PROGRAMMING AIDS T1
062-5971-01

PROGRAMMING AIDS T1 is a tape collection of 16 programs to aid you in creating or dissecting a 4050 BASIC program. Employ these routines to produce your overlays, structure program flow, track variables, convert from FORTRAN to BASIC, draw flow diagrams and aid you in other programming techniques. The individual abstracts describe each program.

Two of the programs maintain their own data files and must be transferred to other tapes before execution. The documentation for each gives specific instructions for accomplishing the transfers.

Title/ Previous Abstract #	
Overlay Drawing Program	51/00-9537/0
Enhanced Program Listings	51/00-8044/0
REMark Outliner	51/00-8035/0
Tape Directory	51/00-8026/0
List Program's Variables	51/00-8002/0
Cross-Reference & List Program Variables	51/00-8004/0
Device Address Adding Program	51/00-8032/0
Log/Linear Axis Labeling Routine	51/00-9504/0
Dashed Lines	51/00-9508/1
Calendar Routines (7-Day Week)	51/00-0902/0
Calendar Routines (5-Day Week)	51/00-0903/0
FORTRAN to BASIC Converter	51/00-7003/0
Flow Diagrammer (tape)	51/00-8015/0
Flow Diagrammer (disk)	51/07-8015/1
Segmented Data Base	51/07-9522/0
Windowing Routines	51/07-9522/0

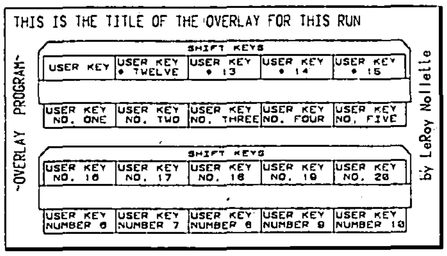
Program 1

Title: **Overlay Drawing Program**
Author: LeRoy Nollette
Tektronix, Inc.
Wilsonville, OR
Memory Requirement: 16K
Peripherals: 4662 Plotter
Statements: 250
Files: 1 ASCII Program
2 ASCII Data (Sample Overlays)
Optional-Pre-MARked data files

The program draws an overlay on the 4662 Plotter that can be cut out and placed over the User-Definable Keys. Key descriptions may be entered from the keyboard. Data may be saved on a pre-MARked data file and re-drawn at a later date.

The program may be modified (one line of code) to draw a large copy of the overlay and then reduce it on a copy machine having reduction capabilities.

By changing one line of code the user may preview the overlay on the screen.



Program 2

Title: **Enhanced Program Listings**
Author: Tim Giesbers
Tektronix, Inc.
Beaverton, OR
Memory Requirement: 8K
Peripherals: Optional—4641 Printer
Statements: 144
Files: 1 ASCII Program

The program will list any ASCII program file, or consecutive files, stored on tape.

The list can be either to the 4050 screen or a 4641 Printer. If the list is to the screen, copies may be made automatically on a 4631 Hard Copy Unit.

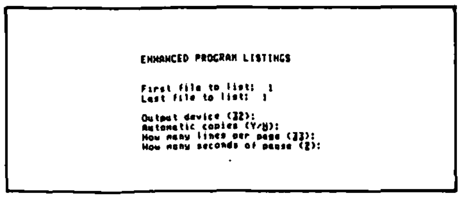
The listing includes file numbers and the length of each file is given in bytes at the end of the listing.

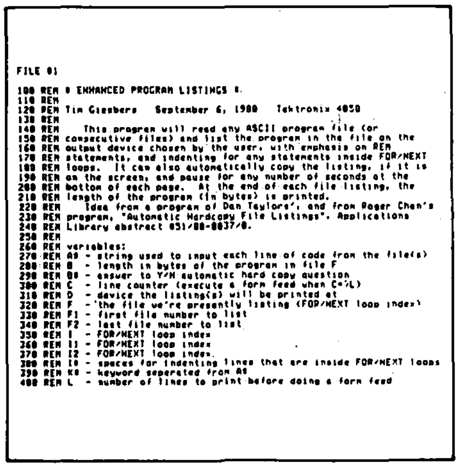
Statements inside FOR/NEXT loops are indented, and REM statements are separated from other program lines by a blank line for emphasis.

New pages are automatic with the user specifying the number of lines per page and the length of the pause between pages. There is no provision for wraparound or truncation of a line which is longer than the width of the printer paper.

User input:

First file number
Last file number
Output device number
Automatic copies Yes/No
How many lines per page
How many seconds of pause


```
ENHANCED PROGRAM LISTINGS

First file to list:  1
Last file to list:  1

Output device (32):
Automatic copies (Y/N):
How many lines per page (33):
How many seconds of pause (2):
```


```
FILE 01

100 REM * ENHANCED PROGRAM LISTINGS *
110 REM
120 REM Tim Giesbers   September 6, 1980   Tektronix 4050
130 REM
140 REM    This program will read any ASCII program file (or
150 REM consecutive files) and list the program in the file on the
160 REM output device chosen by the user, with emphasis on REM
170 REM statements, and indenting for any statements inside FOR/NEXT
180 REM loops.  It can also automatically copy the listing, if it is
190 REM on the screen, and pause for any number of seconds at the
200 REM bottom of each page.  At the end of each file listing, the
210 REM length of the program (in bytes) is printed.
220 REM    Idea from a program of Dan Taylors', and from Roger Chen's
230 REM program, "Automatic Hardcopy File Listings", Applications
240 REM Library abstract #51/00-8037/0.
250 REM
260 REM variables:
270 REM A$ - string used to input each line of code from the file(s)
280 REM B  - length in bytes of the program in file F
290 REM B$ - answer to Y/N automatic hard copy question
300 REM C  - line counter (execute a form feed when C=L)
310 REM D  - device the listing(s) will be printed at
320 REM F  - the file we're presently listing (FOR/NEXT loop index)
330 REM F1 - first file number to list
340 REM F2 - last file number to list
350 REM I  - FOR/NEXT loop index
360 REM I1 - FOR/NEXT loop index
370 REM I2 - FOR/NEXT loop index
380 REM I$ - spaces for indenting lines that are inside FOR/NEXT loops
390 REM K$ - keyword seperated from A$
400 REM L  - number of lines to print before doing a form feed
```

Program 3

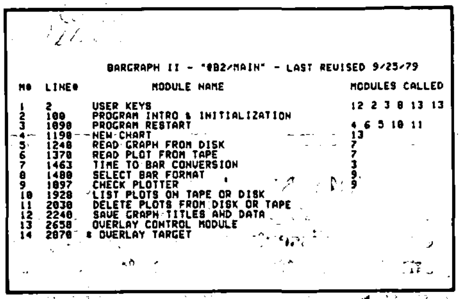
Title: **REMark Outliner**
Author: Mallory M. Green
U.S. Dept. of HUD
Washington, D.C.
Memory Requirement: 8K
Statements: 141
Files: 1 ASCII Program

REMark Outliner is intended as a tool for the programmer who writes a structured program. It inputs a structured ASCII program and prints out a program outline. The outline includes subroutine names, line numbers and flow between subroutines.

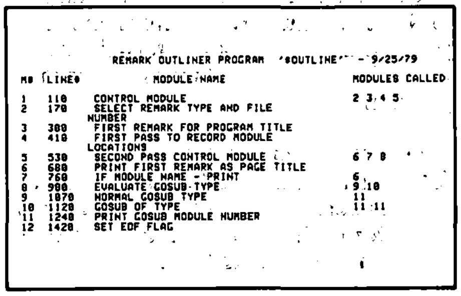
The following programming techniques are required for REMark Outliner to work effectively.

1. Subroutines make up the program with GOSUB or GOSUB OF statements controlling program flow.
2. Subroutines begin with REMark statements describing the subroutine's function. These REMark statements are separated from other REMark statements by special characters; i.e., REM * or REM / and so on.
3. Hierarchical subroutines.
4. Program's name contained in first program REMark.

REMark Outliner uses the special REMark statement to identify the modules and it traces program flow only through GOSUB or GOSUB OF statements. It makes two passes through a program: the first pass creates a table of subroutine locations; the second pass prints the program outline.



```
          BARGRAPH II - "@B2/MAIN" - LAST REVISED 9/25/79
N0  LINE#          MODULE NAME                       MODULES CALLED

1   2       USER KEYS                                12 2 3 8 13 13
2   100     PROGRAM INTRO & INITIALIZATION
3   1090    PROGRAM RESTART                          4 6 5 10 11
4   1190    NEW CHART                                13
5   1240    READ GRAPH FROM DISK                     7
6   1370    READ PLOT FROM TAPE                      7
7   1463    TIME TO BAR CONVERSION                   3
8   1480    SELECT BAR FORMAT                        9
9   1897    CHECK PLOTTER                            9
10  1928    LIST PLOTS ON TAPE OR DISK
11  2030    DELETE PLOTS FROM DISK OR TAPE
12  2240    SAVE GRAPH TITLES AND DATA
13  2658    OVERLAY CONTROL MODULE
14  2870  & OVERLAY TARGET
```



```
          REMARK OUTLINER PROGRAM  '@OUTLINE' - 9/25/79
N0  LINE#          MODULE NAME                       MODULES CALLED

1   110     CONTROL MODULE                           2 3 4 5
2   170     SELECT REMARK TYPE AND FILE
           NUMBER
3   300     FIRST REMARK FOR PROGRAM TITLE
4   410     FIRST PASS TO RECORD MODULE
           LOCATIONS
5   530     SECOND PASS CONTROL MODULE               6 7 8
6   680     PRINT FIRST REMARK AS PAGE TITLE
7   760     IF MODULE NAME - PRINT                   6
8   900     EVALUATE GOSUB TYPE                      9 10
9   1070    NORMAL GOSUB TYPE                        11
10  1120    GOSUB OF TYPE                            11 11
11  1240    PRINT GOSUB MODULE NUMBER
12  1420    SET EOF FLAG
```

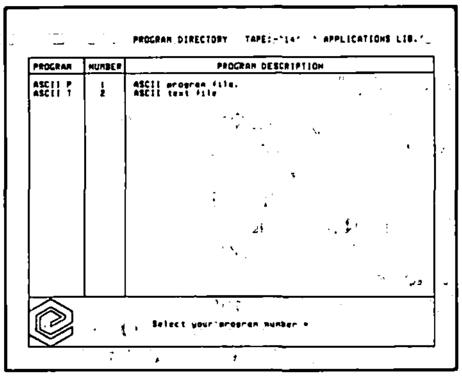
Program 4

Title: **Tape Directory**
Author: Nick Ogbourne
Comalco Aluminum Ltd.
George Town, Tasmania,
Australia
Memory Requirement: 8K
Peripherals: Optional-4051R06 Editor ROM
Statements: 90
Files: 1 ASCII Program
1 ASCII Text

The program, located as the first ASCII program file on a tape, operates using the AUTOLOAD, provides a tape 'directory' multipage if necessary, and controls access to, and execution of program files.

The user creates and maintains an 'index' in File 2 (ASCII) which provides file number, program name and program description to the 'directory' program. File 2 may be updated using the 4051R06 Editor ROM or a simple BASIC program. (An example of the index is included.)

It is not necessary to specify to the directory the type of the program (ASCII or Binary). Programs not required to be accessed by the directory, data files and text files may be recorded in file 2, providing a rapid means of 'TLIST'ing a tape.

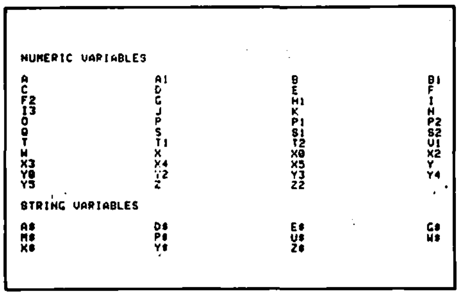


```
             PROGRAM DIRECTORY   TAPE:-'14'  ' APPLICATIONS LIB.'
PROGRAM  | NUMBER | PROGRAM DESCRIPTION
ASCII P  |   1    | ASCII program file.
ASCII T  |   2    | ASCII text file

                Select your program number »
```

Program 5

Title: **List Program's Variables**
Author: Brian Diehm
Tektronix, Inc.
Wilsonville, OR
Memory Requirement: 8K
Statements: 105
Files: 1 ASCII Program

This program reads a tape file containing a BASIC ASCII program and prints an alphabetized list of all the variables used in that program. The program first asks the user which tape file contains the program to be analyzed. Then, after reading the file, two alphabetized lists of variables are printed on the screen. The first list gives all of the numeric variables' names, the second list gives all the string variables' names. Provision is made to allow processing of several files, combining the results into one list. The files do not have to be sequential but operator input is required for each one as they are processed. Listing of files as they are processed is optional.

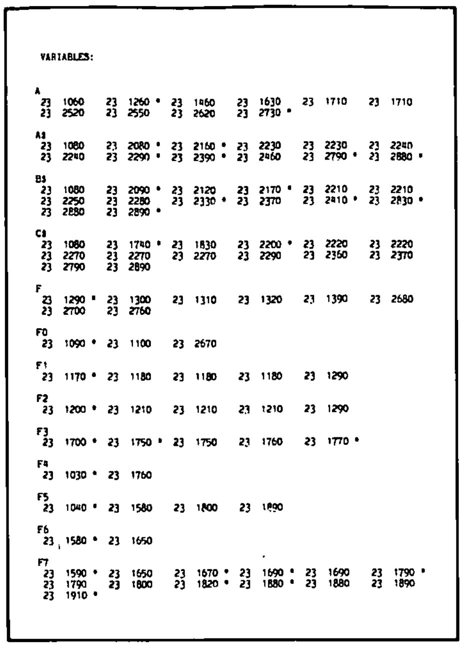


```
NUMERIC VARIABLES

A          A1         B          B1
C          D          E          F
F2         G          H1         I
I3         J          K          N
O          P          P1         P2
Q          S          S1         S2
T          T1         T2         U1
W          X          X0         X2
X3         X4         X5         Y
Y0         Y2         Y3         Y4
Y5         Z          Z2

STRING VARIABLES

A$         D$         E$         G$
M$         P$         U$         W$
X$         Y$         Z$
```

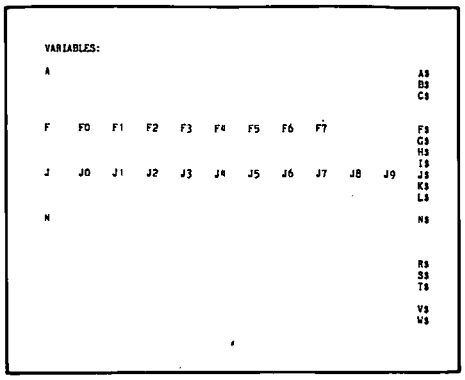
Program 6

Title: **Cross-Reference & List Program Variables**
Author: Dan Taylor
Tektronix, Inc.
Wilsonville, OR
Memory Requirement: 16K
Peripherals: Optional-4641 Printer
Statements: 192
Files: 1 ASCII Program

This program reads a BASIC ASCII program from tape and produces an alphabetized table of the variables used in the program. It also produces a cross-reference for each variable used which shows the BASIC line numbers where that variable is used and indicates if a value is assigned to that variable in that line of code. The BASIC program may be stored on multiple sequential tape files. Three variables must be changed to output to 4641 Printer.



```
VARIABLES:

A
 23  1060    23  1260 *  23  1460    23  1630    23  1710    23  1710
 23  2520    23  2550    23  2620    23  2730 *

A$
 23  1080    23  2080 *  23  2160 *  23  2230    23  2230    23  2240
 23  2240    23  2290 *  23  2390 *  23  2460    23  2790 *  23  2880 *

B$
 23  1080    23  2090 *  23  2120    23  2170 *  23  2210    23  2210
 23  2250    23  2280    23  2330 *  23  2370    23  2410 *  23  2830 *
 23  2880    23  2890 *

C$
 23  1080    23  1740 *  23  1830    23  2200 *  23  2220    23  2220
 23  2270    23  2270    23  2270    23  2290    23  2360    23  2370
 23  2790    23  2890

F
 23  1290 *  23  1300    23  1310    23  1320    23  1390    23  2680
 23  2700    23  2760

F0
 23  1090 *  23  1100    23  2670

F1
 23  1170 *  23  1180    23  1180    23  1180    23  1290

F2
 23  1200 *  23  1210    23  1210    23  1210    23  1290

F3
 23  1700 *  23  1750 *  23  1750    23  1760    23  1770 *

F4
 23  1030 *  23  1760

F5
 23  1040 *  23  1580    23  1800    23  1890

F6
 23  1580 *  23  1650

F7
 23  1590 *  23  1650    23  1670 *  23  1690 *  23  1690    23  1790 *
 23  1790    23  1800    23  1820 *  23  1880 *  23  1880    23  1890
 23  1910 *
```



```
VARIABLES:

A                                                              A$
                                                               B$
                                                               C$

F    F0   F1   F2   F3   F4   F5   F6   F7                     F$
                                                               G$
                                                               H$
                                                               I$
J    J0   J1   J2   J3   J4   J5   J6   J7   J8   J9           J$
                                                               K$
                                                               L$

N                                                              N$


                                                               R$
                                                               S$
                                                               T$

                                                               V$
                                                               W$
```

Program 7

Title: **Device Address Adding Program**
Author: Jan Broenink
Tektronix International Inc.
European Marketing Centre
Amstelveen, Holland
Memory Requirement: 16K
Peripherals: 4924 Tape Drive
Optional-4641/4642 Printer
Statements: 402
Files: 1 ASCII Program

The program reads a tape file from the 4924 containing a 4050 BASIC program in ASCII format and updates the program by adding a device address (for graphics and alphanumerics) to output statements without a device address or with address 32 (without a secondary address) and saves the updated file to the tape in the internal tape drive.

The program searches for the following output statements without a device address or with address 32:

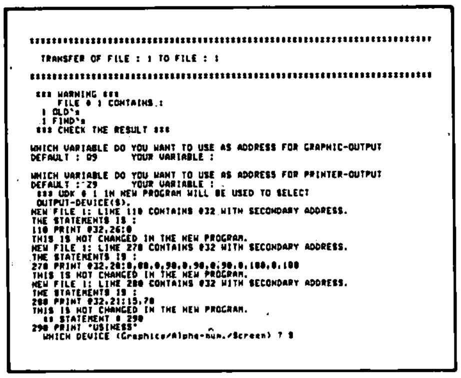
PRINT	MOVE
LIST	DRAW
RMOVE	AXIS
RDRAW	GIN

and will automatically or with user interaction add a device address. Interaction allows the user to define more than one output address within a program. For instance, user instructions may be directed to the screen while graphs may be directed to the plotter.

If APPEND, OLD, and FIND's are used in a program, a message is given how many APPEND's, etc., have been traced. In some cases the user has to check the result if the new program is still usable in relation with other program(s) or routine(s).

A routine is added to the original program to define a device address for graphic and alphanumeric output. An unused User Definable Key in the original program may be used to call this routine.

The original program may be stored on several sequential tape files.



```
***************************************************************************

  TRANSFER OF FILE : 1 TO FILE : 1

***************************************************************************

 *** WARNING ***
     FILE # 1 CONTAINS :
  1 OLD's
  1 FIND's
 *** CHECK THE RESULT ***

WHICH VARIABLE DO YOU WANT TO USE AS ADDRESS FOR GRAPHIC-OUTPUT
DEFAULT : 09        YOUR VARIABLE :

WHICH VARIABLE DO YOU WANT TO USE AS ADDRESS FOR PRINTER-OUTPUT
DEFAULT : 29        YOUR VARIABLE :
 *** UDK # 1 IN NEW PROGRAM WILL BE USED TO SELECT
 OUTPUT-DEVICE(S).
NEW FILE 1: LINE 110 CONTAINS @32 WITH SECONDARY ADDRESS.
THE STATEMENTS IS :
110 PRINT @32,26:0
THIS IS NOT CHANGED IN THE NEW PROGRAM.
NEW FILE 1: LINE 270 CONTAINS @32 WITH SECONDARY ADDRESS.
THE STATEMENTS IS :
270 PRINT @32,20:0,00,0,90,0,90,0,90,0,100,0,100
THIS IS NOT CHANGED IN THE NEW PROGRAM.
NEW FILE 1: LINE 280 CONTAINS @32 WITH SECONDARY ADDRESS.
THE STATEMENTS IS :
280 PRINT @32,21:15,70
THIS IS NOT CHANGED IN THE NEW PROGRAM.
  ** STATEMENT # 290
290 PRINT "USINESS"
  WHICH DEVICE (Graphics/Alpha-num./Screen) ? S
```

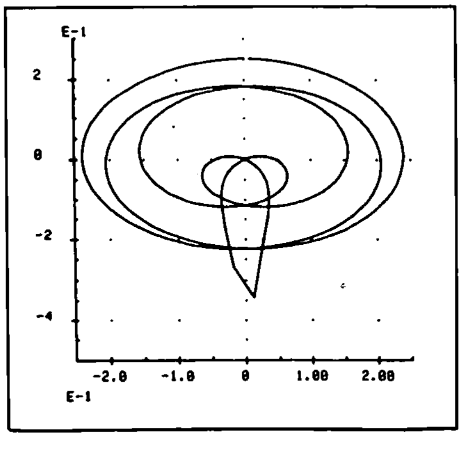
Program 8

Title: **Log/Linear Axis Labeling Routine**
Author: Steven Den Beste
Tektronix, Inc.
Wilsonville, OR
Memory Requirement: 16K
Statements: 281
Files: 1 ASCII Program

This program is a subroutine designed to be used with a user program. The subroutine generates an L-shaped axis with logarithmic or linear labeling on either axis, covering any range of positive values, and placed anywhere on the screen.

All labels are 4 characters, including a decimal point and a sign (if negative).

A pair of transformation functions are defined by the user before generating the plot.



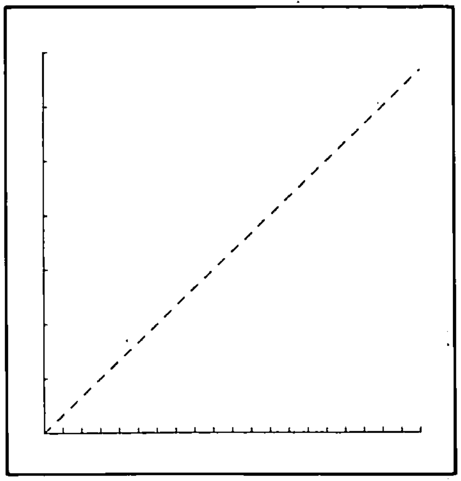
Program 9

Title: **Dashed Lines**
Author: Bob Ross
Tektronix, Inc.
Wilsonville, OR
Memory Requirement: 8K
Peripherals: Optional-4662 Plotter
Statements: 154
Files: 1 ASCII Program

Three subroutines draw dashed lines for:

1. A Y array with X values stepped linearly from a starting to an ending value;
2. Points stored in X and Y arrays;
3. A sequence of X and Y values.

The dashes are a constant length regardless of the viewport and window chosen. The dash length and ratio of dash to dash plus space are selectable. The line can start and end on a full dash or full space.



Program 10

Title: **Calendar Routines (7-Day Week)**
Author: Judy Peterman
Tektronix, Inc.
Wilsonville, OR
Memory Requirement: 8K
Statements: 200
Files: 1 ASCII Program

This program contains five calendar utility routines based on a seven-day week. Sunday through Saturday. They have been designed specifically for use in programs that calculate and graph financial and other business data, but can be used in any program that involves the collection or display of time related data. The routines:

1. Gives the date a specific number of time segments before or after a specific date.
2. Gives day number, week number, and day of the week of a specific date based on January 1, 1900.
3. Gives the number of time segments between two specific dates.
4. Verifies a date entry.
5. Unpacks a date.

All routines accommodate five time frames: days, weeks, months, quarters, and years. For example, if you are using days as the time segment in routine #1, 11/17/74 +2 yields 11/19/74; in weeks 11/17/74 +2 yields 12/1/74. The routines will not produce results prior to January 1, 1901.

The routine package comes with examples. Routines and examples require 7.9k bytes to run; the routines alone require 5.3k bytes.

Program 11

Title: **Calendar Routines (5-Day Week)**
Author: Judy Peterman
Tektronix, Inc.
Wilsonville, OR
Memory Requirement: 8K
Statements: 210
Files: 1 ASCII Program

This program contains five calendar utility routines based on the premise that a week is five days, Monday through Friday. The routines are the same as those found in program 10 the calendar routines for a 7-day week:

1. Date n time segments,
2. Date #, week #, and day of the week,
3. Time segments, between dates,
4. Date entry verify, and
5. Unpack date.

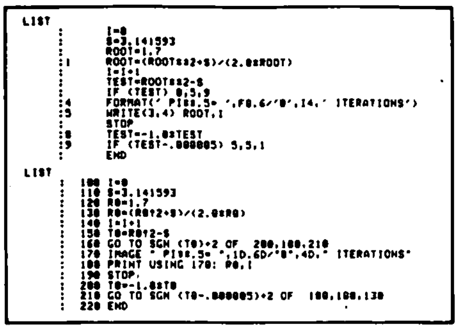
The routines with the examples take 8.3k bytes to run but the routines alone take only 5.6k bytes.

Program 12

Title: **FORTRAN to BASIC Converter**
Author: Mark R. Mehall
Tektronix, Inc.
Wilsonville, OR
Memory Requirement: 32K
Peripherals: 4924 Digital Tape Drive
4050R06 EDITOR ROM
Statements: 977
Files: 2 ASCII Program
Requires separate tape

This program is designed to convert FORTRAN to 4050 Series BASIC. The program is based on the USA Standard FORTRAN, X3.9-1966. The FORTRAN statement labels, variables and subroutine names are changed to their BASIC counterparts and remembered for references throughout the program. The majority of FORTRAN statements are changed into BASIC by this program. The statements that are not directly compatible are made into REMARK's and can be modified using the EDITOR ROM or the 4050 Series Line Editor. The FORTRAN statements: READ, WRITE, FORMAT, IF, GOTO, DO, DIMENSION, CALL, END, RETURN, STOP, SUBROUTINE, and CONTINUE are automatically changed to BASIC. The FORTRAN internal routines are also converted to the corresponding BASIC routines.

The program also prints tables of corresponding FORTRAN statement numbers to BASIC line numbers, FORTRAN variable names to BASIC variables, and FORTRAN subroutine names to BASIC Line numbers.



```
LIST
  :        I=0
  :        S=3.141593
  :        ROOT=1.7
  :1       ROOT=(ROOT**2+S)/(2.0*ROOT)
  :        I=I+1
  :        TEST=ROOT**2-S
  :        IF (TEST) 8,5,9
  :4       FORMAT(' PI**.5= ',F8.6/'0',I4,' ITERATIONS')
  :5       WRITE(3,4) ROOT,I
  :        STOP
  :8       TEST=-1.0*TEST
  :9       IF (TEST-.000005) 5,5,1
  :        END
LIST
  :   100 I=0
  :   110 S=3.141593
  :   120 R0=1.7
  :   130 R0=(R0↑2+S)/(2.0*R0)
  :   140 I=I+1
  :   150 T0=R0↑2-S
  :   160 GO TO SGN (T0)+2 OF  200,180,210
  :   170 IMAGE " PI↑.5= ",1D.6D/"0",4D," ITERATIONS"
  :   180 PRINT USING 170: R0,I
  :   190 STOP
  :   200 T0=-1.0*T0
  :   210 GO TO SGN (T0-.000005)+2 OF  180,180,130
  :   220 END
```

Program 13

Title: **Flow Diagrammer (tape)**
Author: Keith S. Reid-Green
Educational Testing Service
Princeton, NJ
Memory Requirement: 16K
Peripherals: 4662 Plotter
Statements: 917
Files: 5 ASCII Program
Requires dedicated tape

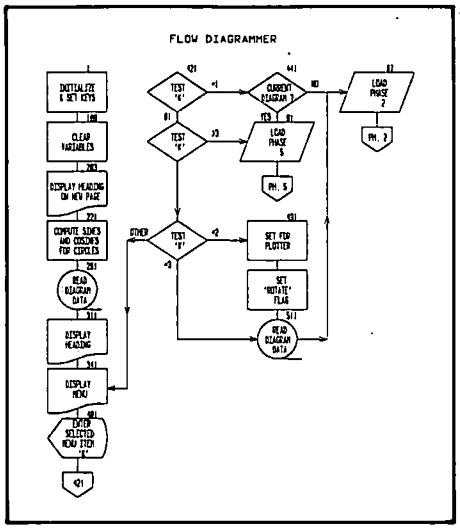
The program is used to design, store, recall and modify flow diagrams for use in program and system documentation.

A diagram consists of a heading, 10 different box types, their connecting lines and labeling. Boxes and lines may be solid or dotted and may be arranged up to 4 across and 9 deep on a page.

The program consists of 5 phases:

1. Main menu and function keys
2. Enter boxes
3. Connect, insert, delete boxes
4. Enter box data and heading
5. Store or retrieve diagrams

The first 62 files of a tape must be dedicated to this program. Files 1 through 5 contain the program; files 6 through 42 contain information about the current diagram; and files 43 through 62 store up to 20 diagrams.



Program 14

Title: **Flow Diagrammer (disk)**
Author: Keith S. Reid-Green
Educational Testing Service
Princeton, NJ
Revised by: L.C. Sheppard
Sheppard Software Co.
Sunnyvale, CA
Memory Requirement: 32K
Peripherals: 4662 Plotter
4907 File Manager
Statements: 923
Files: 1 Program
Requires dedicated data disk

The program is used to design, store, recall and modifiy flow diagrams for use in program and system documentation.

A diagram consists of a heading, 10 different box types, their connecting lines and labeling. Boxes and lines may be solid or dotted and may be arranged up to 4 across and 9 deep on a page.

The program consists of 6 phases:

1. Create data disk
2. Main menu
3. Enter boxes
4. Connect, insert, delete boxes
5. Enter box data and heading
6. Store/retrieve/destroy diagrams

A disk will hold 200 diagrams with up to 2000 charcters of "box data" per diagram.

Programs 15-16

Title: **Segmented Data Base and Windowing Routines**
Author Leslie L. Brabetz
Tektronix, Inc.
Wilsonville, OR
Memory Requirement: 32K
Peripherals: Optional-4907 File Manager
Statements: 701
Files: 3 ASCII Program
2 Binary Data

A series of articles in TEKniques (Vol. 1 No. 10, and Vol. 2 Nos. 1 and 2) described the theory and operation of creating a segmented data base from a large serial data base for fast windowing. Five files included in this program illustrate the mechanics of carrying out segmentation and windowing.

One routine allows definition of rectangular data windows. A master file may be read in and the vectors which begin and end or intersect the data window are stored in a segment file. The coordinates of intersection with the boundaries are calculated and stored in the segment file. The master data file must be in the form of arrays, with the number of coordinate pairs, N, followed by the coordinate arrays, X, Y.

N, X_1, X_2, . . . , X_n Y_1, Y_2, . . . , Y_n

Output segment files are created with the same format. To apply this routine to a user's data will require some revision of the program I/O and segment definition.

A small routine is included which generates the two data files. A third routine similar to the first is included. However, it directs the output to the display rather than a segment file and input files are read from the tape rather than the disk.