE

Disk Directory

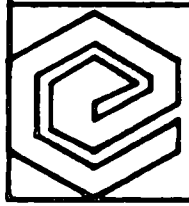
## Loading Directory

Keying in RUN or pressing UDK #3 will print the current directory. To choose your program, input the number.

Or, to end or return to another function, simply press RETURN without a response.

DISK DIRECTORY. 19-MAY-81 08:27:07

| FILE | NUMBER | FUNCTION |
|---|---|---|
| TESTPROG | 1 | A program to test a 468/4050 system |
| FLOW/DIAGR | 2 | Flow Diagrammer for the 4907 File Manager |
| FLOW/CHART | 3 | Flow Charting for a Program on Tape |



Select program =



000-6405-02

TITLE

Disk Directory

## Deleting a Program

Press UDK #2 to delete a program from the directory.

The program will print the file name and ask you to confirm. Responding with a Y will delete the program; an N will leave it in the INDEX.

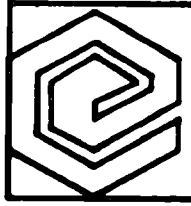
```
Delete which item = 1

This entry is for program
TESTPROG
Description :-
A program to test a 468/4050 system
OK y
```

## VARIABLES USED

| | |
|---|---|
| P$ (10) | File Name |
| D$ (55) | File Description |
| LØ (44) | Company Logo |
| DØ | Screen Address |
| N1 | Index Record # (actual record N1+1) |
| NØ | Total Records in Index |
| S$ (72) | Scratch |
| F$ (200) | File Status Message |
| Q$ (1) | Scratch |
| X$ | Prompt |

DISK DIRECTORY. 19-MAY-81 08:46:43

| FILE | NUMBER | FUNCTION |
|---|---|---|
| FLOW/DIAGR | 1 | Flow Diagrammer for the 4907 File Manager |
| FLOW/CHART | 2 | Flow Charting for a Program on Tape |



Select program =

Tektronix®
COMMITTED TO EXCELLENCE

# DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

| TITLE | | |
|---|---|---|
| File Identifier | | |
| ORIGINAL DATE | REVISION DATE | EQUIPMENT AND OPTIONS REQUIRED |
| July, 1979 | | 8K |
| AUTHOR | Comalco Aluminium Ltd. George Town, Tasmania | PERIPHERALS |
| Nick Ogbourne | | 4907 File Manager |

ABSTRACT

Files: 1 ASCII Program

Statements: 111

The program is a subroutine that compiles a file identifier which will comply with the 4907 File Manager rules.

The program prompts the user to select libraries to the selected level, up to level 4, including SYSLIB or SCRATCHLIB. Passwords for any or all libraries may be added.

Following library selection, file selection on the same basis occurs, plus the selection of a file extension.

The valid file name is then return in E$ and a flag, EØ, assumes a value of Ø if the file does not currently exist and 1 if it does currently exist.

TITLE

FILE IDENTIFIER

OPERATING INSTRUCTIONS

Append the routine at a suitable point in your program and address it via a suitable GOSUB.

The routine prompts the user for the Level #1 library, allowing the selection of SYSLIB by entering a '$' and SCRATCHLIB by pressing <RETURN>.

The valid file name is returned at line 680 in E$, together with the flag EØ defining the current presence or absence of the file on the currently mounted disc.

All variables except E$ and EØ are scratch. The user will need to ensure that this applies prior to calling the routine.

Variable E$ should be dimensioned to the maximum FI requirement prior to calling the routine, i.e.

```
        $)
or      @)      + 4 * 22 + 26  =  115
         )

              Libraries  File
```

TITLE

FILE IDENTIFIER

EXAMPLE

In the example, Level #1 library name 'USERLIB' is entered. The routine checks for validity, advising the user if the entry is incorrect and prompting for correction.

A similar procedure occurs for PASSWORD selection, a blank <RETURN> indicating no password required.

Entering a '/' when prompted for a level library will lead to file name selection, the procedure for file name and password being as above, plus selection of an extension if required.

```
Level # 1 library.Maximum 10 characters.
Press <RETURN> for SCRATCHLIB,enter `$' for SYSLIB.
Enter name for USERLIB.
USERLIB
Password.Maximum 10 characters.Press <RETURN> if not required.
LIBPASS
Level # 2 library.Maximum 10 characters.
Enter `/' to select file. LEVEL2
Password.Maximum 10 characters.Press <RETURN> if not required.

Level # 3 library.Maximum 10 characters.
Enter `/' to select file. /
File name.(Maximum 10 characters.) = FILENAME
Password.Maximum 10 characters.Press <RETURN> if not required.
FILEPASS
Extension.Maximum 4 characters.Press <RETURN> if not required.
EXTE
FILE = @USERLIB:LIBPASS/LEVEL2/FILENAME:FILEPASS.EXTE FLAG = 0
```

TITLE

FILE IDENTIFIER

TITLE

FILE IDENTIFIER

| Variable | Use to Store | Type |
|---|---|---|
| EØ | Flag file presence or absence | Simple |
| E1 | Scratch | Simple |
| E2 | Scratch | Simple |
| E3 | Scratch | Simple |
| E4 | Scratch | Simple |
| E5 | Scratch | Simple |
| E$ | File Name | String |
| L$ | Scratch | String |
| Q$ | Scratch | String |
| S | Scratch | Simple |
| S$ | Scratch | String |