TEK APPLICATIONS LIBRARY
PROGRAM DOCUMENTATION

PART NO.
062-6715-00

TEKNIQUES
VOL. 7 NO. 1 D1

APPLICATIONS LIBRARY
APPLICATIONS LIBRARY
APPLICATIONS LIBRARY
APPLICATIONS LIBRARY
APPLICATIONS LIBRARY
APPLICATIONS LIBRARY
APPLICATIONS LIBRARY
APPLICATIONS LIBRARY
APPLICATIONS LIBRARY
APPLICATIONS LIBRARY
APPLICATIONS LIBRARY
APPLICATIONS LIBRARY

Tektronix®
COMMITTED TO EXCELLENCE

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

TEKNIQUES
VOL. 7 NO. 1 D1
062-6715-01

DOCUMENTATION

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

2/83

TITLE	PART NUMBER
TEKniques Vol. 7 No. 1 D1	062-6715-01
ORIGINAL DATE: January 1983 / REVISION DATE:	EQUIPMENT, OPTIONS AND SOFTWARE REQUIRED (INCLUDING PERIPHERALS AND HOST SYSTEM)
AUTHOR	

ABSTRACT

TEKniques Vol. 7 No. 1 D1 disk contains six 4050 Series programs: one CAD, one Project Aids, and four Utility. The individual abstracts describe each program.

Each of these programs should be transferred to its own dedicated disk. The programs all create or require files for working, modifying, and storing.

***** READ THE DOCUMENTATION BEFORE RUNNING!! *****

Title/ Program #	File Name	Documentation Page #
Directory	$DIRECTORY OLD"$DIRECTORY"	
PICTURE30 Program 1	SCRATCHLIB/PICTURE30 @DEMO/PDB/MOTH (demo picture) @DEMO/PDB/BUTTERFLYS (demo picture) @DEMO/PDB/MODULE (demo picture) @DEMO/PDB/ICLAYOUT (demo picture) @DEMO/PDB/SAMPLE (demo picture) @DRIVE/PDB/SUB5HPDR.BASE (demo picture) @DRIVE/PDB/SUB40HFL (demo picture) @DRIVE/PDB/SUBTERM1 (demo picture) @DRIVE/PDB/SUBTEST1 (demo picture) @DRIVE/PDB/SUBZJ544 (demo picture)	1
DISK-TO-TAPE BACKUP/ RESTORE UTILITY (51/07-8048/0 incl with PICTURE30)	$BACKUP $BACKUP.BØ1 $BACKUP.BØ2 $BACKUP.BØ3 $BACKUP.RØ1 $BACKUP.RØ2 $BACKUP.RØ3	47

Tektronix®
COMMITTED TO EXCELLENCE

DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

TITLE	ABSTRACT NUMBER
PICTURE30	
ORIGINAL DATE: June 1, 1982 / REVISION DATE:	EQUIPMENT AND OPTIONS REQUIRED: see below
AUTHOR: J.A. Loughran, A.J. Yerman, General Electric Company, Schenectady, New York 12345	PERIPHERALS: see below

ABSTRACT

1. INTRODUCTION

Program Abstract

PICTURE30 is a simple but very capable interactive picture composition program. It uses short, sensible, and easy to learn mnemonic keyboard entry commands. The program uses the joystick (or thumb wheels) or, optionally, the graphics tablet, to define and manipulate six primitive objects. The combination allows the user to create elementary to complex "pictures": sketches, diagrams, drawings, space layouts, flow charts, electronic schematics, etc. Stored automatically on the 4907 disc as they are created, most of these pictures can then become additional objects for inclusion in subsequent pictures. The program makes extensive use of the 4054 Dynamic Graphics Option 30, hence its name PICTURE30. It also uses the 4052R11 Character Generator ROM Pack for enhanced text and refresh graphics capabilities.

Equipment Requirements

4054 Graphic Systems Desktop Computer, with
Option 24 64K Memory expansion
Option 30 Dynamic Graphics
4052R11 Character Generator ROM Pack
4952 Joystick, Option 2
4907 Disc File Manager with one or more drives
4663 Plotter (4662 optional with some program changes)
4956 Graphics Tablet (or equivalent)

Files

1 - Binary program, PICTURE30, 1810 statements
10 - Binary data files, demonstration pictures

DEMO/PDB/MOTH	DRIVE/PDB/SUB5HPDR.BASE
DEMO/PDB/BUTTERFLYS	DRIVE/PDB/SUB40HFL
DEMO/PDB/MODULE	DRIVE/PDB/SUBTERM1
DEMO/PDB/ICLAYOUT	DRIVE/PDB/SUBTEST1
DEMO/PDB/SAMPLE	DRIVE/PDB/SUBZJ544

7 - Binary program, Disk-to-Tape Backup/Restore Utilities
51/07-8048/0

TITLE

PICTURE30

ABSTRACT NUMBER

2. ACKNOWLEDGMENTS

PICTURE30 has its roots in two programs created by Ms. Connie Breithaupt, of Tektronix, Rockland, Maryland: 52/07-9538/0 4052/4 Drafting Program[1] and another, privately provided by her to the authors of PICTURE30. Despite the merits of both programs, neither satisfied the desired performance requirements, but the combination became the precursor for PICTURE30 as recorded by the authors. With due respect, Ms. Breithaupt must also be considered a coauthor of PICTURE30.

The authors are also indebted to Edward H. Schadler, now at the Massachusetts Institute of Technology, for his contribution of the polygon concept during his brief summer association with the authors.

TITLE	ABSTRACT NUMBER
PICTURE30	

3. TERMS AND DEFINITIONS

Picture Data Base (PDB)

A picture is defined, in the context of this program, as a collection of individual primitive objects which when combined create a representation of a larger, more complex object or scene. The primitive objects generated by the program are the following: lines, labels, crosshatch, arrows, circles, and a particularly versatile one broadly called "polygon." Once created, most pictures can then become an additional object available for inclusion into another, new picture.

Thus a complex picture can be created from a collection of these primitive objects. As the picture is created, object by object, the data describing these objects is sequentially stored in a user-named, random access file on a floppy disc in the 4907 File Manager System. This file and its contents is defined as a Picture Data Base (PDB). The file length is limited only by the storage space available on the disc. Obviously, as the file length increases, so does the time required to access the picture data.

Occasionally, it may be advantageous to subdivide an elaborate picture into a group of related, individual PDBs, or overlays, to substantially decrease the time required to access the individual picture components. The final, complete picture is then generated on the plotter by sequentially overlaying the individual PDBs in sequence. This approach affords both reduced access time to the picture elements and the opportunity to change plotter pen colors between overlays.

PDBs are named in accord with the normal 4907 file identifier (FI) procedures. First, the user is prompted to enter a file name, which can consist of as many as ten alphanumeric characters followed by an extension of four additional characters preceded by a period. The program automatically prefixes a lowest level library name of "PDB" to the file name and requests entry of another library name in which to store or find the PDB. The user may then assign one or more library names. A single name is entered without the usual "/" delimiter—the program automatically provides it. Multiple library names must be separated by the "/" delimiter and end without it. The leading character for file, extension, and library names must be alphabetic, and nonalphanumeric characters are not acceptable.

The default 4907 library for PDB storage is USERLIB, automatically selected by the program. When given a PDB file identifier, the program first examines USERLIB to determine if the file already exists. If found, the file is opened for editing. If not found, a new file is created and opened for use. The program PICTURE30 resides in SCRATCHLIB.

It should be noted that the 4907 file manager recognizes the distinction between upper and lower case characters in a file identifier; to the file manager, "Filename" and "FILENAME" are not equivalent. Thus one must be consistent in the use of case in file identifiers.

PDB File Structure

All PDB data is stored in binary random access files. Each data entry occupies a 256 byte record. The first file record contains only a number indicating the total number of records occupied by object data. Each successive record contains the data for an object and its identification code. The objects, their codes, and data entries are as follows:

TITLE
PICTURE30

ABSTRACT NUMBER

Object	ID Code	Data Entry
Line (LIN)	1	Layer, X,Y (start), X,Y (end), dash code
Label (LAB)	2	Layer, X,Y (start), rotation, height, italics, auxiliary font number, text string
Crosshatch (XHA)	3	Layer, X,Y (start), (number of vertices)-1, angle, displacement, vertex array
Sub-PDB (PDB)	4	Layer, X,Y (start), rotation, scale, file name, Y factor, X factor
Arrow (ARW)	5	Layer, X,Y (start), X,Y (end), head angle, head size
Circle (CIR)	6	Layer, X,Y (origin), radius, start angle, end angle, rotation, width, X factor, Y factor
Polygon (POL)	7	Layer, X,Y (start), vertex array

An individual picture may contain from 1 to 100 layers, but layer 0 is reserved for a dedicated purpose, as explained later. Dividing a picture into layers has a definite advantage. The various layers may be displayed or plotted collectively or individually. Layering allows for distinctive color changes when plotting a picture, and for separating picture components, such as text from outlines.

Objects 1 through 6 can be entered only with the joystick (or thumbwheels). A polygon can be entered either with the joystick or the optional graphics tablet cursor.

Sub-PDBs

Most existing pictures can be reused as individual objects when creating a new picture. When reused, they are defined as a "sub-PDB." The entire sub-PDB data file is treated as though it were a single object as that object is added into the current PDB. The data contained in the sub-PDB file is never merged into the current PDB file, however. Only a reference to the sub-PDB file is added as the new object. Therefore, the source of the sub-PDB data must always be available whenever required by a PDB.

The use of a sub-PDB is subject to three minor restrictions:

1. The sub-PDB layer 1 cannot contain within itself a reference to another PDB. The program cannot cope with such "nested" PDBs. If encountered, the program ignores the nested sub-PDB, displays an error message, and continues to operate on the remainder of the sub-PDB.
2. The sub-PDB should be created approximately centered in the screen viewport. The center of the viewport becomes the center of rotation for an inserted sub-PDB. An off-center sub-PDB will appear to translate as it is rotated.
3. The sub-PDB must be inserted at the center of the viewport. To aid in doing so, a crosshair appears, along with viewport center tic marks along its left and bottom edges. Aligning the cursor to the tic marks assures correct centering. This is necessary to avoid clipping if part of the sub-PDB were to fall outside the viewport.

Sub-PDBs are particularly useful for building libraries of frequently used objects, such as symbols, for easy inclusion into new pictures. For best results they should be made centered in the viewport.

TITLE

PICTURE30

ABSTRACT NUMBER

A sub-PDB should be created with its important, most frequently used detail confined to layer 1. Other details should be relegated to layer 2 or higher. These other layers can contain additional data pertaining to the sub-PDB to help identify it as an individual PDB. For example, layer 2 could contain identifying text or other details unnecessary when the object is used as a sub-PDB. The program is keyed to bring in only layer 1 of a sub-PDB into the current PDB.

A useful technique for creating small symbols is to "zoom in" on a small central viewport area with the WINdow command. This scales the smaller area up to the full viewport size. Later, the enlarged working area is restored to normal area again with the WINdow command.

Polygons

Despite its classic and formidable name, polygon, a unique program feature, can be a complex array of one or more lines or simply a single point. As an array, it draws rapidly and consumes only one record in a PDB file for as many as 18 contiguous lines. The lines must be contiguous—there is no provision for interrupted lines. This restriction can be easily circumvented by "doubling back" if required. For example, a cross can be constructed by drawing a horizontal line first, drawing back to its midpoint, then to one of its vertical extremes and finally to its remaining other vertical extreme.

A polygon can be a figure of any shape constructed with contiguous straight solid lines, to a maximum of 18 lines. The figure need not close, unless crosshatching is required. Open polygons are automatically extended by the program. When entry of the 18th line is completed the current polygon is automatically stored, and another new polygon is started with its origin at the last point of the previous polygon. Thus, polygons can be extended indefinitely to create very complex figures. When digitizing from the tablet, it is often useful to mark the location of another object which cannot be entered via the tablet. This can be accomplished by entering a polygon of zero length, a point, as a location marker for subsequent joystick entry of the required object.

There are practical limitations imposed upon polygons. To be correctly crosshatched, they must close with a maximum of 18 sides. Crosshatching is performed automatically when the desired polygon is selected. When rotated in the current PDB, extended polygons will "fall apart," each rotating about its own origin. After rotation, the individuals can be combined again with the CHAnge command if desired. Extended polygons contained in a sub-PDB will be rotated and scaled in concert as the sub-PDB is rotated or scaled.

Discs and Drives

The program is designed to operate with either a single or a multiple drive 4907 Disc System. The disc containing the PICTURE30 program must operate, at least initially, in drive 0 to load the program and is considered the system disc. After starting the program, another working disc can be exchanged for the system disc, if desired, as prompted by the program. If an exchange is made, the program will automatically mount the working disc. All data storage occurs on the disc currently mounted on drive 0.

In a multiple drive system, a library disc can simultaneously reside in drive 1 for convenient access to sub-PDB libraries. A library disc is prepared by using it as a temporary working disc in drive 0, as described above. Thus PDBs can be added, altered, or removed and become library sub-PDBs when the disc is returned to drive 1. This

TITLE	ABSTRACT NUMBER
PICTURE30	

arrangement has the advantage of conserving free working file space on the drive 0 disc because the program searches both locations for the desired sub-PDB.

The user is urged to maintain adequate backup copies of all discs and periodically update them. Updates are best done in a multiple drive system with the CALL "DUP" routine. Files automatically extended by the 4907 file manager, such as PDB files, become scattered about the disc. Thus, they take longer to run. An update done with CALL "DUP" makes all files contiguous, and collects all unused space into one contiguous block. Contiguous PDB files run significantly faster with PICTURE30 than scattered ones.

Single disc 4907 users can employ the 51/07-8048/0 Disc-to-Tape Back/Restore Utilities[1] program for disc backup on tape.

Screen Display Format

The 4054 screen is divided into two areas. A column is provided along the left edge for interactive prompting and response entries. The second area is the picture composition working viewport, outlined as a box. The viewport is 100 x 100 graphic display units (GDU) in size, deliberately removed from the right edge of the screen for improved linearity. The viewport is approximately 10 inches square and has a default window of 4000 x 4000 user defined units (UDU), about the resolution limits of the screen display area. There is an implied default scale of 400 UDU per inch of viewport, used to advantage by the program for creating scaled pictures as explained later. The viewport origin is its lower left corner. See the illustration on page 8.

Grids and Axes

The program contains an optional capture or "jump-to" grid feature. The viewport is divided into a virtual X,Y grid structure whose default spacing is 50 UDU in both X and Y directions. With the jump-to grid feature enabled (default), a data point entered by the current graphic input device will "jump" to the nearest virtual grid point. Thus position input data is always rounded off to the nearest grid position and is always an integer value. Defeating the jump-to grid (see SELect command) allows a data point to be entered as a true current position, to the nearest 0.1 decimal fraction.

The grid is omnipresent, enabled or not. Its spacing is selectable (see SEL GRD command) and it can be displayed at will with the DIS GRD *n* command. The grid itself, as displayed, always employs the jump-to feature for accuracy. The grid is always "painted" in 10 x 10 point blocks in refresh and then fixed using the Option 30 for maximum speed, in response to the command DIS GRD *n*. The numeric *n* specifies every n^{th} point of the selected grid to be displayed. Some combinations of grid spacing and the specified value of *n* will create a grid display which overflows the upper and right bounds of the viewport. This is of no practical concern, since the points displayed within the viewport are true grid locations. Overflow is the result of "block" grid painting.

Another feature of the program is its axis command (see DIS AXS *n* command). Axis generates tic marks along all four sides of the viewport. The location of these tic marks is a function of the current grid spacing selection. The axis command displays a tic mark at each n^{th} grid point. Some combinations of AXS *n*, grid spacing, and current window size may produce conflicting tic intervals along the upper and right edges of the viewport. In this case, the tic marks along the left and bottom edges are the correct ones and the others should be ignored.

TITLE	ABSTRACT NUMBER
PICTURE30	

Grid and axis are useful in defining a scale for the viewport. For example, the default grid of 50, 50 divides the 10 x 10 inch viewport into 1/8 inch increments, DIS AXS 8, then adds tic marks defining 1 inch intervals along the viewport bounds. Similarly, other combinations of GRD *n* and AXS *n* can conveniently rescale the viewport to represent a workspace of different working dimensions. Scale is treated further in "Plotting a Picture."

Graphic Input Devices

The program frequently invokes the pointer function which requires input from a graphic input device (GIN). The default GIN device is the 4054 Thumb Wheels or the alternative, and much preferred, 4952 Joystick, Option 2. Either is used to insert a new object into the current PDB. The exception is polygon, which can be inserted via either graphic tablet cursor, or by the default GIN device (see Polygon Graphic Tablet Entry). Individual, isolated, or contiguous lines are inserted via the graphic tablet as "polygons" having one or more sides. Either way, polygon lines are always drawn as solid lines.

Before using the tablet, the program must be given coordinates which will identify the active tablet working area. Since the screen viewport is square, the tablet workspace must also be square, otherwise aspect ratio distortion will result. The DIGitize command is used to digitize the diagonal corners of the workspace. Only one DIG command execution is required during current operation of the program unless the workspace area is changed. Then, or if the program is restarted, the DIG command must be repeated. See "Plotting a Picture" for the effect of workspace size on scaling.

Graphic Tablet

The 4956 Graphic Tablet (or equivalent) is the alternative GIN device for polygon. The tablet must employ the new style four-button (square buttons) tablet cursor. These buttons are used to enter commands when using the tablet. The tablet control must be set for STREAM $\overline{\text{SWITCH}}$, HIGH rate (lever at far right) and counts as its output. The program uses GPIB Address 8 for the tablet.

A highly recommended addition to the tablet work surface is a precision ruled polyester film grid, such as those available from Bishop Graphics, Inc.[2] or other sources. The grid is most valuable for establishing a workspace suitable for subsequent scaled plotting of data entered via the graphics tablet. The grid can be taped to the tablet surface, but great care must be taken to prevent skewing the grid to the tablet axes.

It is imperative that the tablet cursor be in intimate contact with the tablet surface while in use. Otherwise the cursor will not pick up the magnetic field signals radiating only a very small distance from the tablet surface. This is particularly important for commands entered via the tablet cursor buttons, where the natural tendency is to first remove the tablet cursor away from the tablet surface before pressing a button.

Cursors

Several forms of cursors appear during program execution in response to various commands which invoke the pointer function:

Default — Normal full screen firmware generated cursor.

Program generated — Viewport width. May be clipped at viewport boundaries. Blinks every two seconds for enhanced visibility.

TITLE
PICTURE30

ABSTRACT NUMBER

Tolerance box — Smallish square containing an inscribed crosshair. Size is scaled by current values of tolerance, grid, and window. Used for search functions.

Circle — Superimposes the program generated cursor at the origin (center) of the circle.

Sub-PDB — Initial insertion employs default cursor. The sub-PDB then becomes the cursor. A repeat insertion of the same sub-PDB saves it as the cursor.

Arc — Pulls a refresh string from a circle origin to starting angle, then from origin to ending angle. Requires resynchronization of default pointer with program generated pointer to continue.

These cursors can be moved beyond the viewport bounds. The program recognizes the out of bounds condition and adjusts a position data entry accordingly.

4. PROGRAM OPERATION

Starting PICTURE30 follows the normal 4907 file manager procedure:

1. Insert the system disc into drive 0.
2. If used, insert a library disc into drive 1.
3. Set the clock, if necessary: CALL "SETTIM", "DD-MMM-YY HH:MM"
4. Mount the system disc: CALL "MOUNT",0,A$
5. Call the program: OLD "PICTURE30"
6. With the program in memory, enter RUN

The program name will appear, momentarily highlighted in the center of the screen. Soon there will appear the first of a series of prompts, responded to with an entry and a RETURN, or a RETURN alone. The following dialog is typical:

```
Is sub-PDB Library Disc on Unit 1? N

Insert Another Working Disk in Drive #0 if Desired.

Indicate name of desired PDB File: BUTTERFLYS

Enter a Library name for File: DEMO

File @DEMO/PDB/BUTTERFLYS is open for editing!
```

The above dialog excluded use of a library disc. Had the library disc option been selected with a Y (yes), the program would then automatically mount the drive 1 disc. The drive 0 disc can be exchanged after the prompt and will also be automatically

TITLE

PICTURE30

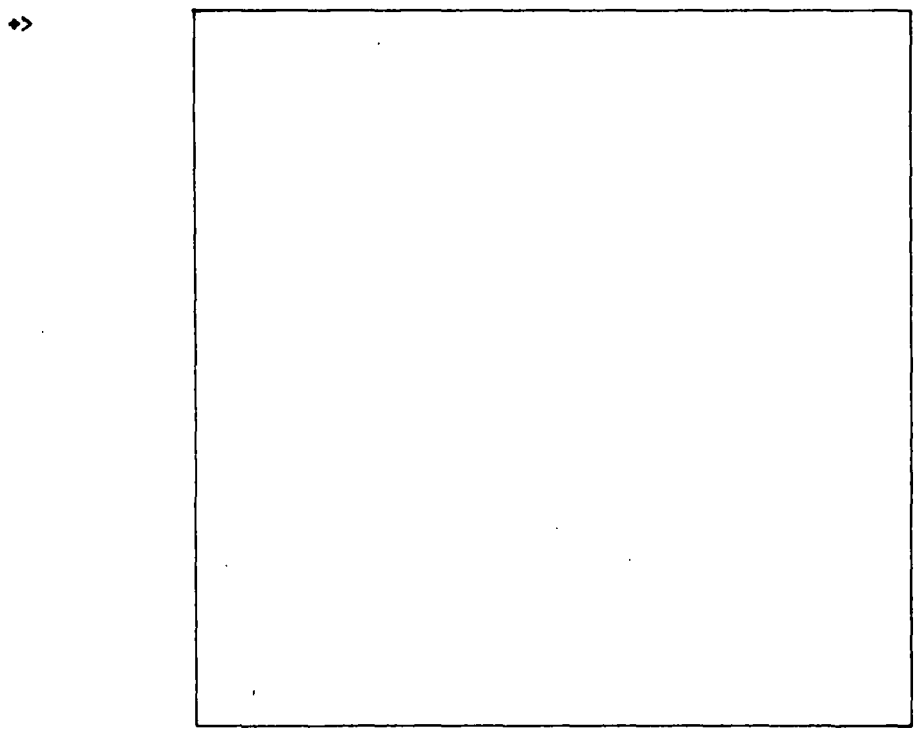
ABSTRACT NUMBER

mounted by a RETURN. Otherwise, simply enter a return. The remaining prompts establish the file name, which is displayed as "open for editing" if an old file, or "created for editing" if starting a new one.

After a one second delay, the screen is cleared and the viewport appears. The command mode prompt indicator "+>" then appears in the upper left corner of the screen. The "+" signals that the jump-to grid is active. The program is now ready to accept a command line entry which is always concluded with a RETURN or with a UDK entry. The figure below illustrates the initial screen display at this point in program operation.

If an existing PDB file has been reopened for editing, a DISplay ALL will display its current contents. A newly created PDB file is currently empty, so it must be constructed with a sequential series of appropriate commands.

Command mode screen display:



A group of User Defined Key (UDK) assignments has been provided for the most frequently performed program functions. As illustrated on the following page, UDKs 1 and 11 reactivate the program from an idle state. UDK1 resumes activity on the current PDB, typically following a PAUse command, and leaves undisturbed all parameters in their current status. UDK11 is nearly equivalent to an initial program RUN, closing any open files and restoring all parameters to their initial default status; essentially, it restarts the program from scratch.

UDKs 2 through 5 and 12 through 15 allow the rapid change of program operating parameters otherwise requiring a SELect command to reset. They represent the most commonly changed operating parameters but are active only during the command entry mode of the program, when the command prompt ">" is displayed on the screen. There are two methods of using these UDKs. While in the command mode, the appropriate UDK may be pressed, resulting in the parameter change followed by the

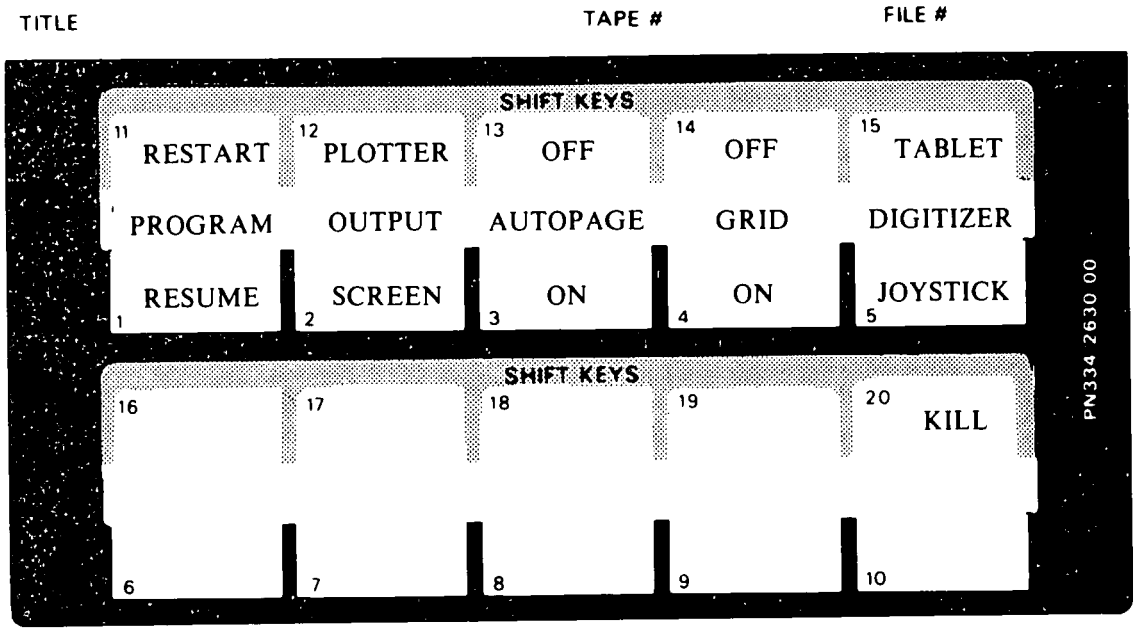
TITLE
PICTURE30

ABSTRACT NUMBER

command mode error message, a bell signal with an accompanying "?" display. Instead, a command line may be entered and terminated with the appropriate UDK instead of the usual RETURN. This method results in combined action on both the parameter change and command line function.

The kill function of UDK20 is active either when the program is in its idle state or in the command mode. Its function is to kill the current PDB if it is no longer useful. It provides a handy, fast method to eliminate a current PDB no longer worth keeping. Its use ends program execution and the program must be restarted with UDK11.

A problem can arise if a working disc is accidentally removed from its drive while a PDB file is still open. A later attempt to display the PDB will result in a program crash and an end of file (EOF) error message, caused by the previous loss of the EOF pointer. To recover from this problem, use the following procedure. Resume the program with UDK1, but do not attempt to display the selected PDB. Instead, do a LIS PDB, which will execute, but crash again, after listing the last object. Note the record number of the last object listed. Then enter, via the keyboard, P5=*n*, where *n* is a number one less than the number of the last object. Again resume the program with UDK1 but do not attempt to display. Now enter any new object—even a dummy if necessary—and Fix it. This will restore the EOF pointer and correct the problem. Then continue as usual. The dummy new entry can then be deleted if required.



User defined key functions:

PROGRAM	UDK1: RESUME	UDK11: RESTART
GRAPHIC OUTPUT	UDK2: SCREEN	UDK12: PLOTTER
AUTOPAGE	UDK3: ON	UDK13: OFF
JUMP-TO GRID	UDK4: ON	UDK14: OFF
DIGITIZER	UDK5: JOYSTICK	UDK15: TABLET
EDITING		UDK20: KILL CURRENT FILE

TITLE

PICTURE30

ABSTRACT NUMBER

5. COMMAND LINE FORMAT

All commands are entered in response to the command mode prompt ">" displayed at the left edge of the screen, and are terminated with a RETURN. Commands are entered via the keyboard as one or more three character mnemonic (MNE) representations of the desired function. All commands require a keyboard mnemonic entry and, in some cases, additional mnemonic keywords and/or numeric data. The command line syntax is rigid and depends somewhat upon the command being entered. The following illustrate typical command line formats:

Command keyword modifiers	(General)
MNE	(1)
MNE MNE *n*[,*m*]	(2)
CHG POL MNE	(3)

Single NME commands (1) must be followed immediately with a RETURN. Additional characters, like a space preceding the RETURN, are not allowed. Double entry commands (2) may contain either an optional [indicated by the use of brackets] or a mandatory numeric, *n*, and for some SELect keywords, an optional second numeric, [,*m*]. The use of numerics is explained in the individual command descriptions. A mandatory single space separates each MNE and also any trailing numerics. When used, the second numeric immediately follows the first, separated only by a mandatory comma. The triple entry MNE (3) is a special case unique to CHAnging a POLygon.

Some commands invoke a pointer function and display on the screen, usually in refresh, either a crosshair, a tolerance box, an object, or a combination of these. These pointer functions employ single character "operators" to perform functions associated with that particular command. These operations include object selection, rotation, scaling, etc., and the active ones are included in the command descriptions.

Operators temporarily alter parameters that were originally set to their default values or reset by a previously executed SELect command. Operators provide the convenient entry of a new parameter for the object currently displayed and are reset to their previous values upon completion of the current command and a return to the command mode. Entering an operator causes the refresh display of the current value and prompts for input of a new one.

Operator Descriptions

A Arc is used with INS CIR. Pull refresh line to starting angle, enter RETURN. Pull second refresh line to ending angle, enter RETURN. Cursor appears; align it with cursor for circle; enter RETURN.

B Build a polygon. Following a mandatory initial Move to a starting point, B pulls a refresh line attached to crosshair cursor from last point entered to current location of crosshair. Entering a B saves the current location. See INS POL for additional description.

C Cancel current operation and return to command mode. Many operations, like INS, are closed loops and use C to escape.

D Line Dash code, 0 to 5. Default is 0, a solid line. See INS LIN for details.

E Erases current object displayed and prompts for a replacement entry.

TITLE	ABSTRACT NUMBER
PICTURE30	

F Fixes current image on screen and may prompt for a new one. Otherwise returns to command mode.

H Height of label text characters. Default is horizontally oriented text with a size of 50. Range is 1-1000. Prefixing the height with a minus sign (−) generates vertically oriented text.

I Italics for label text. Default is OFF for no italics. Choice: OFF=0, ON=1.

L Layer number assigned to the current PDB activity. Default=1. Range 1-100.

M Move to current cursor position to start the definition of a new object.

P Mirror the Polygon currently displayed in refresh about its own X or Y axis. Cancel mirror by entering a P followed by a RETURN.

R Rotate the object currently displayed in refresh about its own origin. Default=0°. Range 0-360°.

S Scale for a sub-PDB or POLygon. Also radius for a CIRcle. Default: PDB or POL=1, CIR=250.

T Type of auxiliary font for label text. See LABel for details. Range 1-12. Default=10.

W Width factor for CIRcle or PDB. Default=1. Same as X axis scale factor at zero rotation, but rotates with object.

X X axis scale factor for CIR, PDB, POL. Default=1.

Y Y axis scale factor for CIR, PDB, POL. Default=1

1 Prompts for keyboard entry of arc starting angle in degrees. Range 0-360°.

2 Prompts for keyboard entry of arc ending angle in degrees. Range 0-360°.

↑ Arrow head size. Default 50. Range 1-1000.

> Arrow head angle. Default 35°. Range 0-360°.

= Translate a selected object *q* times by SEL TRA *n*,*m*. Default: *q*=0. Range 1-100.

6. COMMANDS

Command Summary

CHG CHanGes the location and/or attributes of an existing object.
Keywords: LIN, LAB, XHA, PDB, ARW, CIR, POL.

COM COMpresses the current PDB, eliminating deleted objects.
Keywords: none.

DEL DELetes an object selected with the search box.
Keywords: LIN, LAB, XHA, PDB, ARW, CIR, POL.

DIG DIGitizes the graphic tablet active workspace.
Keywords: none.

DIS DISplays graphics on the screen.
Keywords: ALL, AXS, GRD, LAY, MIR, ORG.

TITLE	ABSTRACT NUMBER
PICTURE30	

END END current session. Closes PDB file and restores normal system operation.
Keywords: none.

INS INSerts an object or a sub-PDB into the current PDB.
Keywords: LIN, LAB, XHA, PDB, ARW, CIR, POL.

LIS LISts the contents of the current PDB or the current operating parameters.
Keywords: PDB, PAR.

PAU PAUse and return to normal keyboard control. To resume activity with current PDB, press UDK1.
Keywords: none.

PLO Direct graphics output to the PLOtter instead of the screen.
Keywords: none.

REP Selects and REPeats a currently displayed circle or polygon.
Keywords: CIR, POL.

RES Closes the current PDB file, resets the default parameters, and REStarts the program in preparation for a new PDB. Same as UDK11.
Keywords: none.

SEL SELects new operating parameters for current PDB.
Keywords: many, see SELect command.

TRA TRAnslates selected object in current PDB.
Keywords: LIN, PDB, CIR, POL.

WIN Adjusts current working WINdow. New area selected with crosshairs.
Keywords: ALL, ZOM.

Command Descriptions

CHANGE

CHG MNE	MNE: LIN, LAB, XHA, POL, ARW, CIR, PDB name
CHG POL MNE	MNE: MOV, POI

CHAnges the location and/or the attributes of a specified object already displayed on the screen. The command entry invokes a pointer function and displays the tolerance box. The object to be changed is selected by centering the tolerance box crosshair at the origin of the object and entering a RETURN. See DIS ORG for locating the origins. The PDB file is then searched to locate the specified object. When it is found, the tolerance box is replaced with a small, blinking confirmation box and the CONFIRM prompt appears beneath the command line. Entering a Y and RETURN selects the object and concludes the search. Entering a N and RETURN rejects that object and continues the search for another similar one. If the object is not found, a message is displayed and the program returns to the command mode.

Confirming the object marks its origin with a # and deletes it from the PDB. The object appears anew in refresh, at first concealed beneath the original object, but attached to the pointer. The object can now be moved and manipulated with the same operators used for the corresponding INSert command. The operator C cancels the command entry.

TITLE	ABSTRACT NUMBER
PICTURE30	

POLygon is a special case. It can be operated upon as an object (MOV) or in a point (POI) mode. The object mode operates as described above, but with the introduction of additional operators not available to polygon in the insert mode.

Operators: C, F, L, P, R, S, X, Y.

The point mode is used to change discrete points or lines within the polygon, otherwise leaving it intact as an object. For the point mode, the polygon is selected as described above. However, upon selection, it is only marked, remaining intact upon the screen and in the data base, and the crosshair appears. To relocate a point, center the crosshair on that point and enter a B. The selected point will be marked with a * and the crosshair will reappear. Move its intersection to the desired new location and enter another B. The effect of the point change will now be seen in refresh as one or two new lines connecting to the new location, depending on whether an end or intermediate point was selected. Repeat the sequence as required, and when done, save the result by entering an F. The program then returns to the command mode.

Operators: C, B, F.

COMPRESS

COM MNE: none

Deleted objects are not physically removed from the data base by DELete. Instead, they are relegated to layer 0, which is never displayed or plotted. COMpress does a sort on the data file and discards all layer 0 entries. It should be used frequently when deleting to reduce the file length, the redraw time, and the time required to compress the file. Many minutes may be required to compress a long file containing numerous deletes.

Warning: Interrupting a compress may destroy the file! Be patient.

DELETE

DEL MNE MNE: LIN, LAB, XHA, PDB, ARW, CIR, POL

DELetes an object from the current PDB. The object is selected and confirmed as described above for CHAnge. Thus selected, the object is not physically removed from the PDB; it is only relegated to layer 0. See COMpress above.

DIGITIZE

DIG MNE: none

Prerequisite: SEL DIG 1

Determines the graphic tablet active workspace for use by the tablet cursor. DIGitize has the effect of normalizing the tablet workspace to the window, and thus establishes an implied scale factor for tablet data input. Appropriate workspace selection can be used to accurately scale an existing drawing into the current data base via the tablet cursor. When entered, DIGitize prompts for tablet cursor input; first, the lower left corner and then the upper right corner of the workspace. Align the tablet cursor to those points, pressing the yellow button for each. The program then returns to the command mode. Once set, the scale remains in effect until the program is restarted or the workspace is changed.

TITLE

PICTURE30

ABSTRACT NUMBER

DISPLAY

DIS MNE MNE: ALL, AXS *n*, GRD *n*, LAY *n*, MIR *n*, ORG [*n*]

DISplays on the screen the function indicated by the keyword and its related numeric, if applicable. DISplay alone, without a keyword, simply clears the screen, resets the prompt position, and draws the viewport in preparation for another command.

DISplay ALL

Displays on the screen all layers contained in the current PDB. It also resets autopage ON so that all subsequent DISplay LAYers or MIRrors will clear the screen prior to the display.

DISplay AXS *n*

Overlays on the screen along all four sides of the viewport a series of axis tic marks. The numeric *n* is a multiple of the current jump-to grid value, representing each n^{th} grid point. See previous discussion on jump-to grid.

DISplay GRD *n*

Overlays on the screen every n^{th} point of the jump-to grid spacing determined either by the default grid spacing (50,50) or by a previously executed SEL GRD *n*[,*m*] parameter. The grid display employs jump-to grid as described under Terms and Definitions.

DISplay LAYer *n*

Displays on the screen a single layer, *n*. Clearing of the screen prior to the display is determined by the current selection of autopage ON or OFF; autopage ON is not reset by displaying LAYer *n*.

DISplay MIRror *n*

A specialized but occasionally very useful feature. It displays on the screen in all four quadrants an axisymmetrical layer which was generated in the first quadrant with its origin at 0,0 (lower left corner of the viewport). This display mirrors about both the original X and Y axes. Its value lies in eliminating the duplication of detail otherwise necessary in generating an entire layer symmetrical about the center of the viewport.

DISplay ORG [*n*]

It is impossible to remember the origins for each object in a PDB! This feature provides a display on the screen of the origins of all objects contained in the current PDB as required for CHG and DEL. DIS ORG displays each origin for all objects, while DIS ORG *n* displays only those origins contained in layer *n*.

END

END MNE: none

END terminates current session, closes all open files, resets unit 0, clears the screen, and restores the normal system operating mode. The program remains in memory and can be restarted with UDK11 if desired.

TITLE	ABSTRACT NUMBER
PICTURE30	

INSERT

INS MNE MNE: LIN, LAB, XHA, PDB, ARW, CIR, POL, KBD

INSerts a new object into the current PDB. All INSert functions are looped: after execution of the current insertion, they expect another insertion of the same kind. Except for KBD, entering the operator C breaks the loop. See KBD for its escape.

INSert LINe

INSerts a LINe into the current PDB. The command invokes a crosshair. Position the crosshair center at the starting point for the line and enter an operator.

Operators: C, M, L, D

Operator M is required to establish the line starting point. Move to the line end point—a refresh string anchored at the start point will follow the pointer. At the end point, enter another operator.

Operators: C, M, F, L, D

The operator F will fix the line and automatically start a move (M) from the current pointer location for another line. A C will cancel insert and return to command mode. Alternatively, a contiguous line can be added to the first line by entering an M without moving the crosshair, or the crosshair can be relocated and an M entered to start a new, isolated line. This procedure can be repeated as desired, then terminated with a C.

It must be noted that contiguous lines make inefficient use of file storage space. Instead, polygon can store up to 18 lines in the file space required by one line. The primary advantage of line over polygon is the dash code available with line but not with polygon.

Line dash code:

n	Line	Type
0	———	solid
1	- - - - -	short dash
2	.....	dotted
3	— — —	long dash
4	—·—·—	dash/dot
5	— - —	long/short dash

INSert LABel

INS LAB text

INSerts a LABel into the current PDB. The command brings the label text into refresh display for manipulation by operators before fixing and saving. After fixing the first label, a prompt appears requesting a new label text entry. The label loop can be escaped by entering a null string (RETURN) or a C and RETURN. Note: to use C as a single character label, insert a space after it before the RETURN. Jump-to grid is not recommended for INS LAB. The text is limited to a maximum of 72 characters.

Operators: F, C, E, T, L, R, H, I

Label text is generated by the 4052R11 ROM Pack, including its unique auxiliary font capability. This A font is selected with operator T and, in the text string, enabled with a control N (N) and terminated with a control O (O). See Appendix A or the ROM Pack instruction manual for the auxiliary fonts available and their use.

TITLE	ABSTRACT NUMBER
PICTURE30	

INSert XHA (crosshatch)

Crosshatching is limited to closed polygons and is done automatically upon selection of the desired polygon. An attempt to crosshatch an open polygon will abort, produce an error message, and a return to command mode. Selection of the polygon is made as in CHG, described earlier. Upon confirmation, the crosshatch is drawn on the screen and then stored. The crosshatch origin is the same as that of the selected polygon.

The crosshatch parameters for angle and displacement are either default (45°, 20) or those selected previously by a SEL XHA *n*,*m* command.

INSert PDB name

When inserted into the current PDB, another PDB is considered a sub-PDB and is subjected to some restrictions as described earlier under Terms and Definitions. A sub-PDB can be a PDB called from a library disc, usually place on the 4907 drive 1 or on the default disc, always on drive 0. The program asks whether a library disc is mounted on drive 1 and, if not found there, will then search the default disc on drive 0. If not found in either location, an appropriate message will be displayed and the program will return to the command mode.

When found, a sub-PDB is entered and displayed in refresh for manipulation prior to inclusion into the current PDB.

Operators: F, C, M, S, R, W, X, Y, L

It must be noted that the sub-PDB data base is never entered into the current PDB file. The sole entry is a reference to the original sub-PDB. Thus, the source of the sub-PDB must always be available to the current PDB whenever it is called. This indirect approach conserves file space in the current PDB.

INSert ARW

INSerts an arrow into the current PDB. The procedure is identical to that of INS LIN except that the arrow head occurs at the origin. The operators are similar to those for INS LIN except for the dash code—arrows are always solid lines and the addition of operators for arrow head size and angle.

Operators: C, M, F, L, ↑, >

INSert CIRcle

INSerts a CIRcle or an arc into the current PDB. A default circle of radius 250 appears in refresh, along with a crosshair at the origin.

The operators are: F, C, A, L, 1, 2, S, R, W, X, Y

Most operators perform functions already described. The exceptions are A, 1, and 2.

Operator A allows the definition of a arc by pulling a refresh string from the origin of the circle, first to the starting angle and entering a RETURN, then to the ending angle and entering a RETURN. Now a new crosshair will appear on the screen at the ending angle location, along with the previously displayed circle with its own crosshair. The new crosshair must be moved to exact coincidence with the crosshair bisecting the circle, and a RETURN entered to synchronize both crosshairs. Following this procedure, the arc is subject to normal operator manipulation.

The alternative and easier arc definition is to enter a 1, which will prompt for the arc starting angle in degrees. Enter this value, followed by a RETURN, followed by a 2. A prompt will then appear for the arc ending angle in degrees. Enter this value followed with a RETURN. The resulting arc will be displayed and respond to normal operator manipulation.

TITLE	ABSTRACT NUMBER
PICTURE30	

INSert POLygon

INSerts a POLygon into the current PDB using either (1) the joystick or (2) the graphic tablet. A polygon can be as simple as a single line or as complex as an 18-sided figure. For crosshatching, the polygon must close with no more than 18 sides. Polygon lines can cross and still be crosshatched correctly. Polygon will extend automatically—after the 18th side has been entered, the current polygon will be fixed and stored, followed by an automatic move to the last point entered. This starts another polygon and is, except for the storage action in the 4907 disc, transparent to the user. Thus, polygon can be continued indefinitely. Extended polygons cannot be crosshatched.

If by chance the current polygon is completed upon entry of the 18th side, thus invoking the automatic store and extend feature, and the next polygon is to be an isolated one, an Erase operator must be entered. Erase breaks the automatic extension and prepares the program to accept a new, isolated polygon.

(1) Joystick polygon entry

This is the default entry mode. It disables the graphic tablet entry mode if that mode had been previously selected. The command invokes the crosshair. Move the crosshair to the polygon starting point and enter Operator L and/or M. Then move the crosshair and its trailing refresh string to a new location corresponding to the end of the first polygon side and enter operator B. Repeat the crosshair move and its operator B until the polygon, as viewed in refresh, is completed. Fix and store it with operator F or, if necessary, erase it with operator E and start anew. Use operator C to escape the polygon entry loop to return to the command mode. Be sure to enter the last point with a B before Fixing.

Operators: B, C, E, F, L, M

(2) Graphic Tablet Entry

Prerequisites: SEL DIG 1 and DIG

Polygon entry from the tablet is similar to joystick entry except that the crosshair cursor is positioned by the tablet cursor and the operators are entered via the cursor buttons in the following code:

Button	Color	Function
Z	yellow	Move or Build after an initial move
1	white	Fix
2	blue	Cancel
3	green	Erase

There is one significant and observable difference when using the tablet input: with the jump-to grid activated the screen cursor will move incrementally, jumping from one to another grid point as the tablet cursor is moved. Deactivating the jump-to grid produces the same smooth screen cursor motion as does the joystick. The tablet cursor must always contact the tablet to be active.

Cancelling the polygon graphic tablet insertion mode automatically restores the default joystick insertion mode. It is not necessary to use SELect DIGitizer or UDK5 for the purpose. DIGitize does require their use; otherwise the program remains in the tablet entry mode.

TITLE	ABSTRACT NUMBER
PICTURE30	

INSert KBD

Interactive insertion of individual lines via keyboard data entry has been provided to facilitate (1) precisely located reference marks and (2) fine detail too difficult to insert via the joystick or graphic tablet cursor. Keyboard data entry ignores the jump-to grid, activated or not, so that decimal fraction data may be entered.

Keyboard entry is a highly interactive mode, with the program prompting for detailed data for individual line: dash code, layer number, line starting and ending points. The data must be entered with care, since no error checking has been provided. Keyboard entry mode is looped. To escape, enter an out-of-range line dash code (>5) to return to command mode.

The following is a typical dialogue.

```
LINES MAY BE ENTERED FROM THE KEYBOARD WITH CARE.
THERE IS NO ERROR CHECKING!

USE OUT OF RANGE DASH CODE TO ESCAPE LOOP.

ENTER DASH CODE (0-5): 1
ENTER LAYER NUMBER (1-100): 1
ENTER START POINT (X,Y): 0,2000
ENTER END POINT (X,Y): 2000,2000

ENTER DASH CODE (0-5): 9
```

LIST

LIS MNE MNE: PAR, PDB

LISt PARameters

Clears the screen, then LISts the current program operating parameters. Ends with a blinking "?". Enter a RETURN to continue, returning to command mode.

LISt PDB

Clears the screen, reads the current PDB data file and displays its contents. Useful in deciding whether a COMpress should be performed to remove deleted objects. Ends with a blinking "?". Enter a RETURN to continue, returning to command mode.

Listing the contents of the current PDB can be directed to output to a line printer instead of the default screen display. Simply enter the command SEL OUT *n*, where *n* is the user's GPIB address for the printer prior to the LIS PDB command entry. When the listing has been completed, the program automatically restores all output to the screen display.

PAUSE

PAU MNE: none

Temporarily halts program execution with a stop message, and returns to normal keyboard control. Handy for coping with operator distractions or for entering program changes. Leaves current PDB file open, so beware! Resume operation on current PDB by pressing UDK1.

TITLE	ABSTRACT NUMBER
PICTURE30	

While in PAUse, the contents of some program variables can be conveniently examined. For example, entering T$ and RETURN displays the current PDB file name if needed. F$ and RETURN displays the name of the last used sub-PDB.

PLOT

PLO [*n*] MNE: none

Prerequisite: SEL OUT *n* *n* = plotter address

Directs the output to the 4663 for PLOtting. Use SEL OUT or UDK 2 to return to screen display mode. Does not affect screen display of commands.

When first executed, PLO calls a highly interactive routine for setting the plotter and for selecting the plot parameters. The plot can be A, B, or C sized with horizontal or vertical sheet orientation, and includes a choice of three types of plots. See PLOTTING section for more detail.

Completing the plotter setup routine displays a prompt and when responded to, returns to the command mode, awaiting a plotting command. PLO alone will then plot all layers in the current PDB. PLO *n* will plot layer *n* (*n*=1-100). PLO 0 will return to the plotter setup routine for changing it. See "Plotting a Picture" for further details.

REPEAT

REP MNE: CIR, POL

Selects, operates upon and REPeats a CIRcle or POLygon already displayed on the screen and included in the current PDB. The circle or polygon is selected, confirmed, and appears in refresh as in CHG, described earlier. The original remains intact in the current PDB.

REPeat CIRcle

Allows manipulation of the selected circle and multiple replication of the result with repeated Fixes.

Operators: F, C, A, L, 1, 2, S, R, W, X, Y

REPeat POLygon

Allows manipulation of the selected polygon and multiple replication of the result with repeated Fixes. Also introduces new operators unavailable with INSert POLygon.

Operators: C, F, P, S, R, X, Y, L

Using these operators, new polygon shapes can be derived from the original one by mirroring, scaling, rotating, adjusting the X or Y scale, or transferring the original into a different PDB layer.

RESTART

RES MNE: none

Closes the currently active PDB file, resets the default program and system operating parameters, and restarts the program in preparation for operating on another PDB. Also provides for a change of discs used in 4907 drive 0. Similar to restarting the program using UDK11.

TITLE

PICTURE30

ABSTRACT NUMBER

SELECT

SEL MNE *n*[,*m*] MNE: many, see summary

SEL MNE M$ M$: ON or OFF

SELects the current program operating parameters. Used to change them from their preset default values or current status.

SELect summary:

ANG *n* — Label text rotation angle in degrees
Default: 0 Range: *n* = 0 to 360

ARW *n* — Arrow head size
Default: 50 Range: *n* = 1 to 1000

DIG *n* — Graphic input device
Default: 0 (joystick/thumb wheels) Alternative: 1 (tablet cursor)

DSH *n* — Line dash code. See INS LIN.
Default: 0 (solid) Range: *n* = 0 to 5

FON *n* — Auxiliary font number (1 to 9 not recommended)
Default: 10 Range: *n* = 1 to 12

GRD M$ — Jump-to grid status
Default: M$ = ON Alternative: M$ = OFF

GRD *n*[,*m*] — Jump-to grid interval. *n* = X, *m* = Y
Default: *n* = *m* = 50 Range: 1 to 1000

ITL M$ — Italics status
Default: M$ = OFF Alternative: M$ = ON

LAY *n* — Currently active layer
Default: 1 Range: 1 to 100 Note: also selected by operator L

OUT *n* — Current graphics output device, screen or plotter
Default: 32 (screen) Alternative: 4 (plotter), 51 (printer)

PAG M$ — Autopage status
Default: M$ = ON Alternative: M$ = OFF

PDB *n*,*m* — Rotation (*n*) and scale (*m*) for next sub-PDB entry
Default: *n* = 0, *m* = 1 Range: *n* = 0 to 360, *m* = 0 to 100

TOL *n* — Tolerance value and size of tolerance box
Default: smaller of grid interval or ½% of current window
Range: *n* = 1 to 1000

TRA *n*,*m* — X, Y displacement for next object TRAnslated
Default: 0, 0 Range: ± 1 to 1000. See TRA command

WIN — Defines a new current window via pointer input.
Default: 0, 4000, 0, 4000 Reset to default by WIN ALL

XHA *n*,*m* — Defines new XHA angle (*n*) and displacement (*m*)
Default: *n* = 45°, *m* = 20 Range: *n* = 0 to 360°, *m* = 1 to 100

TITLE	ABSTRACT NUMBER
PICTURE30	

TRANSLATE

TRA MNE q MNE: LIN, PDB, CIR, POL
q = 1 to 100

TRAnslate allows a selected object to be repeated q times, each new object being displaced by a predetermined distance. Translate calls up the tolerance box as for CHG or DEL and operates in a similar manner. Position the box for the desired object and enter a RETURN. If the object is found, the CONFIRM prompt will appear on the screen. Entering a Y selects the object. The object will then be marked as selected and the refresh crosshair will appear.

Translate has three operating modes:

1. Automatic translation

   Enter an = and the crosshair will disappear. The selected object will be redrawn q times, each one translated at a distance determined by the current SEL TRA n,m parameters.

2. Manual selection of a translation distance

   Position the crosshair at any arbitrary point on the screen. Displaying a grid is useful for selecting the desired translation distance. Enter a RETURN and the crosshair will remain on the screen. Reposition the crosshair to a new position which represents the desired translation distance and direction, then enter a RETURN. The new object(s) will be drawn and translated by the difference between the two crosshair entry positions.

3. Repeating an object at a new location

   Enter the TRA command with q=1 and proceed with selection as described above. Position the first crosshair at the origin of the object (or at either end of a line) and enter a RETURN. Then move the crosshair to the new position desired for the origin of the object (or the same end for a line) and enter a RETURN. The object will then be repeated at the position of the second crosshair.

WINDOW

WIN MNE: ALL, ZOM

WINdow ZOoM effectively magnifies the working area within the viewport by setting a new, smaller workspace window. The WINdow ZOoM command invokes a crosshair which is first positioned to the lower left corner of the area to be enlarged and a RETURN is entered. The crosshair then reappears and is positioned in the upper right corner of the area and a second RETURN is entered. The area defined is then automatically squared, the screen is cleared, the viewport is displayed, and the command prompt appears. Enter a DIS ALL and only the pointer defined areas of the picture will be displayed within the viewport.

WINdow ALL restores the full default window, cancelling the magnification set by ZOoM, and returns to the command mode for the next instruction.

Displaying a zoomed workspace may seem, to the user, to be a slow operation. This impression is caused by the need to search through all objects selected by the DIS command parameter. The resulting display will contain only those objects found within the current, magnified window. Other objects are clipped at the viewport bounds.

TITLE

PICTURE30

ABSTRACT NUMBER

Objects created in a zoomed workspace are stored in the PDB using the full, default window. Thus, if the window is subsequently reset, either by a WIN ALL command or by restarting the program, it is necessary to re-zoom to the original workspace size to restore such objects to their corresponding, enlarged size.

7. DEMONSTRATION PICTURES

Several picture examples are included on the PICTURE30 System Disc. They reside in the following PDB files:

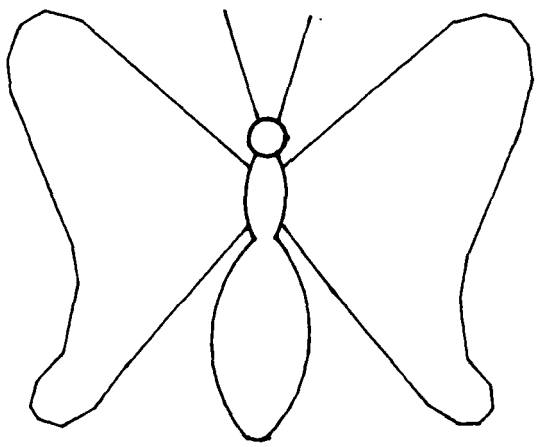
DEMO/PDB/MOTH
DEMO/PDB/BUTTERFLYS
DEMO/PDB/MODULE
DEMO/PDB/ICLAYOUT
DEMO/PDB/SAMPLE

These pictures are accessed by responding to the program starting prompts, first for the file name (i.e., MOTH), then for the library name, DEMO. The sub-library name PDB is automatically added by the program when the pictures are created but is never entered during the prompting. The picture is then produced with a DISplay ALL command.

DEMO/PDB/MOTH

The first example illustrates the steps of generating a sub-PDB picture and later using it to draw a more complex picture. The sub-PDB is shown in Figure 1 as plotted out on the Tektronix 4663 plotter. It is composed of two layers, the moth drawing on layer 1 and the label on layer 2.

The drawing itself was made up of an assortment of primitive objects which Table 1 enumerates (obtained by typing LIS PDB). The head and body section were formed of a circle and three circular arcs, respectively. The arcs have three different radii and were appropriately rotated (element code 6 in list). Each wing consists of an open polygon of 17 points (element code 7), and the two antennae are line segments (element code 1). The two labels (element code 2) reside in layer 2.



DRIVE/PDB/SUBBFY
820313-AJY

Figure 1. DEMO/PDB/MOTH

TITLE

PICTURE30

ABSTRACT NUMBER

Table 1
LISTING OF COMPONENTS FOR FIGURE 1

```
PDB:  $DEMO/PDB/MOTH

 2 #6 1 2000 2000 50 0 360 0 1 1
 3 #6 1 2100.1 1830.1 250 151.952936423 286.237650286 0 1 1
 4 #8 1 1800 1950 250 151.952936423 286.237650286 100 1 1
 5 #8 1 2250 1450 400 -44.0883634855 40.1213273215 100 1 1
 6 #6 1 1713.9 1479.7 400 -44.0883634855 40.1213273215 0 1 1
 7 #8 1 1875 1230 50 235.485609653 382.102854200 0 1 1
 8 #7 1 2041.6 1920
 9 #7 1 1954.6 1920
10 #1 1 0 2174.7 1592.3 2041.6 1766.4
11 #1 1 0 2031.4 2048 2118.4 2324.5
12 #1 1 0 1888 2330.8 1975 2853.1
13 #2 2 2220.8 931.8 0  30 0 10 DRIVE/PDB/SUBBFY 1 1
14 #2 2 2615 820.4 0  25 0 10 820313-AJY 1 1
```

DEMO/PDB/BUTTERFLYS

The picture in Figure 2, except for text, is composed simply of nine MOTHs as sub-PDBs entered using the INS PDB name command. In this case the name used was SUBTEST1. Since this is a looped command it had to be issued only once so that picture composition consisted simply of rotating and scaling the sub-PDB before each Fix command. Following this, the two labels were entered in layer 1 and a small one in layer 2. The Table 2 listing of this picture illustrates the components.

DEMO/PDB/MODULE

The third example illustrates the use of PICTURE30 for technical design and layout. Figure 3 is a scaled plan view of a solid state power control module. Using the jump-to grid feature, the digitizing tablet, a 50 mil grid spacing and 4X scale, sub-PDBs were prepared of the baseplate, separate large and small semiconductor chip carriers, and a terminal lug. The figure shows one arrangement of the baseplate and chip carriers with an overlay of a printed circuit board used for interconnections. The copper patterns on the two sides of the board are represented by crosshatching at two different angles. The terminal lugs which will ultimately be soldered to the board are shown as an additional overlay. A total of five layers was used with text labels occupying a separate layer. This permitted highlighting various features of the design by appropriate choice of different colored inks during plotting. Since an objective of this design study was to consider the space requirements for interconnecting up to eight semiconductor elements on a common baseplate, once the sub-PDBs were prepared variations in layout could be rapidly plotted for detailed study. A listing of this PDB is given in Table 3.

TITLE	ABSTRACT NUMBER
PICTURE30	

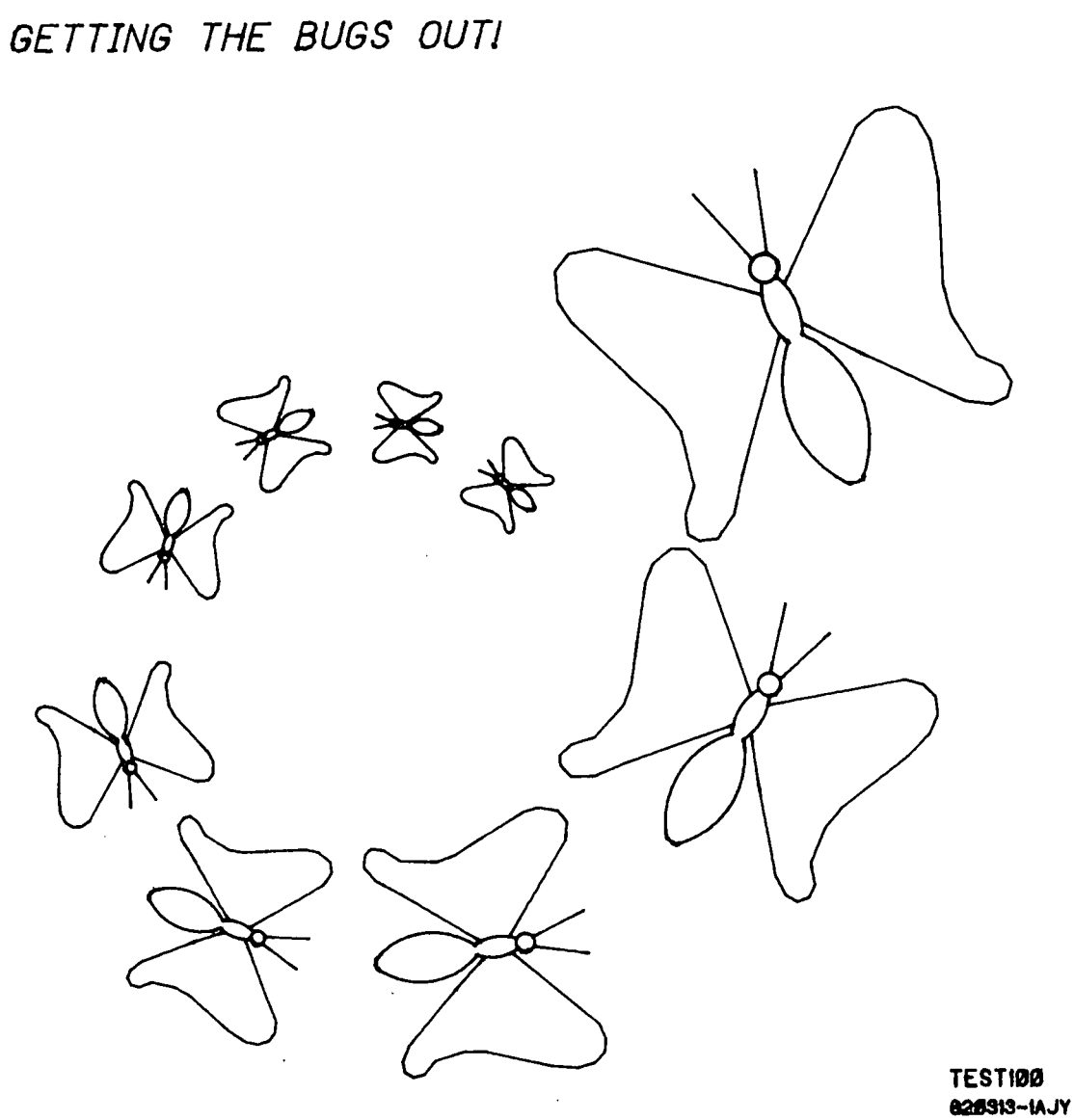


Figure 2. DEMO/PDB/BUTTERFLYS

Table 2
LISTING OF COMPONENTS FOR FIGURE 2

```
PDB:  @DEMO/PDB/BUTTERFLYS

 2 *4 | 2055.7 2005.2 25 | @DRIVE/PDB/SUBTEST1 | |
 3 *4 | 2055.7 1551.4 330 0.8 @DRIVE/PDB/SUBTEST1 | |
 4 *4 | 1095.5 680.5 200 0.65 @DRIVE/PDB/SUBTEST1 | |
 5 *4 | 1083.7 686.1 250 0.5 @DRIVE/PDB/SUBTEST1 | |
 6 *4 | 620.5 1200 200 0.4 @DRIVE/PDB/SUBTEST1 | |
 7 *4 | 756.5 2012.2 165 0.3 @DRIVE/PDB/SUBTEST1 | |
 8 *4 | 1000.5 2416.6 120 0.25 @DRIVE/PDB/SUBTEST1 | |
 9 *4 | 1565.4 2467.0 85 0.2 @DRIVE/PDB/SUBTEST1 | |
 10 *4 | 1023.8 2273.3 50 0.2 @DRIVE/PDB/SUBTEST1 | |
 11 *2 | 230.4 3773.4 0  50 | 10 GETTING THE BUGS OUT! | |
 12 *2 | 3470.1 46.1 0  25 0 10 020313-1AJY | |
 13 *2 2 3470.1 143.4 0  30 0 10 TEST100 | |
```

TITLE

PICTURE30

ABSTRACT NUMBER

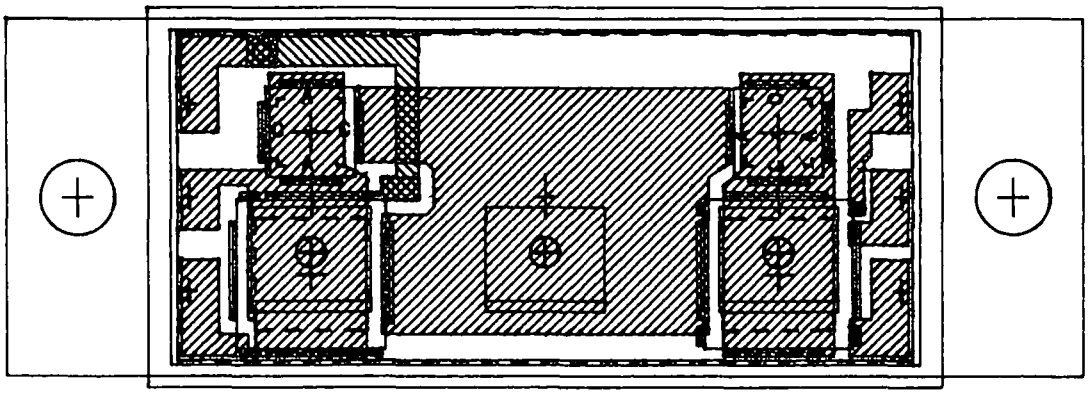


Figure 3. DEMO/PDB/MODULE

Table 3
LISTING OF COMPONENTS FOR FIGURE 3

```
PDB.  @DEMO/PDB/MODULE

 2 *4 1 2001 2007 0 1 @DRIVE/PDB/SUB5HPDR.BASE 1 1
 3 *4 1 1370 1000 0 1 @DRIVE/PDB/SUBZJ544 1 1
 4 *4 1 2030 1000 90 1 @DRIVE/PDB/SUBZJ544 1 1
 5 *4 1 1370 2100 0 1 @DRIVE/PDB/SUB40HFL 1 1
 6 *4 1 2630 2100 90 1 @DRIVE/PDB/SUB40HFL 1 1
 7 *7 2 1120 1840
 8 *7 2 1280 2360
 9 *6 2 1240 2400 25 0 360 0 1 1
10 *2 4 2640 1240 0  30 0 10 DRIVE/PDB/MOD102 1 1
11 *2 4 2040 1160 0  25 0 10 820304-2AJY 1 1
12 *7 2 2000 2450
13 *7 3 1560 2000
14 *4 5 1370 1860 0 1 @DRIVE/PDB/SUBTERM1 1 1
15 *4 5 2000 1860 0 1 @DRIVE/PDB/SUBTERM1 1 1
16 *4 5 2630 1860 0 1 @DRIVE/PDB/SUBTERM1 1 1
17 *7 2 2000 1840
18 *7 2 2000 2000
19 *2 4 2740 1100 0  25 0 10 REV.1 820324-1AJY 1 1
20 *3 2 1120 1840 6  45 20
21 *3 2 1280 2360 6  45 20
22 *7 2 1560 2000
23 *3 2 1560 2000 4  45 20
24 *3 3 1560 2000 8  135 20
25 *3 2 2000 1840 6  45 20
26 *7 2 2000 2340
27 *3 2 2000 2340 10  45 20
28 *3 2 2000 2000 6  45 20
29 *7 2 1260 2340
30 *7 2 1500 2300
31 *7 2 2530 2340
32 *3 2 1500 2300 12  45 20
33 *3 2 2530 2340 6  45 20
34 *3 2 1260 2340 14  45 20
```

Suggestion: the above program displays only layer 1 of the enumerated sub-PDBs. Experiment by calling and displaying them as individual PDBs to examine the details contained in their other layers.

TITLE	ABSTRACT NUMBER
PICTURE30	

DEMO/PDB/ICLAYOUT

This example is an actual rough trial design for a complex ceramic silicon integrated circuit package. A total of 196 connection patterns were required, distributed equally and symmetrically among the four conventional quadrants. The axial symmetry of the required conductor pattern prompted the development of the MIRror display and plot modes. The conductor pattern, layer 1, was created only in the first quadrant. Keyboard line entry was used to enter the fine detail immediately adjacent to the X and Y viewport bounds, too small to be accurately entered with either the joystick or tablet cursor. Each of these individual, half pattern open objects, defined with lines, forms a closed pattern when layer 1 is mirrored. The other closed patterns are polygons, entered via the tablet, using a jump-to grid spacing of 20 UDUs.

The design goal was a final overall package size of 1.40 in. square. A tablet workspace of 10 in. square was digitized and a crude 7 x 7 in. sketch (10X) of a pattern, drawn on grid film, was placed in the lower left corner of the workspace. The sketch represented only one quarter of the final pattern, an area of 0.7 in. square. Selecting a grid spacing of 20 and axis tic marks of 20 grid points effectively rescaled the viewport to represent an area 1 in. square with the axis tics representing 0.1 in. intervals and the grid representing 0.005 in. intervals at the desired 10:1 reduction. Reminder: the 10 in. viewport dimension represents 4000 UDUs, or 400 UDUs/in. (0.0025 in.) or, at 10X, 0.00025 in.; selecting grid 20 equates each grid interval to 0.005 for the original 10X enlargement.

Figure 4 shows layer 1 as it was generated. It also shows the command lines used to select the grid spacing, display every fifth grid point, and produce an axis tic mark every 20 grid points. The display command recalled layer 1. Figure 5 shows the results of the MIR LAY 1 command. Layer 1, originally generated only in the first quadrant, is now mirrored into all four quadrants. In Figure 6, layer 2, the package outline, is added to layer 1. This is done by first entering DISplay LAYer 2 and terminating the command line by pressing UDK 13, autopage OFF, instead of the usual RETURN. This method prevents automatic paging between displaying successive layers and allows screen overlays. Now enter DISplay LAYer 4; the result appears in Figure 7. There is also a layer 3 which, although important to the package design, contributes nothing useful to the demonstration.

Figures 4 through 7 are screen hard copy reproductions and fail to adequately demonstrate the resolution capabilities of PICTURE30 and the 4054 Graphics System. Suggestion: repeat the demonstration using the 4663 plotter, which will display the results more precisely, even to the mistakes made in layer 1! Try a C size sheet for best resolution.

TITLE

PICTURE30

ABSTRACT NUMBER

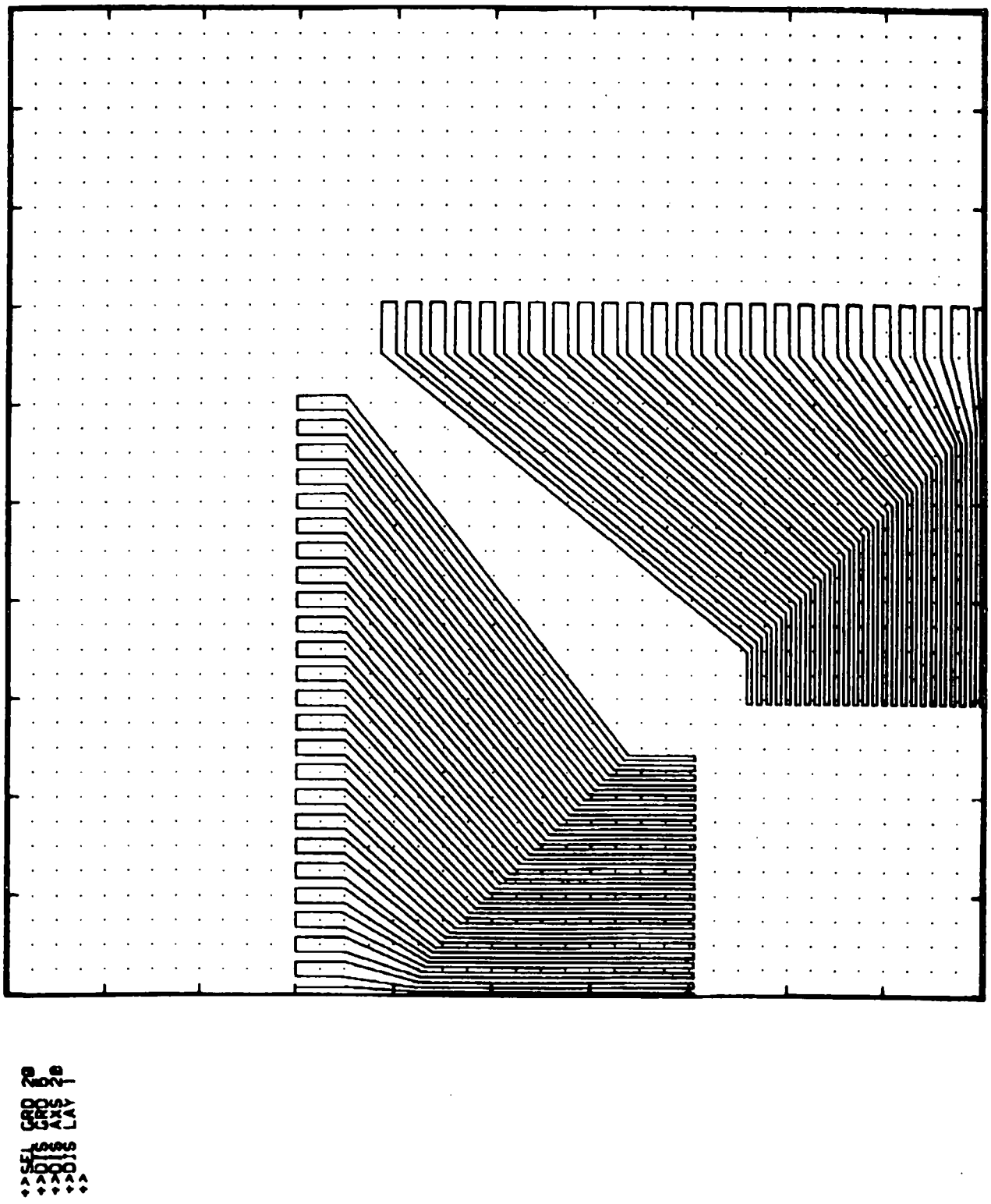
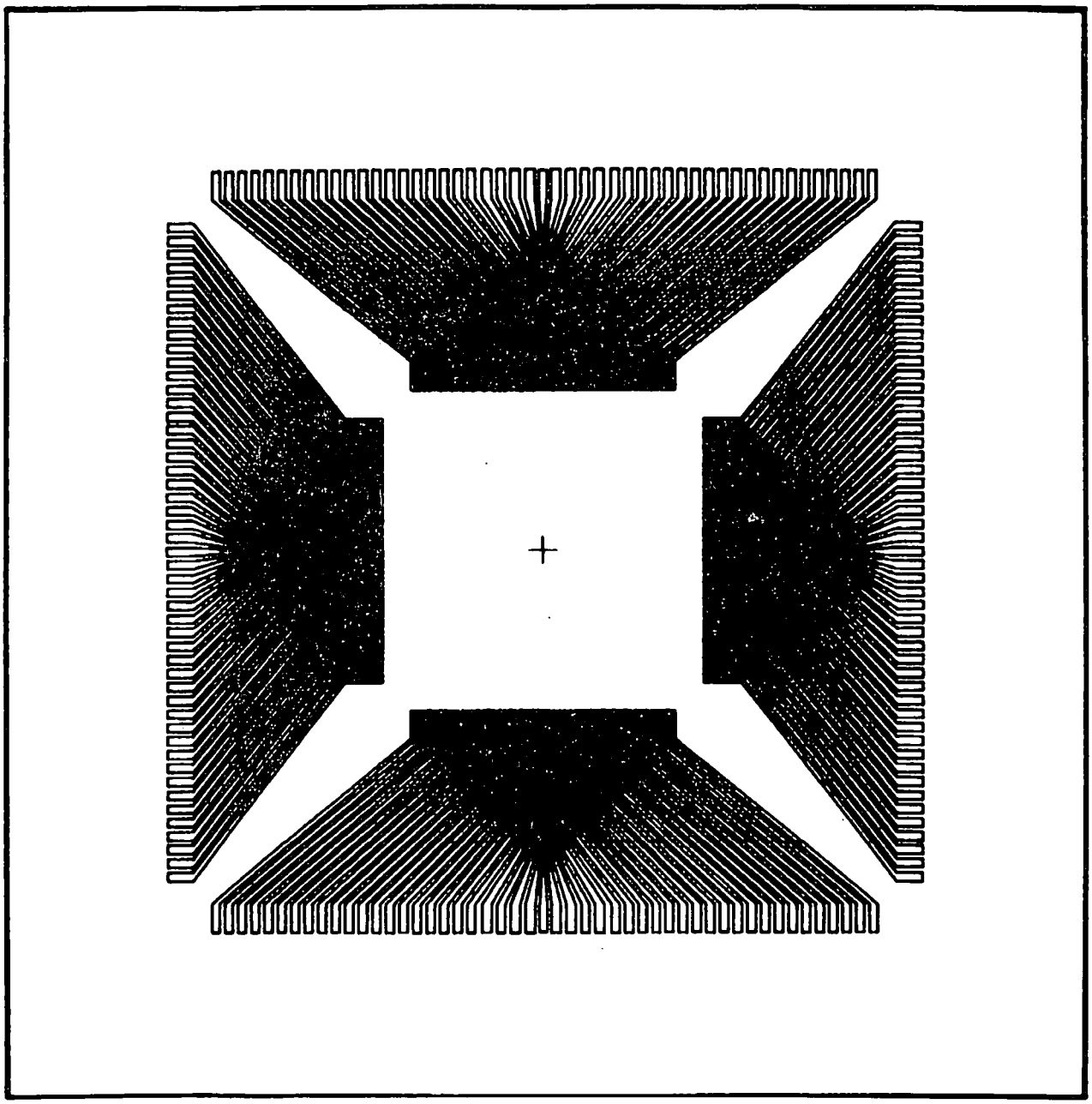


Figure 4. DEMO/PDB/ICLAYOUT. Conductor pattern, layer 1, as generated.

TITLE

PICTURE30

ABSTRACT NUMBER



```
*>DIS MIR 1
*>
```

Figure 5. DEMO/PDB/ICLAYOUT. Conductor pattern, layer 1, mirrored.

TITLE

PICTURE30

ABSTRACT NUMBER

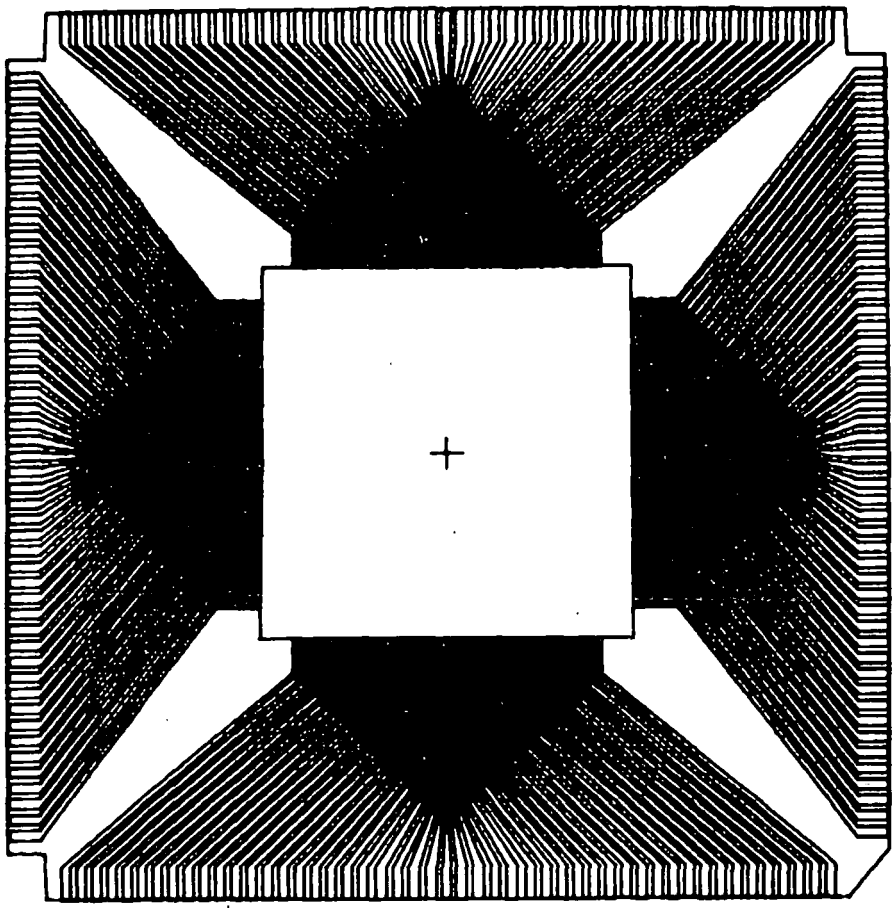
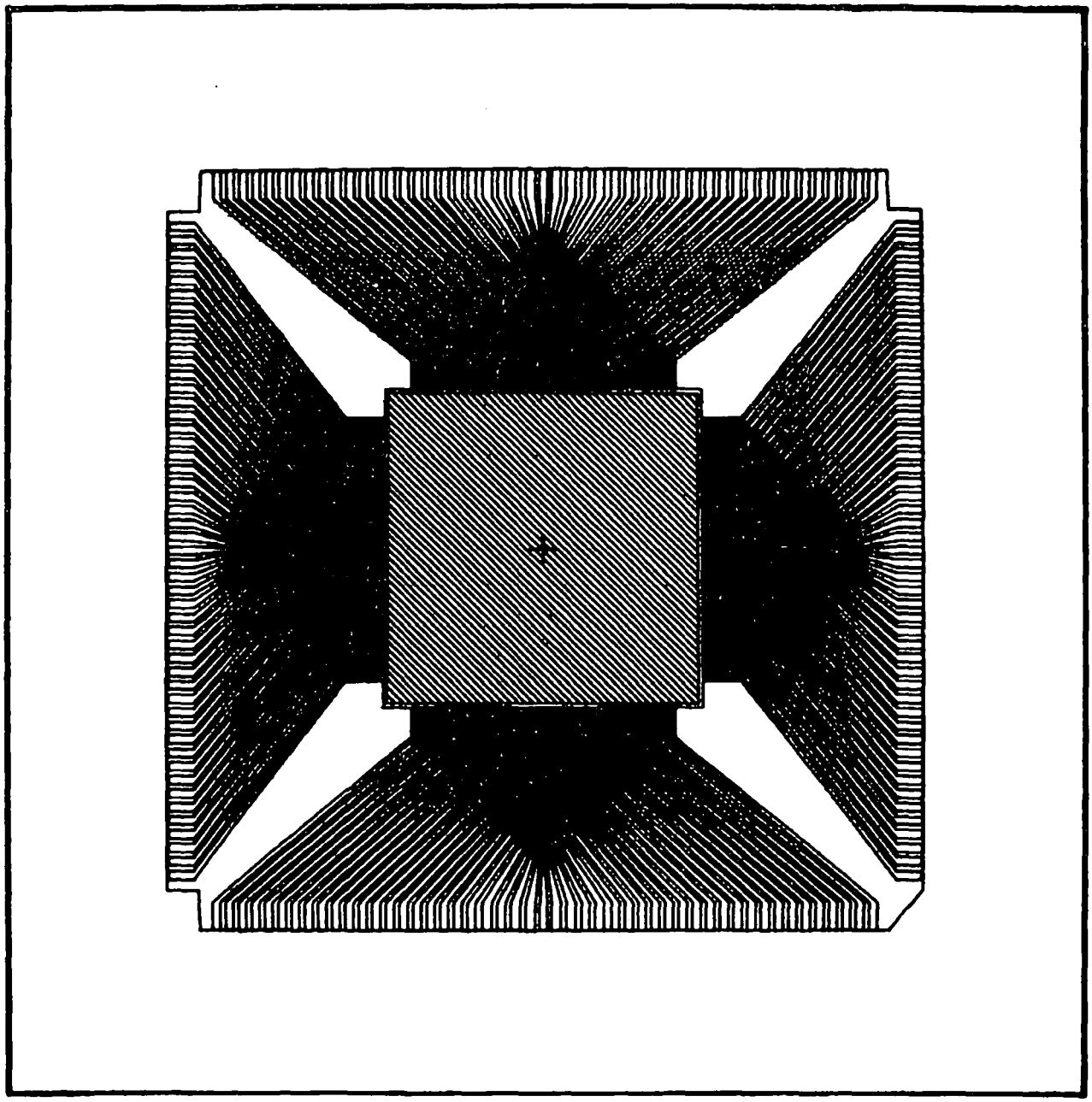


Figure 6. DEMO/PDB/ICLAYOUT. Package outline, layer 2 added to mirrored layer 1.

TITLE

PICTURE30

ABSTRACT NUMBER



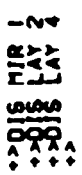


Figure 7. DEMO/PDB/ICPACKAGE. Completed design.

TITLE	ABSTRACT NUMBER
PICTURE30	

DEMO/PDB/SAMPLE

This example, shown in Figure 8, is a sample of some PICTURE30 capabilities, particularly for crosshatching. The crosshatched objects are closed polygons. The block letters I and H were made by first outlining them as polygons, crosshatching, and then deleting the surrounding polygons. Also shown is a text sample, done first in font 1 and then repeated in font 10 which produced some equivalent Greek characters. A listing of this PDB is given in Table 4.

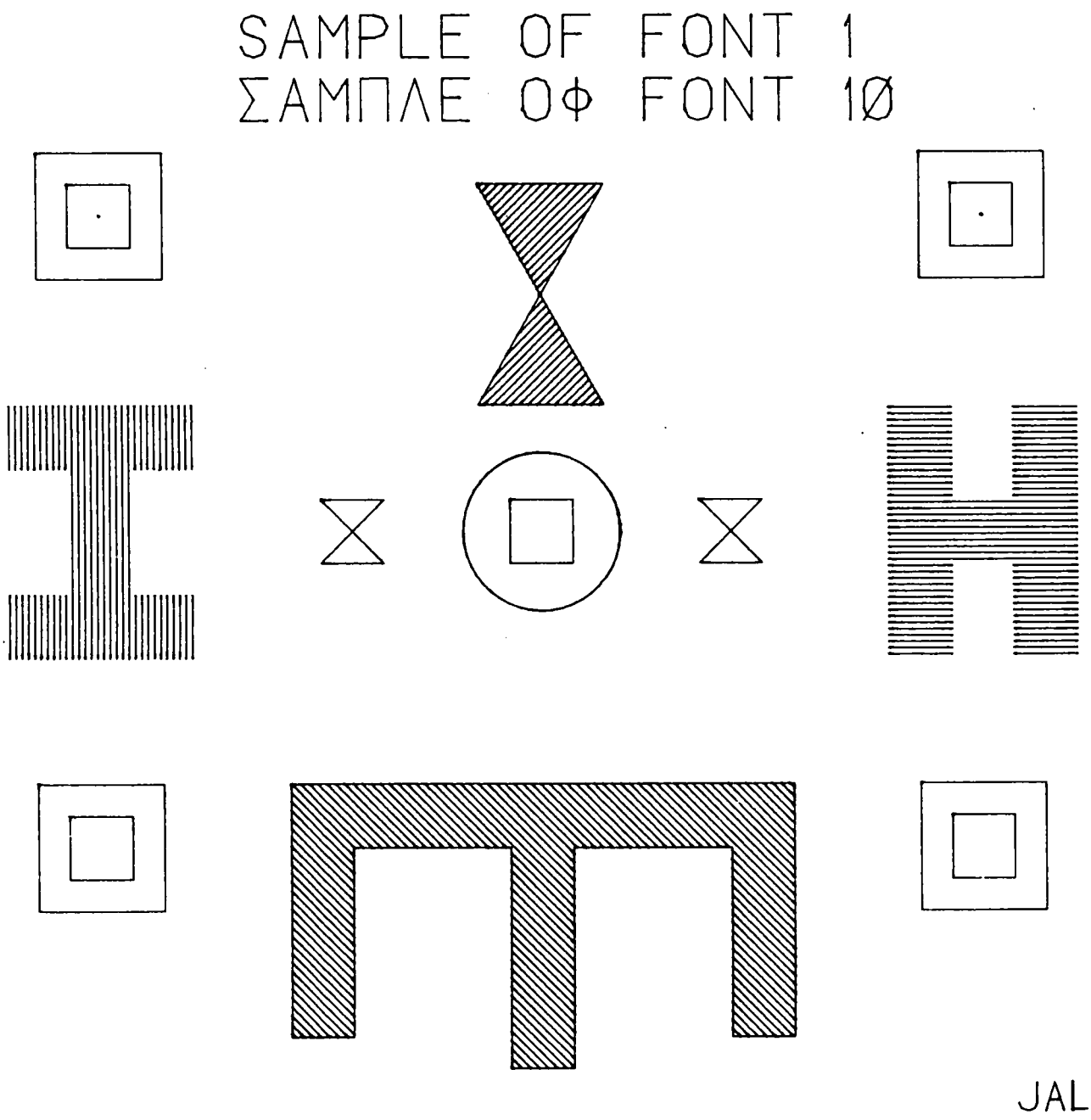


Figure 8. DEMO/PDB/SAMPLE

TITLE

PICTURE30

ABSTRACT NUMBER

Table 4
LISTING OF COMPONENTS FOR FIGURE 8

```
PDB:  #DEMO/PDB/SAMPLE

 2 #6 1 2000 2000 250 0 360 0 1 1
 3 #2 1 1050 3500 0  75 0 10 SAMPLE OF FONT 1 1 1
 4 #2 1 1050 3300 0  75 0 10 SAMPLE OF FONT 10 1 1
 5 #7 1 1000 3100
 6 #3 1 1000 3100 4  45 20
 7 #7 1 1200 1200
 8 #3 1 1200 1200 12  135 20
 9 #7 1 400 3200
10 #7 1 3200 3200
11 #7 1 3200 1200
12 #7 1 400 1200
13 #7 1 600 3100
14 #7 1 3300 3100
15 #7 1 3300 1100
16 #7 1 600 1100
17 #7 1 1300 2100
18 #7 1 1900 2100
19 #7 1 2500 2100
20 #7 1 600 3000
21 #7 1 3400 3000
22 #2 1 3650 150 0  50 0 10 JAL 1 1
23 #7 0 300 2400
24 #7 0 3100 2400
25 #3 1 300 2400 12  90 20
26 #3 1 3100 2400 12  0 20
```

Most of the above components are polygons. Elements 23 and 24 are the deleted (layer 0) polygons used to generate the crosshatched letters I and H, elements 25 and 26. Note that in their listing, the two text samples, elements 3 and 4, appear to be identical. In reality, they are not. Invisible in the listing is a leading nonprinting control N (N), which activated auxiliary font 10 and a trailing, nonprinting control O (O), inserted between "OF" and "FONT 10," which deactivated the auxiliary font.

8. PLOTTING A PICTURE

Another feature of PICTURE30 is its interactive and versatile plotting capabilities. The user is prompted through the plotter setup and enters the required information. Some advice is also given on setting up the plotter. A sample dialog is reproduced below:

```
SET PLOTTER MENU FOR, SHEET, SIZE, DRAFTING, HORIZONTAL OR VERTICAL AND FULL PAGE.
USE NORMAL (N), MIRROR (M) OR SCALED (S) PLOTTING? N
NORMAL (N) OR HORIZONTAL PLOT ON VERTICAL SHEET (V)? V
                SET 'INITIAL AXIS ORIENTATION' TO COLUMN 5 ON PLOTTER MENU CARD,
                                VERTICAL X AND HORIZONTAL Y.
ENTER A SHEET SIZE (A,B OR C), A
ENTER A SCALE FACTOR (MAX OF 1 IS TYPICAL), 1
                PREPARE PLOTTER, 'RETURN' TO CONTINUE.
```

It is assumed that the user is familiar with the plotter and has already set the other necessary parameters. Plotter parameter settings must then match program prompt entries.

TITLE	ABSTRACT NUMBER
PICTURE30	

Three types of plots are available: normal, mirror, and scaled. The normal plot will center the picture on the sheet for a specified size and orientation. At an arbitrary scale of 1, the picture will be plotted at the largest practical size for the specified sheet size. Reducing the scale will shrink the picture size accordingly. A "mirror" plot is also provided to accomodate mirrored displays of axisymmetrical picture layers and, otherwise, performs like the normal plot. Thus normal and mirrored layers can be overlayed with a common center on the plotter as well as on the screen. Creating a picture in layers allows for changing pen colors when plotting the individual layers for a colorful picture.

These plotting modes usually employ the standard 4663 plotter menu Initial Axis Orientation setting for horizontal X and vertical Y plot directions. This is the selection of column 1 for that menu line. With this selection, the plot directions automatically rotate for horizontal or vertical sheet orientation. The X direction is maintained as the long axis of the sheet regardless of its orientation. The plotting routine also allows the user the option of using a vertical sheet orientation with a horizontal, left-to-right X axis plot orientation. This is illustrated in the above dialog. This option requires use of vertical page orientation, Initial Axis Orientation selection of column 5, vertical X and horizontal Y axis directions and the prompt entry shown above. This option is not available to the scaled plot mode described below.

Also included in the plotting routine is the capability of generating a scaled plot whose origin is in the lower left corner of the sheet. Pictures created for scaled plotting should also use that corner of the screen viewport as their origin. This plotting mode is most useful for reconstructing scaled drawings which have been digitized by the graphics tablet. By coordinating the original workspace size with the original scale of the digitized drawing and the plotting prompt entry, an accurately enlarged, same size, or reduced plot of the drawing can be made. In the scaled plot mode, the program tests the sheet size, original workspace size, and scale entries to determine whether the plot will fit onto the selected sheet size.

The following dialog is typical for the scaled plot mode:

```
SET PLOTTER MENU FOR: SHEET, SIZE, DRAFTING, HORIZONTAL OR VERTICAL AND FULL PAGE.
USE NORMAL (N), MIRROR (M) OR SCALED (S) PLOTTING? S
NORMAL (N) OR HORIZONTAL PLOT ON VERTICAL SHEET (V)? N
ENTER A SHEET SIZE (A,B OR C): B
                PLOTTING SPACE AVAILABLE IS 15.75 X 10.25 INCHES.
                (LARGEST X DIMENSION)X(SCALE) MUST FIT INTO THIS SPACE.
                OVERSIZED PLOTS WILL BE CLIPPED AT THE TOP.
ENTER ORIGINAL WORKSPACE X,Y SIZE IN INCHES, SCALE : 20,20,1
TOO LARGE -- TRY AGAIN!
                SPACE NEEDED WOULD BE 20 X 20 INCHES.
SET PLOTTER MENU FOR: SHEET, SIZE, DRAFTING, HORIZONTAL OR VERTICAL AND FULL PAGE.
USE NORMAL (N), MIRROR (M) OR SCALED (S) PLOTTING? S
NORMAL (N) OR HORIZONTAL PLOT ON VERTICAL SHEET (V)? N
ENTER A SHEET SIZE (A,B OR C): C
                PLOTTING SPACE AVAILABLE IS 21 X 15.5 INCHES.
                (LARGEST X DIMENSION)X(SCALE) MUST FIT INTO THIS SPACE.
                OVERSIZED PLOTS WILL BE CLIPPED AT THE TOP.
ENTER ORIGINAL WORKSPACE X,Y SIZE IN INCHES, SCALE : 20,20,1
                PREPARE PLOTTER. 'RETURN' TO CONTINUE.
```

TITLE

PICTURE30

ABSTRACT NUMBER

In the above example, the plotting space available for the initially selected B sheet was displayed, but was deliberately ignored. The selected workspace and scale overflowed the available plotting space and was rejected with an error message which included the plotting space required. Selection of the C size sheet then satisfied the minimum acceptable plotting space. The program will accept a combination that overflows the plotting space at the top edge, but not at the right edge of the sheet. Overflow of the top edge will, of course, clip the drawing along that bound.

The plotting scale is determined by coordinating the original workspace size with the plotting space available on the selected sheet size. Note that a 20 x 20 in. workspace on the tablet, when projected into the 10 x 10 in. screen viewport results in a screen scale of 0.5. The original workspace can later be reconstructed on the plotter by using a C size sheet and responding to the prompt, "Enter original workspace X, Y in inches, scale:" with "20, 20, 1." This entry reproduces the original workspace size within the 21 in. of plotting space available in the X direction but could cause clipping at the top of the plot if the available 15.5 in. is exceeded. Similar scaling methods can be used with B and A size sheets and other tablet workspaces.

9. PROGRAM DATA STRUCTURE

PICTURE30 operates with an absolute minimum of main memory space devoted to the active PDB. Only the currently selected object resides in main memory. This tactic conserves memory capacity, making more of it available to the resident program. While in refresh, the object data resides in the independent 32 Kbyte capacity Option 30 refresh memory. Text consumes a large amount of this memory compared to other objects, because of the many short vectors needed to stroke generate each character. Thus, a large amount of text is more likely to overflow the Option 30 memory than a complex collection of other objects. When fixed, the refresh object is reduced to a minimum of main memory space and is immediately stored on the 4907 disc. With some commands, the object data remains in the refresh memory for immediate reuse.

Most of the key program operating parameters reside in two principal variables, the S (SELect) array and a corresponding, but transient, R (REFresh) array. When first started, the program loads the S array with the default SELect values, which can later be altered with the SELect command. Initially, the R array is also set to the same default values. The R array parameters are those temporarily altered by operators while an object is displayed and manipulated in refresh, and those stored with the object when it is Fixed. The R parameters are then reset to the current S parameters following the completion of each command execution.

The most frequently accessed operating parameters are stored in the S array, and most are those altered by execution of an appropriate SELect command. They are semi-permanent, retaining their currently selected values until the program is restarted or they are deliberately altered. RUN, REStart, or the use of UDK11 restores them to their initial default values. PAUse followed by a UDK1 RESume preserves the current S parameter assignments. A few most frequently altered SELect parameters have been made accessible via UDKs as described in "Program Operation."

Some operating parameters as defined in the original program may require adjustment to conform to the user's system operating profile. The most likely ones are the GPIB addresses for the plotter and the graphics tablet. Program lines for these devices are as follows:

TITLE	ABSTRACT NUMBER
PICTURE30	

Plotter, Printer:
48 O1=[user's plotter address]
17540 GO TO 1*(O1=[user's plotter address])+2*(O1=[user's printer address]) etc.

Tablet:
15870 INPUT @ [user's tablet address]:X3,Y3,K$

REFresh and SELect Array Variables

R/S	Contents	Default
(1)	Current PDB scale	1
(2)	Current PDB rotation, degrees	0
(3)	Current PDB width factor	1
(4)	Current PDB Y factor	1
(5)	Current PDB X factor	1
(6)	Auxiliary character font(0-12)	10
(7)	Layer number (1-100)	1
(8)	Jump-to grid flag, OFF=0, ON=1	1
(9)	Line dash font, default is solid	0
(10)	Arrow head size, UDU	50
(11)	Arrow head angle, degrees	35
(12)	Label text rotation, degrees	0
(13)	Label text height, UDU	50
(14)	Label italics flag, OFF=0, ON=1	0
(15)	Crosshatch angle degrees	45
(16)	Crosshatch line displacement, UDU	20
(17)	Arc starting angle, degrees	0
(18)	Arc ending angle, degrees	0
(19)	Circle radius, UDU	250
(20)	Circle rotation, degrees	0
(21)	Circle width factor	1
(22)	Circle Y factor	1
(23)	Circle X factor	1
(24)	Sine of sub-PDB rotation	0
(25)	Cosine of sub-PDB rotation	1
(26)	Sine of circle rotation (20)	0
(27)	Cosine of circle rotation (20)	1
(28)	Digitizer input device, 0=joystick, 1=tablet	0
(29)	Polygon function:0=NOR, 1=MIR-Y, 2=MIR-X	0

TITLE

PICTURE30

ABSTRACT NUMBER

User Defined Transformation Functions

Function	Transformation
FNA	Scale, rotate, and translate object X coordinate
FNB	Scale, rotate, and translate object Y coordinate
FNC	Scale and rotate circle X coordinate
FND	Scale and rotate circle Y coordinate
FNE	Mirror polygon X coordinate
FNF	Mirror polygon Y coordinate
FNI	Select jump-to grid X coordinate value
FNH	Horizontal jump-to grid ON/OFF
FNZ	Select jump-to grid Y coordinate value
FNV	Vertical jump-to grid ON/OFF

Index to Routines

TITLE	ABSTRACT NUMBER
PICTURE30	

8000	Repeat circle/polygon
8230	Draw dashed line
8860	ROM character generator
9560	Redraw PDB
10070	Generate and draw xhatch
10570	Find sub-PDB
10700	Insert sub-PDB
12040	Store sub-PDB
12130	Store arrow
12230	Draw arrow
12410	Translate and store LIN/PDB/CIR/POL
12890	Find closest object
13400	Alter sub-PDB
13460	Alter arrow
13530	Draw circle
13800	Store circle
13855	Angle adjustment
14000	Check disc space
14090	Prompt position
14200	Draw viewport
14320	Alter circle
14390	Insert polygon
14780	Transform tablet coordinates
14850	Draw polygon
15020	Read polygon
15090	Store polygon
15180	Alter polygon
15420	Polygon point change
15760	Polygon MOV/POI
15860	Read tablet cursor
15925	Mark work area
16000	Select tablet or joystick
16110	Polygon tablet input
16350	Polygon axis
16440	Polygon rotate/mirror
16520	Build crosshair
17000	Set cursor
17100	Clear cursor
17200	Keyboard line entry
17500	Plotter setup
18200	Plot
19000	Mirror display
19200	Display origins
19500	Pack polygon data
19600	Unpack polygon data
19700	Pack xhatch data
19800	Unpack xhatch data
20000	Compress

TITLE

PICTURE30

ABSTRACT NUMBER

Table of Variables

.	A0	.	A2	A3	A4	.	.	A7	.	A9	A$
.	.	B1	B2	B3	B4	B5	B6	.	.	.	B$
.	.	C1	C2	C3	C4	C5	C6	C7	C8	C9	C$
.	.	.	.	.	.	D5	D6	D7	.	D9	D$
.	.	E1	E2	.	.	.	E6	.	.	.	.
.	.	F1	.	.	.	.	.	.	.	.	F$
.	G0	G1	G2	G3	G4	.	.	.	.	.	.
.	.	H1	H2	H3	.	.	.	.	.	H9	.
.	I0	.	.	.	.	I5	I6	I7	I8	I9	.
.	J0	.	.	.	.	.	.	.	.	.	.
.	K0	.	.	.	.	.	.	.	.	.	K$
.	.	L2	.	.	.	.	.	.	.	.	
.	M0	.	.	M3	M4	.	.	.	.	.	M$
.	N0	N1	.	N3	N4	.	.	.	.	.	.
.	.	O1	.	O3	.	.	.	.	.	.	.
P	P0	P1	P2	P3	.	P5	.	P7	P8	.	.
.	.	.	.	.	.	Q5	Q6	Q7	.	.	.
R	.	.	.	.	.	.	.	.	.	.	.
S	.	.	.	.	.	S5	S6	.	.	S9	.
.	.	T1	T2	.	.	.	.	.	.	.	T$
.	U0	.	.	.	.	U5	U6	U7	U8	.	.
V	V0	.	.	.	.	.	.	.	.	.	.
.	W0	W1	W2	W3	W4	.	.	W7	.	.	.
.	X0	X1	X2	X3	X4	X5	X6	X7	X8	X9	.
.	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7	Y8	Y9	.
.	Z0	Z1	.	Z3	.	.	.	.	.	.	Z$

Variable Assignments

Variable	Description
A0	Dummy loop index, scratch
A2	Scale factor for plotter, X; compress
A3	Scale factor for plotter, Y; compress
A4	Plotter keyboard scale entry
A7	Flag: Line=1, Arrow=2, PDB=3
A9	Object identifier for branching; 1-7
A$	Parsed mnemonic from command string
B1	Mirror polygon, X coordinate; compress
B2	Mirror polygon, Y coordinate; compress
B3	Arrow; compress
B4	Arrow; compress
B5	Arrow; compress
B7	Arrow; compress
B$	Command string, scratch

TITLE	ABSTRACT NUMBER
PICTURE30	

C1	Object identification code; 1-7
C2	Position of operator in M$
C3(20)	Polygon point X array; packed polygon vertex array
C4(20)	Polygon point Y array; packed xhatch vertex array
C5	Lower left tablet workspace X coordinate
C6	Lower left tablet workspace Y coordinate
C7	Upper right tablet workspace X coordinate; tablet X scale
C8	Upper right tablet workspace Y coordinate; tablet Y scale
C9	Cosine of xhatch angle
C$	Label text string
D5	Dashed line generator: horizontal length
D6	Dashed line generator: vertical length
D7	Dashed line generator: dash vector
D9	Disc drive unit number, global
D$(160)	Message string, scratch
E1	Error checking flag, alphanumeric; 0=OK, 1=NG, global
E2	Error checking flag, comma; 0=found, 1=none, global
E6	Object found flag; 0=found, 1=missing, global
F1	Library disc mounted flag; 0=yes, 1=no, global
F$(60)	sub-PDB file identifier, global
G0	Jump-to grid X interval, global
G1	Jump-to grid Y interval, global
G2	Holder for G0 during grid display
G3	Holder for G1 during grid display
G4	Holder for *n* during DIS GRD *n*/DIS AXS *n*
H1	Type of plot: N, M, or S; compress
H2	Sheet size: A, B, or C; compress
H3	Sheet orientation: N, V; compress
H9	Cosine, arrow head angle
I0	Loop index, scratch
I5	Counter, number of objects in current PDB, scratch
I6	Counter, number of vertices in crosshatch array
I7	Counter, crosshatch generator
I8	Loop index, scratch
I9	Loop index, scratch
J0	Crosshatch generator
K0	Crosshatch generator
K$	Command line string, scratch
L2	Layer number, scratch
M0	Change flag: 0=INS, 1=MOV, 2=POI, 3=REP, 4=TRA, global
M3	Translate X displacement
M4	Holder for X translation entry
M$(30)	Active operator string

TITLE	ABSTRACT NUMBER
PICTURE30	

N0	(Number of crosshatch vertices)-1
N1	Number of translations
N3	Translate Y displacement
N4	Holder for Y translation entry
O1	Output device address (GPIB), global
O3	Prompt text line position, global
P(2,20)	Crosshatch X,Y vertex array
P0	Branch parameter, scratch
P1	Position of comma in command line or period in file name
P2(20)	Polygon X coordinate array
P3(20)	Polygon Y coordinate array
P5	Number of objects in a current PDB, global
P7	Number of objects in a sub-PDB for insertion
P8(2)	Holder for crosshatching line points
Q5	Holder for first command line numeric, scratch
Q6	Holder for second command line numeric, scratch
Q7	Holder for crosshatch Y min coordinate
R(29)	Refresh parameter storage array
S(29)	Select parameter storage array, global
S5	Dashed line generator, 1% of current X window
S6	Dashed line generator, 1% of current Y window
S9	Sine of current crosshatch angle
T1	Tolerance limit for object selection, global
T2	Tolerance flag; default=0=ON, 1=OFF, global
T$(60)	Current PDB file identifier, global
U0(N0)	Crosshatch X coordinate array
U5-U8	Crosshatch generator functions
V(4)	Viewport coordinate array for plotting
V0	Arrow length vector
W0	Holder for crosshatch Y max coordinate
W1-W4	Current window, global
W2(2,N0+1)	Crosshatch rotated line end point array
X0	Transformed X coordinate, scratch
X1-X8	Object X coordinates, scratch
X9	X coordinate for transformation, scratch
Y0	Transformed Y coordinate, scratch
Y1-Y8	Object Y coordinate, scratch
Y9	Y coordinate for transformation, scratch
Z0	Autopage flag; default=0=ON, 1=OFF, global
Z1	Disc space/List variable/Display flag; default=0=all, 1=layer *n*
Z3	Record number, scratch
Z$(1)	General purpose, scratch

TITLE	ABSTRACT NUMBER
PICTURE30	

10. 4052R11 CHARACTER GENERATOR ROM PACK

The 4052R11 provides a wide variety of stroke generated text fonts, as described in the following pages. PICTURE30 normally uses default font 0 for its text generation. It also makes available another 12 auxiliary character fonts for text variety, all created by the character generator ROM. One of these other available character fonts can be called by the SELect FONt *n* command or by the operator T. Upon selection, an auxiliary font can be interwoven within the standard default font 0 by the use of control characters imbedded into the label text command line string.

The auxiliary font is enabled by entering a control N (N) before the desired text and disabled by the entry of a control O (O) following the text. Thus, mixed font characters can be combined within a single label text command. Prefixing the text height, operator H, with a minus sign (−) while the label is displayed in refresh converts the normal left-to-right horizontal orientation to a top-to-bottom vertical orientation.

The 4052R11 is also used to provide several enhance refresh graphic functions not otherwise available. It allows combining, in refresh, an object with its own software generated cursor, as is used with INSert CIRcle and with INSert POLygon.

The available character fonts are illustrated in Appendix A.

11. REFERENCES

1. Application Library Program, Tektronix Inc., Information Display Division, Group 451, P.O. Box 500, Beaverton, OR 97077.
2. Bishop Graphics, Inc., 20450 Plummer St., Chatsworth, CA 91311.

12. PROGRAM CHANGES AND COMMENTS

The present version (V-3A) of PICTURE30 contains some changes made after the above documentation was prepared.

1. The command line prompt (page 9) now includes an indication of the currently active PDB layer. The format is "*n*>" or "*n*+>" where *n* is the layer number. As before, the "+" signals that the jump-to grid is active.
2. The CHanGe confirmation prompt (page 13) now indicates the layer number in which the object was found. The format is "Layer: *n* Confirm ?" where *n* is the layer number. This feature simplifies the selection of identical objects having the same origins, but located in different layers.
3. INSertion of a polygon consisting of a single point is not displayed in refresh. It does store, however, and will be seen after a DISplay.
4. The jump-to grid must be disabled to INSert a sub-PDB at the exact location as displayed. Otherwise it will be moved to the nearest grid location when redrawn.
5. Files in the DRIVE library are all sub-PDBs that are called by some of the demonstration programs.
6. Unequal **x** axis and **y** axis scale factors will automatically disable **R**otation when INSerting a sub-PDB (page 17). The result is a sub-PDB rotation of 0°.
7. All sub-PDB LABel text must be within a 4000 UDU diameter circle inscribing the viewport to assure correct insertion at all angles of sub-PDB rotation. Text outside of this circle will rotate correctly at 90, 180, and 270°, but may fail at some other angle, thus disrupting the sub-PDB.
8. Program line number 10840 selects only layer 1 for INSert PDB. This line can be changed by the user, if desired, to include other layers.

TITLE
PICTURE30

ABSTRACT NUMBER

9. Simple circles and arcs are now drawn faster using the 4052R11 ROM firmware routines.
10. End of file (EOF) error trapping has been included to eliminate the problem described on page 10.
11. A new command has been added:

MAGNIFY

MAG *n* $n=1,2,4,5,8,10,16,20$

MAGnify fills the viewport with an enlarged area centered about a specific point. The amount of enlargement is determined by the magnification factor *n*. The crosshair selects the point about which the area becomes centered. MAGnify, by setting a new, smaller workspace window, is most useful for inserting fine detail into a picture or for close examination of existing detail.

Entering MAG *n* invokes the pointer crosshair which is then centered on the desired point in a display. A RETURN selects the point and clears the display. The program returns to the command mode ready to receive any command except DIGitize, WINdow, or DISplay MIRror. A DISplay ALL, for example, will display all detail contained within the enlarged area surrounding the selected point. Successive entries of MAG *n* and DIS can be used to progressively enlarge or reduce a displayed area. Horizontal, vertical, or diagonal panning of an enlarged area can also be done with a repeating sequence of MAG *n*, crosshair positioning, and redisplay operations, with a constant value of *n*.

Care must be used when the selected point is near a viewport boundary. When enlarged, especially at low magnification, part of the area may fall outside the normal working window. Detail entered into this outside area will be lost later when the normal working window is restored. A warning message is displayed in refresh when this condition can occur.

MAGnify is cancelled by entering MAG 1 or, with the crosshair displayed, by entering a C instead of a RETURN. The use of DISplay AXS is not recommended with an enlarged area, as inconsistent axes may result. When in use, the magnification factor is displayed in the upper right corner of the screen.

The new routine begins at line 9100 and introduces a new global variable, A1, the magnification factor *n*.

TITLE

PICTURE30

ABSTRACT NUMBER

Appendix A

```
 ! " # $ % & ' ( ) * + , - . /
Ø 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z [ \ ] ^ _
` a b c d e f g h i j k l m n o
p q r s t u v w x y z { | } ~
```

A. FONT 0 CHARACTER SET.

FONT	35	48	64	91	92	93	94	123	124	125
Ø:	#	Ø	@	[	\	]	^	{	|	}
1:	#	Ø	@	Ä	Ö	Å	^	ä	ö	å
2:	£	Ø	@	Ä	Ö	Ü	^	ä	ö	ü
3:	£	Ø	@	[	\	]	^	{	|	}
4:	#	Ø	@	¡	Ñ	¿	^	{	|	}
5:	#	Ø	§	[	\	]	↑	←	↙	→
6:	#	Ø	@	[	\	]	↑	{	|	}
7:	SAME AS FONT Ø									
8:	£	O	@	[	\	]	↑	{	|	}
9:	#	O	@	Æ	Ø	Å	^	æ	ø	å

ASCII DECIMAL EQUIVALENT

B. VARIATIONS IN FONTS 1-9 (compared to Font 0; remainder of characters are the same as those in Font 0).

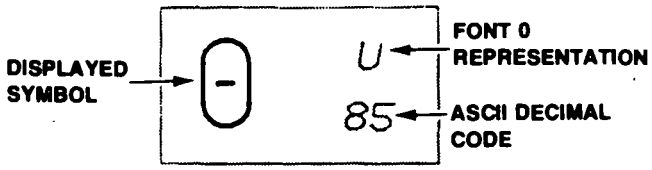
3794-8A

Fonts 0 – 9.

TITLE

PICTURE30

ABSTRACT NUMBER



	SP 32	Ø	0 48	@	@ 64	Π	P 80	\	` 96	π	p 112
!	! 33	1	! 49	A	A 65	Q	Q 81	α	a 97	q	q 113
"	" 34	2	2 50	B	B 66	P	R 82	β	b 98	ρ	r 114
#	# 35	3	3 51	Ψ	C 67	Σ	S 83	ψ	c 99	σ	s 115
$	$ 36	4	4 52	Δ	D 68	T	T 84	δ	d 100	τ	t 116
%	% 37	5	5 53	E	E 69	Θ	U 85	ε	e 101	ϑ	u 117
&	& 38	6	6 54	Φ	F 70	Ω	V 86	φ	f 102	ω	v 118
'	' 39	7	7 55	Γ	G 71	W	W 87	γ	g 103	ς	w 119
(	(40	8	8 56	H	H 72	X	X 88	η	h 104	χ	x 120
)	) 41	9	9 57	I	I 73	ϒ	Y 89	ι	i 105	υ	y 121
*	* 42	:	: 58	ϑ	J 74	Z	Z 90	ξ	j 106	ζ	z 122
+	+ 43	;	; 59	K	K 75	[	[91	κ	k 107	{	{ 123
,	, 44	<	< 60	Λ	L 76	\	\ 92	λ	l 108	∂	| 124
-	- 45	=	= 61	M	M 77	]	] 93	μ	m 109	}	} 125
.	. 46	>	> 62	N	N 78	^	^ 94	ν	n 110	~	~ 126
/	/ 47	?	? 63	O	O 79	_	_ 95	ο	o 111		DEL 127

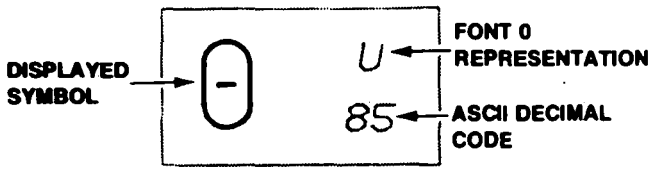
3704-9A

Font 10.

TITLE

PICTURE30

ABSTRACT NUMBER

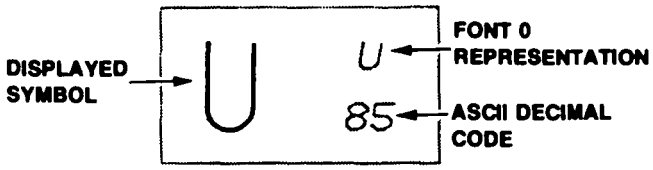


	SP 32	0	0 48	2	@ 64	Π	P 80	-1	` 96	π	p 112
·	! 33	∫	1 49	A	A 65	√	Q 81	α	a 97	—	q 113
□	" 34	∴	2 50	B	B 66	P	R 82	β	b 98	ρ	r 114
○	# 35	∃	3 51	Ψ	C 67	Σ	S 83	ψ	c 99	σ	s 115
△	$ 36	⊂	4 52	Δ	D 68	T	T 84	δ	d 100	τ	t 116
+	% 37	∪	5 53	E	E 69	Θ	U 85	ε	e 101	ϑ	u 117
◇	& 38	⊃	6 54	Φ	F 70	Ω	V 86	φ	f 102	ω	v 118
⊠	' 39	∩	7 55	Γ	G 71	W	W 87	γ	g 103	ς	w 119
⌑	(40	∞	8 56	H	H 72	X	X 88	η	h 104	χ	x 120
⊕	) 41	∇	9 57	I	I 73	ϒ	Y 89	ι	i 105	υ	y 121
⊞	* 42	:	: 58	⁻	J 74	Z	Z 90	ξ	j 106	ζ	z 122
+	+ 43	≠	; 59	K	K 75	[	[91	κ	k 107	ℓ	{ 123
⊥	, 44	≤	< 60	Λ	L 76	±	\ 92	λ	l 108	∂	| 124
‖	- 45	≡	= 61	M	M 77	]	] 93	μ	m 109	↓	} 125
.	. 46	≥	> 62	N	N 78	↑	^ 94	ν	n 110	~	~ 126
÷	/ 47	∝	? 63	O	O 79	_	_ 95	o	o 111		DEL 127

3794-10A

Font 11.

TITLE	ABSTRACT NUMBER
PICTURE30	



SP 32	0 48	@ 64	P 80	` 96	p 112
! 33	1 49	A 65	Q 81	a 97	q 113
" 34	2 50	B 66	R 82	b 98	r 114
# 35	3 51	C 67	S 83	c 99	s 115
$ 36	4 52	D 68	T 84	d 100	t 116
% 37	5 53	E 69	U 85	e 101	u 117
& 38	6 54	F 70	V 86	f 102	v 118
' 39	7 55	G 71	W 87	g 103	w 119
(40	8 56	H 72	X 88	h 104	x 120
) 41	9 57	I 73	Y 89	i 105	y 121
* 42	: 58	J 74	Z 90	j 106	z 122
+ 43	; 59	K 75	[91	k 107	{ 123
, 44	< 60	L 76	\ 92	l 108	| 124
- 45	= 61	M 77	] 93	m 109	} 125
. 46	> 62	N 78	^ 94	n 110	~ 126
/ 47	? 63	O 79	_ 95	o 111	DEL 127

3704-11A

Font 12.

Disk-to-Tape Backup/Restore, Utilities program 51/07-8048/00 written by John H. Grant, Tektronix, Inc.

The DISK-TO-TAPE BACKUP/RESTORE UTILITIES will archive disk files onto tape and restore them to disk. Specific capabilities include the following:

- o All file types supported by the 4907 File Manager ROM pack may be archived and restored; i.e., ASCII, binary, and host files, as well as random and sequential file formats. All files are stored as binary data on the backup tape(s) and the RESTORE utility will be required to recover most files from the tape(s).

- o Multiple volume backups are supported; i.e., if the information on the disk exceeds the capacity of a single tape, more than one tape will be used.

- o Directory editing for total backups is supported: files may be omitted from the backup process and file names which require passwords may (must) be modified to include the password.

- o Selective backups and restores are also supported: information stored on a single file or a group of files may be archived or restored individually. Using selective backup/restore can significantly reduce the time required to archive and restore files. The original tape file will be overwritten when it exists; however, the routine checks the size of a file before attempting to backup the information on the same tape file.

- o If a 4631 Hard Copy Unit is available, hard copies of the archived and restored file names will automatically be generated.

The DISK-TO-TAPE BACKUP/RESTORE programs were designed specifically for owners of a single-drive 4907 who have a need to backup information from their disks, but they may also be used in multiple-drive configurations.

OPERATION

The disk-to-tape backup/restore utilities include the following:

```
$BACKUP
$BACKUP.BØ1
$BACKUP.BØ2
$BACKUP.BØ3
$BACKUP.RØ1
$BACKUP.RØ2
$BACKUP.RØ3
```

Four programs are provided: total backup, selective backup, total restore, and selective restore. Any one of the four programs may be accessed via the driver program, named $BACKUP.

To run the DISK-TO-TAPE BACKUP/RESTORE UTILITIES insert disk in drive and:

```
CALL "MOUNT",Ø,A$
OLD "$BACKUP"
RUN
```

You will be prompted for the disk drive unit number (usually Ø), whether you wish to backup "B" or restore "R" a disk and whether the operation is to be a total "T" or selective "S". Default responses throughout the program will be underlined (e.g., B/R indicates taht any entry other than "R" - for restore - will result in "B" - for backup).

After you have responded to the activity interrogation, the driver program will append in the appropriate files.

TOTAL BACKUP

If you have selected "B" and "T" in response to the activity interrogation, the total backup program ($BACKUP.BØ2) and the backup file copy subroutines ($BACKUP.BØ3) will be appended and executed.

You will be requested to insert the disk to be archived in the appropriate disk drive and to insert you backup tape into the internal tape dirve. Note that any information on your tape will be destroyed.

After a directory has been obtained of the disk, the program will ask if you need to edit the internal directory to delete file names which are not to be archived, or to modify file names to include a password. Note that all files which use passwords must be modified to include the password in the file name specification. Failure to include passwords will result in a program abort.

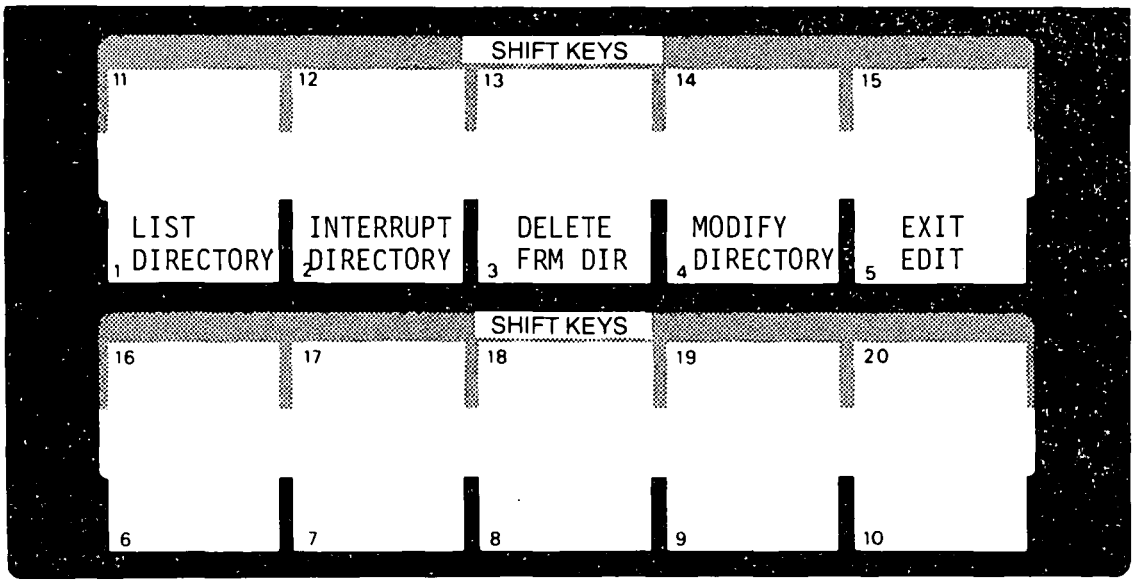
If you have indicated that you need to edit the directory, the user definable keys will be enabled and a menu will be displayed. All edit functions will be accessible via the user definable keys as follows:

UDK 1 -- Causes the current directory to be listed on the screen so that you may determine which file names need to be deleted or modified and to verify any prior editing.

UDK 2 -- Allows you to interrupt the directory listing in the event that the listing is long and you don't need to see it all.

UDK 3 -- Used to access the delete function. You will be prompted for the name of the item to delete. A null response (i.e., just a RETURN) will exit the delete mode.

TITLE TAPE # FILE #



If you specify the file name (or any portion of a file name), the string you enter will be compared against the directory and, in the event of a match, you will be asked to verify the file name found. Upon affirmative verification, the file name will be deleted from the directory. Negative verification will continue the directory compare until either no match is found or until you respond affirmatively.

UDK 4 -- Used to access the modify function and should be used to include passwords where appropraite. You will be prompted for the name of the item to modify. A null response will exit the modify mode. The same compare verification is done for the modify function as for the delete function. After responding with an affirmative verification, you will be prompted to input the file name as it should appear, including passwords. Note that any incorrect file name specification will result in a program abort later.

UDK 5 -- Used to exit the edit mode and to begin the actual archive process.

After the editing has been completed, the actual disk-to-tape backup will begin. The files archived will be displayed on the screen. If all of the disk files cannot be archived on the tape, you will be prompted at the appropriate time to insert another tape.

SELECTIVE BACKUP

If you have selected "B" and "S", the main program will append and execute the selective backup program ($BACKUP.BØ1) and the backup file copy subroutines ($BACKUP.BØ3).

Insert the disk containing the file(s) to be archived in the appropriate disk drive.

The program will mount the disk and ask for the name of the file to be archived. Enter the file name. BE SURE TO SPECIFY @ in front of the name for anything residing in a library other than SCRATCHLIB.

When the disk file has been verified to exist, you will be prompted to insert the tape.

NOTE: If you are using a tape not previously used in any of the backup routines, you must MARK file 1 for 10000 bytes and print the directory into it:

```
FIND 0
MARK 1,10000
FIND 1
PRI @33:"***DISK BACKUP TAPE***"
```

The tape directory is searched for a matching file name. If a match is found and that file on the tape is marked large enough, the disk file is archived and you'll be returned to the file name prompt.

Otherwise, you'll be notified that the file is not currently archived on the tape, or that the marked file is not large enough (whichever is appropriate), and asked if you want a new file created on the tape. If you are certain that enough room remains on the tape, you may respond affirmatively and the disk file will be archived on the current tape. Otherwise, you can respond negatively. In this event, you'll be asked if the file should be archived on another tape. A negative response will return you to the file name prompt. Otherwise, the file name (if it exists) is deleted from the current tape directory and you'll be requested to insert another tape. The preceding procedure is then executed again with the new tape, beginning with a search of the tape directory.

In summary, the program will try to backup the disk file on the current tape. Failing that, the file may be archived on another tape.

TOTAL RESTORE

If you have selected "R" and "T", the total restore program ($BACKUP.RØ2) and the restore file copy subroutines ($BACKUP.RØ3) will be appended and executed.

Insert the disk to be restored in the appropriated disk drive, and insert your backup tape into the internal tape drive. Note that all files on the backup will overwrite files with the same file names on the disk. The files being restored will be displayed on the screen as they are being restored.

When the entire tape has been restored to disk, you'll be asked if there are any further tapes to restore on the same disk.

After the disk has been fully restored, you'll be asked if you have any other disks which need to be restored.

SELECTIVE RESTORE

If you have selected "R" and "S" in response to the activity interrogation, the selective restore program ($BACKUP.RØ1) and the restore file copy subroutines ($BACKUP.RØ3) will be appended and executed.

You will be requested to insert the disk to be restored from in the appropriate disk drive. Note that all files on the disk which are restored from the tape are overwritten.

After the disk has been mounted, you will be prompted for the name of the file to be restored. Any time this prompt is given, you may press UDK 1 to obtain a directory of the tape. Be sure to insert the tape before pressing UDK 1. Also, a null response (i.e., just a RETURN) to this prompt will exit the program.

After you enter the full file name, the tape directory is searched for the entry. If the entry is not found, you will be prompted for a file name. If the file nmae is found in the directory, the disk file will be restored.

DATA FILES

The following disk files are used:

$BACKUP.DIR is a sequential binary ($BACKUP.BØ1) or ASCII ($BACKUP.BØ2, $BACKUP.RØ2) file used as a scratch directory file. The data format (binary vs. ASCII) varies depending on whether or not a carriage return is needed as a record separator.

$BACKUP.DAT is binary sequential file consisting of one record. The file contains the "extended" ASCII alphabet in a binary string 256 bytes long. For example, position 64 of the string contains the ASCII character whose decimal equivalent is 65 - i.e., the character "A".

$BACKUP.DIR is typically assigned to logical file number 1. $BACKUP.DAT is typically assigned to logical file number 2, but only long enough to write to it and read it back. All disk files involved in backup or restore operations are assigned to logical file number 2.

Each directory item stored in $BACKUP.DIR is of the following format:

```
filename$attributes#allocated%used&record length
```

and a carriage return is used to separate items from each other. "Allocated" refers to the files allocated file length and "used" refers to the actual space used by the file. "Record length" is zero for sequential access files.

PROGRAM VARIABLES

The following variables are used in $BACKUP

D	disk drive number
E	end of file flag
N	scratch scalar
M$	carriage return string
O$	overlay file number
Z$	scratch string

The following variables are used in $BACKUP.BØ1

D	disk drive number
E	end of file flag
F	record number counter
F1	record number marker
I	loop variable
N	scratch scalar
U	number of bytes used by the file
A$	input string
B$	scratch string
C$	scratch string
D$	scratch string
E$	scratch string
F$	file name
T$	alphabet string
Z$	scratch string

The following variables are used in $BACKUP.BØ2

CØ	directory starting point for current tape
C1	number of files counter
C2	file copy done flag
D	disk drive number
E	end of file flag
F	file number
F1	bytes used on tape counter
I	loop variable
I1	loop variable
N	scratch scalar
NØ	scratch scalar
PØ	string pointer
U	number of bytes used by the file
A$	directory string
B$	scratch string
C$	scratch string
D$	scratch string
E$	scratch string
F$	scratch string
M$	carriage return string
T$	alphabet string
Z$	scratch string

The following variables are used in $BACKUP.BØ3

D	disk drive number
J	loop variable
N	scratch scalar
A$	output string
B$	scratch string
T$	alphabet string

The following variables are used in $BACKUP.RØ1

D	disk drive number
E	end of file flag
F	file number
N	scratch scalar
A$	scratch string
Z$	scratch string

The following variables are used in $BACKUP.RØ2

D	disk drive number
E	end of file flag
F	file number
N	scratch scalar
A$	scratch string
Z$	scratch string

The following variables are used in $BACKUP.RØ3

A	file allocation size
D	disk drive number
F	file number
I	loop variable
N	scratch scalar
S	record size
A$	scratch string
B$	file attributes
Z$	scratch string

TITLE	PAGE NUMBER
PICTURE30	55

TITLE	PART NUMBER
EDITPROG	062-6715-01 Program 2
ORIGINAL DATE / REVISION DATE	EQUIPMENT, OPTIONS AND SOFTWARE REQUIRED (INCLUDING PERIPHERALS AND HOST SYSTEM)
AUTHOR: John Harms — Tektronix, Inc. Wilsonville, OR	4051/52/54 with 8K 4907 File Manager Optional - 4641/3 Printer

ABSTRACT

Files: 2 Program

Statements: 151

This program will take any number of 4907 host binary files and perform string replacement. For example, a disk with 50 program files can be easily edited. EDITPROG will allow you to change every occurrence of "FOO" to "FUMP", "CAT" to "KITTEN", "A$" to "B$", and/or any other combination of strings.

The filenames are specified in a list in DATA statements, and each name may include wildcard characters. The report that lists each change can be optionally directed to a printer.

The user interface has been left unstructured to allow a programmer to easily change the program for other purposes (e.g., listing all occurrences of a string, etc.). Internal program documentation is in SUPERLIST format (see TEKniques Vol. 6, No. 2 D1).

RESTRICTION: Program line numbers must be less than 65000.

INSTALLATION INSTRUCTIONS

Transfer the EDITPROG files to a working disk as the program disk is also used to store some scratch files.

Transfer using the COPY "@EDITPROG/*" from the original disk to the destination disk.

OPERATING INSTRUCTIONS

To run EDITPROG:

```
OLD "@EDITPROG/MAIN"
```

from the disk.

1. Modify variables D, S, and P (lines 65022-65026) to reflect the desired values:

   D = Printer address. If D=32, output will be to the screen. If D=33, output will go to file #2 of a listing tape. Output may be to a printer.

   S = Scratch unit. This is the disk on which EDITPROG resides; it is also used for some scratch files.

   P = Program unit. This is the disk on which your programs reside.

2. Put your filenames in the DATA statements starting at line 65050. The program is shipped with some example names. Note that wildcards are allowed in the filenames. Terminate the names with a null string.

3. Put your strings to edit starting at line 65300. For each replacement, set S$ = "Search string" and R$ = "Replacement string", then GOSUB 65450. This sequence may be repeated as many times as necessary. When the program is run, every occurrence of "Search string" will be changed to "Replacement string". The program is shipped with some example strings.

4. Run the program.

PROGRAM CHARACTERISTICS

The program takes each host binary file and makes an ASCII copy on file @EDITPROG/TEMP1. This file is then examined line-by-line for occurrences of the search strings specified in Step 3 above. As the strings are found, they are changed to the corresponding replacement string. A new ASCII file @EDITPROG/TEMP2 is constructed during this process that reflects these changes. If at least one change was made, @EDITPROG/TEMP2 is then converted back to a host binary file and is saved on top of the original.

After each file has been edited, file @EDITPROG/RESTORE is appended to restore program lines needed for the ASCII/Binary conversion process.

A listing that shows what changes have been made is sent to the selected listing device. If a line would be too long after replacement (i.e., over 72 characters), that fact is noted on the listing and the original line is not changed.

CAUTION: Be sure that your editing changes do not create invalid BASIC statements. Such a file cannot be converted back to host binary from the ASCII form.

NOTE: In order to allow proper editing of upper/lower case characters, a SET NOCASE is used. This means that "FOOBAR" will not match "Foobar".

TEKTRONIX®

APPLICATIONS LIBRARY PROGRAM

TITLE	ABSTRACT NUMBER
PRODUCTION CAPACITY FORECASTING MODEL	062-6715-01 Program 3
ORIGINAL DATE: MARCH 1982 / REVISION DATE:	MEMORY REQUIREMENT: 32K
AUTHOR: LEONARD WEITMAN/MARK LYMAN	PERIPHERALS: 4907 FILE MANAGER (DUAL) 4631 HARD COPY 4643 PRINTER

ABSTRACT

The purpose of the Capacity Forecasting Model is to provide production managers and engineers with information concerning maximum production capacities within a production area. The mathematical model is based on the interaction of process yields, processing times, and machine utilization rates with the forecasted market demand for the products manufactured in the area. The results generated by the model answer such questions as:

- Where are possible bottlenecks in the production line?
- What equipment will be needed to meet the forecasted production requirements?
- When will this equipment be needed?
- How much additional space will be required?

The computer program is an integrated system of modules performing three main functions:

1. Data Storage: Lets the user create a data base containing the information needed for running the model.
2. Data File Management: Aids in the manipulation of data files. Files may be listed, updated, or deleted. This gives the user flexibility in making changes within the data base.
3. Calculate/List Production Capacities: Draws information from the data files and calculates the results. The listing routine lets the user output the results in a clear, meaningful manner.

The answers to the questions given above, combined with the data storage and management, make the Capacity Forecasting Model a versatile tool when evaluating production areas.

I. PROGRAM INTRODUCTION

A. Hardware Requirements

1. 4052/4054 with 32K memory
2. 4907 File Manager (dual drive)
3. 4641 Matrix Printer (optional)
4. 4631 Hard Copy (optional)

B. Memory Requirements

With a 32K core memory, the model is capable of handling an array with four hundred elements; this means we can handle any problem such that:

(# processes * # products) ≤ 400

Example:

10 products	20 products
*40 processes	*20 processes
400 total	400 total

C. Examples

The examples shown in Section III covering program execution are provided to assist the user in running the program. You should note that all user input is underlined with a solid line and returns are designated by large dots.

Example:

INPUT YOUR NAME: Mark Lyman ●

TITLE	ABSTRACT NO:
PRODUCTION CAPACITY FORECASTING MODEL	062-6715-01 Program 3

II. DATA REQUIREMENTS

A. List of Requirements

Before running the Capacity Forecasting Model, certain data files must be created and filled with appropriate information by the user. The first step is to gather this information from the production area being evaluated. Below is a list of the data requirements.

1. List of products coming out of the production area.
2. List of all equipment used in the production area.
3. Process flowchart showing the yields for each process.
4. Processing time per part required at each process.
5. Production time available and utilization factor for processes.
6. An estimate of future demand for products.

B. Sample Data

The following short example helps explain the above requirements. The sections given correspond to the list of requirements.

Production Area: Large Screen C.R.T.

1. Count the total number of products. For each product, assign a product number beginning at "1" and increasing sequentially.

Product Number	Product Name
1	19" C.R.T.
2	25" C.R.T.

Total Number of Products: 2

TITLE

PRODUCTION CAPACITY FORECASTING MODEL

ABSTRACT NO:

062-6715-01
Program 3

2. Count the total number of processes. For each process, assign a process number beginning at "1" and increasing sequentially.

Process Number	Process Name
1	Drill
2	Vapor Blast
3	Dag Wash

Total Number of Processes: 3

3. For each product, use the process flowchart to compute net yield rates at each process. The net yield rate reflects the percentage of good finished units produced at each process. The net yield rate is found by multiplying individual process yields, beginning at the final process and moving up to the process being evaluated.

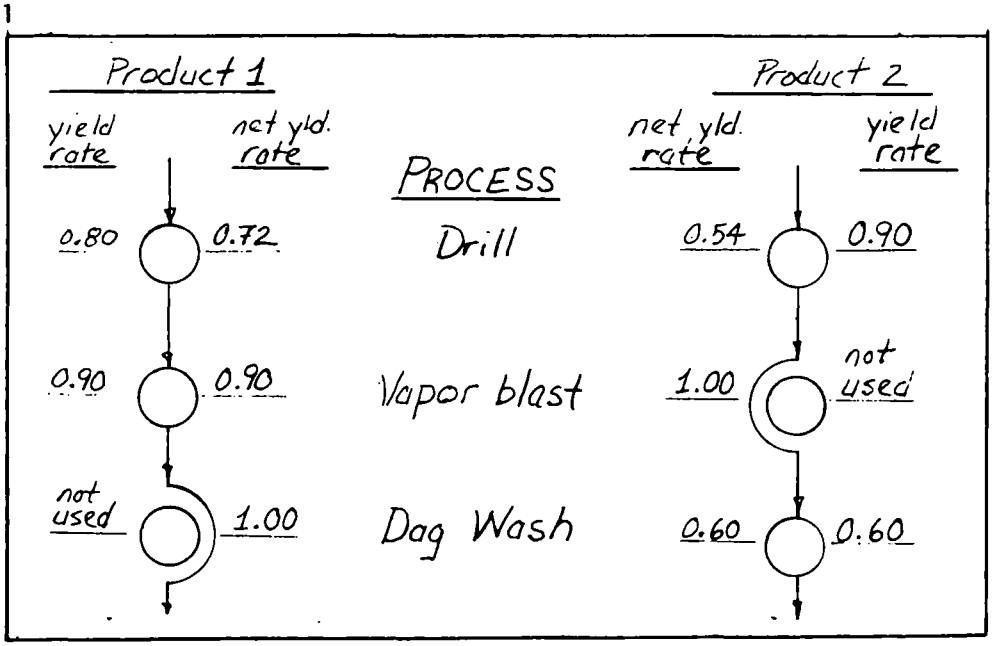
* If a given product does not go through a certain process, assign it a net yield rate of "1.0".

Example: Net Yield Rate: Product 1/Process 1

.9 * .8 = .72

Figure II.1

Process Flowchart



TITLE

PRODUCTION CAPACITY FORECASTING MODEL

ABSTRACT NO:

062-6715-01
Program 3

4. For each process station, find the amount of processing time required in making each finished product. If a given product does not go through a certain process, assign a process time of "0.0". Be able to index the net yield rates and processing times using product and process numbers. An example is given below.

Figure II.2

Yield Rate/ Processing Time Data

Process #/Product #	Net Yield Rate	Processing Time (Min/Prod)
1/1	.72	1.6
1/2	.54	1.6
2/1	.90	2.75
2/2	1.00	0.0
3/1	1.00	0.0
3/2	.60	15.0

5. Find the total amount of production time available for each process. You must account for multiple machines performing the same process and the shifts over which the processes are run. The utilization factor indicates what percentage of time is actually available for processing. This takes into account such things as maintenance and machine breakdowns.

Figure II.3

Processing Time/Machine Utilization Data

Process #	Time Available (Min/Day)	Utilization
1	2520	.9
2	1260	.9
3	1260	.9

6. Future demand - This is crucial to running the model. The future demand must be put into equation form to fit into the capacity model. This is covered in the following section.

TITLE	ABSTRACT NO:
PRODUCTION CAPACITY FORECASTING MODEL	062-6715-01 Program 3

C. Equations

This section is devoted to the formulation of equations necessary in running the program. There are two types of equations necessary:

- o Demand Equations
- o Generalized Capacity Equation

1. Demand Equations

a. Formulation of equations: The forecast of future demand for each product is reflected in the demand equation. Each product should have a corresponding demand equation. The equation should show the expected demand as a function of time. The forecasted demand for each product is generally supplied by the Marketing Department.

Example

Demand for Product 1 (Units/Day) = 25 + 14 * (Time)

25: Current Demand

14: Expected increase in demand for one time unit.

The current demand represents the demand for Product 1 in units/day as it now exists. The future demand shows the expected increase in demand (units/day/year) over a certain time period. (In this example the time unit is per year. To generate a meaningful output, the user must be sure to use compatible units in the processing time, production time available, and demand equations. This is necessary to provide a meaningful unit to the time to reach capacity.

The units should multiply out in the following fashion:

Processing Time * Increasing Demand * (Time) = Production Time Available

(Min./Unit) * (Units/Day/Year) * (Years) = (Min./day)

TITLE	ABSTRACT NO:
PRODUCTION CAPACITY FORECASTING MODEL	062-6715-01 Program 3

Notice that our time value is in respect to years; this means that our output for time to reach capacity is also in years. The key to finding the units for output is to look at the unit of time over which the demand is expected to increase. The units used may be different from the ones shown, but they must be consistent with one another and multiply out in the fashion shown on the previous page.

Example

Time (Years)	Expected Demand Product 1 (Units/Day)
Now	25
1	39
2	53

b. Format of equations: The program allows entry of any form of demand equation; however, the equation must be entered in the proper format for correct program execution. This format is reached by substituting required variable names into the equations.

* It is essential to program execution that the equations are in correct form.

The example below shows the demand equations in rough form, followed by their precise form. The precise form is the one used when entering the equation to the computer. You should have as many demand equations as there are products.

Example

Rough Form: Demand (Product 1) = 25 + 14 * (Time)
Demand (Product 2) = 16 + 5.4 * (Time)

Precise Form: D(1) = 25 + 14 * T(I)
D(2) = 16 + 5.4 * T(I)

D( ): Represents the demand for whatever product number is in the brackets. If you have three products, you should have three demand equations represented by D(1), D(2), and D(3).

TITLE	ABSTRACT NO:
PRODUCTION CAPACITY FORECASTING MODEL	062-6715-01 Program 3

T(I): Represents the time variable. Substitute this directly into the equation wherever the time variable arises.

The example below shows the demand equations on entry into the computer.

Figure II.4

Demand Equations Entered Into The Computer

```
7000 D(1)=25+14*T(I)
7100 D(2)=16+5.4*T(I)
```

* The demand equation does not have to be linear; any equation with time as the variable may be used. Examples of other types of equations are given below.

Other Possible Equations

(These are not related to our sample problem.)

D(5) = 50 - 13 * T(I) - Represents a product with decreasing demand.

D(2) = 25 * (1 + .12) * T(I) - Represents a growth of 12 percent per time period.

More examples of demand equations may be seen in Section V covering references.

2. Generalized Capacity Equation

a. Formulation of equation: The generalized equation is written in a form which allows it to calculate the production capacities for each process/product combination. Only one general capacity equation is required for evaluating a production area. The development of the generalized equation will follow in three examples; a simplified example, a usable example, and finally, the generalized equation in its precise form.

Simplified Equation:

The basic idea behind the capacity equation is simple; given a limited amount of production time and a certain required processing time, there is a limit to the maximum number of units which can be turned out at a process. The equation below shows this:

TITLE	ABSTRACT NO:
PRODUCTION CAPACITY FORECASTING MODEL	062-6715-01 Program 3

Processing Time $\left(\frac{\text{Min.}}{\text{Unit}}\right)$ * Number of Units/Day $\leq$ Production Time Available (Minutes/Day)

Usable Equation

This equation takes into account the different products going through only one process, along with the associated net yield rates and utilization factor. It is in the same form as the simplified equation, but is expanded to include the extra information. This equation could be solved to evaluate the capacity at the drill process.

Example: Process 1 - Drill

Process time for Product 2 @ Process 1

Time available @ Process 1

$$\frac{1.6}{.72} * \left(\text{Units of Product 1}\right) + \frac{1.6}{.54} * \left(\text{Units of Product 2}\right) = (2520 * .9)$$

Each module accounts for a product going through a process. This one is for Product 1 going through drill process.

Net yield rate of Product 2 @ Process 1

Utilization @ Process 1

b. Format for Generalized Capacity Equation:

The usable equation shown in part "a" will solve the capacity at one process. The generalized equation is written to handle all processes. To accomplish this, we must write the equation using a set of required variable names. The following steps should help in writing the generalized capacity equation.

STEP 1: Each product may be viewed as adding a module to the generalized equation. (See the usable equation in part "a"). Each module represents the processing time for that product. For a production area with two products:

TITLE

PRODUCTION CAPACITY FORECASTING MODEL

ABSTRACT NO:

062-6715-01
Program 3

Module Product 1 + Module Product 2 = Production Time Module

STEP 2: Consider the individual module representing Product 1:

$$\text{Module Product 1} = \frac{\text{Processing Time}}{\text{Net Yield Rate}} * (\#\ \text{Units of Product 1})$$

STEP 3: Use the required variable names to convert the equation module in Step 2 into a generalized equation covering all processes.

$$C(I,1) = \frac{\text{Process Time}}{\text{Net Yield Rate}}$$

Product number for module you are working on.

Module Product 1 = C(I,1) * (# Units of Product 1)

STEP 4: Substitute the demand equation, for the product you are looking at, into the equation in Step 3. The demand equation represents the number of units of the product being considered. See Section II.C.1.b. for explanation of demand equations. The demand equation must be entered in its correct format.

D(1) = 25 + 14 * T(I)

(# Units of Product 1) = 25 + 14 * T(I)

Module Product 1 = C(I,1) * [25 + 14 * T(I)]

Product Number

STEP 5: Repeat Steps 2 through 4 for each product module. When you have done this, add the product modules together as shown in Step 1.

For our two product example:

[C(I,1) * (25 + 14 * T(I))] + [C(I,2) * (16 + 5.4 * T(I))] = Prod. time module

Module Product 1

Module Product 2

TITLE

PRODUCTION CAPACITY FORECASTING MODEL

ABSTRACT NO:

062-6715-01
Program 3

STEP 6: Change the production time module into the generalized form. This module accounts for the production time available and the utilization factor for each process.

Production Time Module = T1(I) * U(I)

This module will always equal the variables given above. The variable T1(I) represents the time available at a process, while U(I) represents the utilization factor. The variable I accounts for the different processes.

Substitute into the equation given in Step 5.

C(I,1)*(25+14*T(I)) + C(I,2)*(16+5.4*T(I)) = T1(I) * U(I)

STEP 7: Solve the equation given in Step 6 for the variable T(I).

In our example:

$$T(I) = \frac{[T1(I) * U(I) - C(I,1) * 25 - C(I,2) * 16]}{C(I,1) * 14 + C(I,2) * 5.4}$$

STEP 8: The computer limits entry to a total of 60 characters; if your equation is longer than 60 characters (as almost all are), it must be broken into sections for entry into the computer. Use the variables X1 - X9 or Z1 - Z9 to represent these sections.

For our example:

X1 = T1(I) * U(I)
Z2 = C(I,2) * 16
X7 = C(I,1) * 14

$$T(I) = \frac{((X1) - C(I,1) * 25 - Z2)}{(X7 + C(I,2) * 5.4)}$$

STEP 9: Congratulations! You have finished writing the generalized capacity equation. You must now enter the equation and sections represented by the variables X1 - X9 or Z1 - Z9 into the computer. The equations must be entered in the exact form given, and the capacity equation, solving for T(I), must be entered at a higher line number than the section equations.

TITLE	ABSTRACT NO:
PRODUCTION CAPACITY FORECASTING MODEL	062-6715-01 Program 3

The example below shows our generalized capacity equation after entry to the computer.

FIGURE II.5 LISTING OF CAPACITY EQUATION PROG./FILE

```
12000 X1=T1(I)*U(I)
12010 Z2=C(I,2)*16
12020 X7=C(I,1)*14
12030 T(I)=(X1-C(I,1)*25-Z2)/(X7+C(I,2)*5.4)
```

* One Final Note: The user should remember that the program is designed to handle large problems. In developing a larger model, simply expand the data and equations using the same formats given throughout this section.

TITLE	ABSTRACT NO:
PRODUCTION CAPACITY FORECASTING MODEL	062-6715-01 Program 3

III. PROGRAM OPERATION

This section follows through a series of examples covering the actual running of the program. The data given in Section II is used for the demonstration.

A. Getting on the System

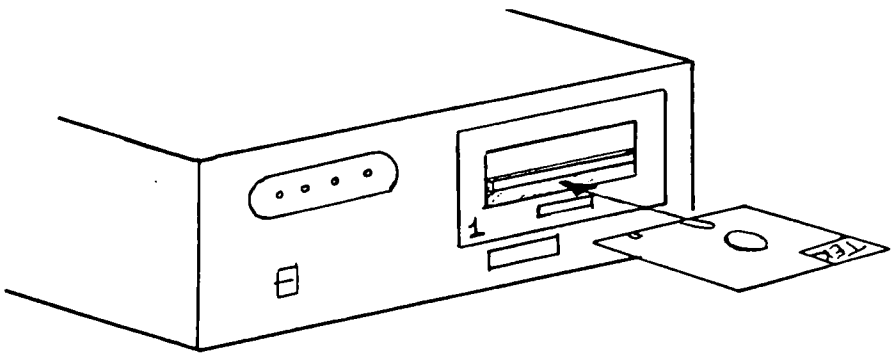
1. System power up: Check the power switches on the equipment. If the power is "off," turn the equipment on in the following order:

   . Second disk drive
   . First disk drive
   . Line printer (optional)
   . 4052/4054

2. Load the disk labeled "PROGRAMS" into disk drive 1. The disk should be inserted in the manner shown. Close the file manager door following insertion.

Figure III.1

Loading the Files Disk Into Unit 0



3. Load the disk labeled "FILES" into disk drive 0 (default drive). Follow the insertion instructions shown above. Make sure the write-protect switch is off. (This is indicated when the red warning light, by the write-protect switch, is off.)

4. An autoload program is listed below. This may be keyed in and saved to tape for an automatic start up from tape to set the clock, mount the disks and load the menu.

```
100 REM             * * * * *  LOADING MAIN DIRECTORY * * * *
110 INIT
115 CHARSIZE 4
120 PAGE
125 PRINT USING "2L5T64(""="")":
130 PRINT USING "L24T""CAPACITY FORECASTING MODEL""":
135 PRINT USING "L5T64(""="")":
138 CHARSIZE 3
140 PRINT "JJRUNNING THIS PROGRAM REQUIRES FAMILIARITY WITH THE SYSTEM."
150 PRINT "PLEASE CONSULT THE OPERATORS MANUAL IF YOU ARE NEW TO THIS"
160 PRINT "PROGRAM."
170 PRINT "JJ  BEFORE STARTING PROGRAM, BE SURE DISKS ARE INSERTED"
180 PRINT "  IN THEIR PROPER DRIVES."
190 PRINT "          DRIVE 0 - FILE DISK"
200 PRINT "          DRIVE 1 - PROGRAM DISK"
210 PRINT "JHAVE ALL START-UP PROCEDURES BEEN FOLLOWED ? (Y/N)";
220 INPUT X$
230 IF X$="N" THEN 1000
240 CALL "TIME",A$
250 IF A$="" THEN 310
260 GO TO 410
310 PRINT "JJ  * THE SYSTEM CLOCK HAS NOT BEEN SET -- THE EXAMPLE BELOW"
320 PRINT "    SHOWS THE PROPER INPUT FORMAT."
330 PRINT "J    EXAMPLE"
340 PRINT "    -------"
350 PRINT "    DD-MON-YR HR:MIJ"
360 PRINT "    05-JUL-80 13:45"
370 PRINT "JENTER DATE AND TIME : ";
380 INPUT X$
400 CALL "SETTIM",X$
410 CALL "MOUNT",0,A$
420 CALL "MOUNT",1,B$
430 UNIT 1
440 OLD "MENU"
450 REM                     MENU IS LOADED
1000 PRI "JJ  * THE PROGRAM IS ENDED. TO GET BACK ON THE SYSTEM FOLLOW"
1100 PRINT "    THE START-UP PROCEDURE SHOWN IN THE MANUEL."
```

The menu may be loaded without the autoload program. The clock must be set, the disks inserted and mounted, then:

```
CALL "UNIT", 1
OLD "MENU"
RUN
```

TITLE	ABSTRACT NO:
PRODUCTION CAPACITY FORECASTING MODEL	062-6715-01 Program 3

5. Press the Autoload key; the following prompts will appear checking for the correct start-up procedures.

Figure III.2
Start-Up Prompts

```
RUNNING THIS PROGRAM REQUIRES FAMILIARITY WITH THE SYSTEM.
PLEASE CONSULT THE OPERATORS MANUAL IF YOU ARE NEW TO THIS
PROGRAM.

  BEFORE STARTING PROGRAM, BE SURE DISKS ARE INSERTED
  IN THEIR PROPER DRIVES.
        DRIVE 0 - FILE DISK
        DRIVE 1 - PROGRAM DISK

HAVE ALL START-UP PROCEDURES BEEN FOLLOWED ? (Y/N)Y •
```

* In the following examples, all information entered by the user will be underlined with a solid line, and all carriage return cues will be marked by a large dot.

```
* THE SYSTEM CLOCK HAS NOT BEEN SET -- THE EXAMPLE BELOW
  SHOWS THE PROPER INPUT FORMAT.

  EXAMPLE
  -------
  DD-MON-YR HR:MI

  05-JUL-80 13:45

ENTER DATE AND TIME : 29-SEP-80 08:48 •
```

In this example note the message saying the system clock has not been set. The system clock is internal to the file manager and must be set before the disk drives will operate. The program provides a check to insure that the clock is set, and prompts the user if it is not. The example shows the case where the clock has not been set. Enter the information as shown in the example.

Once the system has been checked and is ready, a listing of the main directory is displayed on the screen.

TITLE	ABSTRACT NO:
PRODUCTION CAPACITY FORECASTING MODEL	062-6715-01 Program 3

B. Main Directory

The directory provides the user with a list of functions. Entering the appropriate number will give the user access to any one of these functions.

MAIN DIRECTORY

NUMBER	FUNCTION
1	CREATE A NEW FILE
2	LIST A FILE
3	UPDATE A FILE
4	KILL A FILE
5	RUN THE MODEL
6	CLOSE FILES/END PROGRAM

SELECT NUMBER : 1 •

Figure III.3

Main Directory

Each of the functions listed above is explained and illustrated in the sections immediately following. The selection of function number 6 simply allows the user to exit the system.

TITLE	ABSTRACT NO:
PRODUCTION CAPACITY FORECASTING MODEL	062-6715-01 Program 3

C. Creating New Files

The new file directory lets the user choose the type of file to be created. New files should be created when you are entering data for the first time. The creation of any file requires a unique file name and a defined production area; that is, a production area with a given number of products and processes. (See Section II concerning data requirements for running the model.) In order for the model to run, the first four file types must be created. The process name/notes and product name files are optional and not required for program execution.

NEW FILE DIRECTORY

FILE TYPE NUMBER	FILE TYPE
1	PROCESS TIME/YIELD RATES
2	PROD. TIME AVAIL./UTILIZAT
3	DEMAND EQUATIONS
4	GEN. CAPACITY EQUATION
5	PROCESS NAME/NOTES
6	PRODUCT NAMES

ENTER 7 TO RETURN TO MAIN DIRECTORY

Figure III.4
New File Directory

SELECT FILE TYPE NUMBER : 1.

YOU WILL BE ASKED TO SELECT A UNIQUE FILE NAME FOR YOUR DATA FILE -- DO YOU WANT TO SEE A LISTING OF FILES ALREADY ON DISK ? (Y/N)Y.

TITLE	ABSTRACT NO:
PRODUCTION CAPACITY FORECASTING MODEL	062-6715-01 Program 3

To create a certain file type, enter the appropriate file type number. To insure the selection of a unique file name, you may view the list of data files currently on disc. The following example shows a listing of data files.

Figure III.5
List of Files

```
LIST OF FILES
-------------
SCRATCHLIB/D6
SCRATCHLIB/D5
SCRATCHLIB/D4/CAP
SCRATCHLIB/D3/DEM
SCRATCHLIB/D2
SCRATCHLIB/CAPTIME
SCRATCHLIB/CAPOUT

   SELECTED FILE NAME : D1.
```

You should notice that the files listed follow two forms:

1. Scratchlib/_____

OR

2. Scratchlib/_____/Dem/Or/Cap

In the first form, the file name is the name immediately following the first slash. In the second form, the file name is the name falling between the first and second slashes. The other parts of the name shown in the listing are assigned by the computer. The user must recognize that a file is always identified by the name assigned by the user, and not by the expanded name shown in the list out.

TITLE	ABSTRACT NO:
PRODUCTION CAPACITY FORECASTING MODEL	061-6715-01 Program 3

Example:

The user assigns a file the name "GOGO." The output, when listing the file names, will appear as follows:

Scratchlib/GOGO

If the file assigned the name "GOGO" had been a demand or capacity equation file, the listing would appear as:

Scratchlib/GOGO/Dem - For a demand equation file

Scratchlib/GOGO/Cap - For a capacity equation file

* The file is always addressed by the name assigned during file creation. In this case the name is "GOGO."

Once an appropriate file name is chosen, the name may be entered, at the terminal. There are four restrictions on file names:

1. The name must begin with an alphabetic character.
2. The name must be nine or fewer characters in length.
3. The symbol "/" may not be used in the name.
4. No spaces may appear in the name.

Following the selection of a file name, the computer will display the selected file name and ask for inputs concerning the number of products and processes in a production area. It is essential that these numbers be the same for all data files relating to a given production area. If they are not, an error check will stop the model from running.

Figure III.6 Entering Area Information

```
PLEASE MAKE NOTE OF YOUR FILE NAME AND THE CORRESPONDING
FILE TYPE --- YOU WILL NEED TO KNOW THESE TO RUN THE
MODEL.
FILE NAME IS  D1
INFORMATION CONCERNING THE PRODUCTION AREA MUST NOW BE
ENTERED.  THIS INFORMATION MUST BE CONSISTENT FOR EACH
DATA FILE CONCERNING A PRODUCTION AREA.


THE NUMBER OF PROCESSES = 3.

THE NUMBER OF PRODUCTS = 2.


  * ARE YOUR INPUTS FOR NUMBER OF PROCESSES AND NUMBER
    OF PRODUCTS CORRECT ? (Y/N) Y.
```

TITLE	ABSTRACT NO:
PRODUCTION CAPACITY FORECASTING MODEL	062-6715-01 Program 3

Once this initial information has been entered to the computer, the actual data entry may begin. The following sections show the entry of data given in Section II.

* The user should note that corrections can only be made following completed data entry; you cannot go back and change your mistakes, you must finish your entry before you may enter the correction mode.

1. Processing Time/Yield Rate File (Required)

   This file accommodates the processing times and net yield rates for each process/product combination. For an explanation of these values see Sections III.B.3 and 4.

Figure III.7 Process Time/Yield Rate File

PROCESS TIME/YIELD RATE FILE

INPUT INFORMATION ON YIELD AND PRODUCTION RATES.
ENTRIES MUST BE MADE FOLLOWING EACH PROMPT. MISTAKES MAY BE CORRECTED USING THE UPDATE ROUTINE -- COMPLETE DATA ENTRY MUST BE MADE BEFORE GOING TO THE UPDATE ROUTINE.

PROCESS NUMBER	PRODUCT NUMBER	NET YIELD RATE	PROCESS TI PER UNIT
1	1	.72	1.6
1	2	.54	1.6
2	1	0	2.75
2	2	1	0
3	1	1	0
3	2	.6	15

TITLE

PRODUCTION CAPACITY FORECASTING MODEL

ABSTRACT NO:

062-6715-01
Program 3

A mistake was made when entering the data. The net yield rate for process two and product one is incorrect. Once data entry is completed, the user may go to the correction routine. The prompts showing this are given in Section III.C.7. The correction of this mistake is shown is Section III.E.1.

2. Production Time Available/Utilization File (Required)

This file holds the production time available and utilization factors for each process. For an explanation of these values see Section II.B.5.

Figure III.8 Processing Time Machine Utilization File

```
PRODUCTION TIME AVAIL./UTILIZATION FILE
---------------------------------------

ENTRIES MUST BE MADE FOLLOWING EACH PROMPT.  MISTAKES MAY
BE CORRECTED USING THE UPDATE ROUTINE -- COMPLETE DATA
ENTRY MUST BE MADE BEFORE GOING TO THE UPDATE ROUTINE.

PROCESS          TIME              MACHINE
NUMBER           AVAILABLE         UTILIZATION
-------          ---------         -----------
  1              2520.             .9
  2              1260.             .9
  3              1260.             .9
```

TITLE	ABSTRACT NO:
PRODUCTION CAPACITY FORECASTING MODEL	062-6715-01 Program 3

3. Demand Equations Program/File (Required)

This file holds the demand equations developed for the model. For an explanation of these equations see Section II.C.1.

* You will notice that both the demand equation and generalized capacity equation are saved in program/files. What this means is that the equations are saved in the computer as programs with corresponding line numbers. This is necessary because of the use of variables in the equations. This program is then referred to as a program/file because it is saving the equations used for evaluating a production area. Because the equations are saved as programs, their entry into the computer follow a different format than the regular data files.

```
DEMAND EQUATIONS PROGRAM/FILE
-----------------------------

THE PROGRAM WILL NOW BE STOPPED AND YOU WILL NEED TO INPUT
THE LINE NUMBERS AND EQUATIONS.  YOUR INPUT SHOULD FOLLOW
AS IN THE INSTRUCTION MANUAL  THE STATEMENTS SHOULD BE ENT-
ERED ANYWHERE BETWEEN LINE NUMBERS 7000 AND 10000.

  * ONCE ALL DATA IS ENTERED, PRESS USER DEFINE KEY 8.

BEGIN YOUR INPUT.


STOP IN LINE 4100 PRIOR TO LINE 4110

7000 D(1)=25+14*T(I).
7100 D(2)=16+5.4*T(I).
```

Figure III.9
Demand Equations File

You will notice that all the equations must be entered at line numbers between 7,000 and 10,000. The maximum length of any demand equation is 60 characters. Again I refer you to Section II.C.1.

TITLE	ABSTRACT NO:
PRODUCTION CAPACITY FORECASTING MODEL	062-6715-01 Program 3

4. Generalized Capacity Equation Program/File (Required)

This file holds the generalized capacity equation developed for the model. For an explanation of this equation see Section II.C.2. For an explanation of a program/file, see Section III.C.3.

Figure III.10 Capacity Equation File

```
GENERALIZED CAPACITY EQUATION PROGRAM/FILE
------------------------------------------

THE PROGRAM WILL NOW BE STOPPED AND YOU WILL NEED TO INPUT
THE LINE NUMBERS AND EQUATIONS.  YOUR INPUT SHOULD FOLLOW
AS IN THE INSTRUCTION MANUAL.  THE STATEMENTS SHOULD BE ENT-
ERED ANYWHERE BETWEEN LINE NUMBERS 7000 AND 10000.

  * ONCE ALL DATA IS ENTERED, PRESS USER DEFINE KEY 8.

BEGIN YOUR INPUT.


STOP IN LINE 4100 PRIOR TO LINE 4110

7000 X1=T1(I)*U(I)
7100 Z2=C(I,2)*16
7500 X7=C(I,1)*14
8000 T(I)=(X1-C(I,1)*25-Z2)/(X7+C(I,2)*5.4)
```

As discussed in the previous section, all equations must be entered at line numbers between 7,000 and 10,000. The maximum length of any demand equation is 60 characters. For a detailed look at the equation, I again refer you to Section III.C.3.

TITLE	ABSTRACT NO:
PRODUCTION CAPACITY FORECASTING MODEL	062-6715-01 Program 3

5. Process Name/Notes File (Optional)

This file stores the names and notes assigned to a process number. The process notes are provided to allow the user a place to enter any special charateristics concerning that process. The maximum length for process names is 15 characters, and for notes it is 20 characters. If the user chooses not to enter a name for a certain process, then the process number should be entered in its place. You are not required to enter anything in the process notes column.

Figure III.11 Process Name/Notes File

PROCESS NAME/NOTES FILE

THIS PROGRAM RECORDS THE NAMES AND CHARACTERISTICS ASSOCIATED WITH A GIVEN PROCESS. THE USER SHOULD KNOW THE INFORMATION CORRESPONDING TO A GIVEN PROCESS NUMBER. AN ENTRY MUST BE MADE FOR EACH PROCESS NUMBER.

PROCESS NAME -- MAX. LENGTH = 15 CHARACTERS
PROCESS NOTES -- MAX. LENGTH = 20 CHARACTERS

LENGTH OF INPUT MUST NOT EXCEED THE LENGTH OF HEADING LINES

PROCESS NUM	PROCESS NAME	PROCESS NOTES
1	DRILL •	ORDERED •
2	VAPOR BLAST •	EXISTING •
3	DAG WASH •	NEW •

This data file is optional and not required for program execution; however, it does provide for a better looking output.

TITLE	ABSTRACT NO:
PRODUCTION CAPACITY FORECASTING MODEL	062-6715-01 Program 3

6. Product Name File (Optional)

This file stores the name assigned to a product number. The maximum length for the product name is 12 characters. If the user chooses not to enter a name for a certain product, then the product number should be entered in its place.

Figure III.12 Product Name File

PRODUCT NAME FILE

THIS PROGRAM ASSOCIATES PRODUCT NAMES TO PRODUCT NUMBERS
THE USER SHOULD KNOW WHICH NAMES CORRESPOND TO WHICH NUMBERS.
AN ENTRY MUST BE MADE FOR EACH PRODUCT NUMBER.

PRODUCT NAME -- MAX. LENGTH = 12 CHARACTERS

INPUT SHOULD NOT EXCEED THE LENGTH OF HEADING LINE.

PRODUCT NUM.	PRODUCT NAME
1	19" CRT •
2	25" CRT •

Again, this data file is optional and not required for program execution; however, it does provide for a better looking output.

TITLE	ABSTRACT NO:
PRODUCTION CAPACITY FORECASTING MODEL	062-6715-01 Program 3

7. Once data entry is ended, the user may direct the path of program execution. The information entered has been stored on the disc under the assigned file name. If you want to make corrections, you may go directly to the correction mode. The sample below shows the prompt displayed following data entry.

Figure III.13 Prompt Following Completion of Data Entry

```
END OF PROGRAM -- DATA ENTERED HAS BEEN STORED ON DISK.
YOU CAN NOW SELECT FROM ANY OF THE FUNCTIONS LISTED BELOW.

   1. DISPLAY MAIN DIRECTORY.
   2. GO DIRECTLY TO THE UPDATE ROUTINE (FOR CORRECTIONS).
   3. LIST OUT FILE.
   4. CREATE A NEW FILE.

SELECTED NUMBER =
```

D. Listing Data Files

Any of the data files on record may be listed for viewing. To get a listing of a file, simply enter the file name. The files may be seen on the CRT, or listed at the line printer. If you want a listing at the line printer, be sure it is turned on. You may use either 11" or 14" paper for the listing.

Figure III.14

Prompt for Listing Data Files

```
                    LISTING DATA FILES

THIS PROGRAM ALLOWS YOU TO LIST A FILE AT THE LINE PRINTER
OR ON THE C.R.T.   IF YOU WANT A LISTING ON THE LINE PRINTER
BE SURE IT IS SWITCHED "ON".


YOU WILL NOW HAVE TO INPUT THE FILE NAME OF THE FILE YOU WANT
TO LIST -- DO YOU FIRST WANT TO SEE A LISTING OF THE FILES
CURRENTLY ON DISK ? (Y/N) N.

SELECTED FILE NAME: D1.


------- LOCATION OF PRINT OUT --------

   1. DISPLAY ON THE SCREEN.
   2. PRINT AT THE LINE PRINTER.

SELECTED NUMBER 1.
```

TITLE	ABSTRACT NO:
PRODUCTION CAPACITY FORECASTING MODEL	062-6715-01 Program 3

Once you have selected the file to be printed and the location of output, the file will be listed giving its file type and associated data. The example below lists out the process time/yield rate file created in Section III.C.1.

Figure III.7 Process Time/Yeild Rate File

PROCESS TIME /YIELD RATE FILE

PROCESS NUMBER	PRODUCT NUMBER	NET YIELD RATE	PROCESSING TIME
1	1	0.72	1.6
1	2	0.54	1.6
2	1	0	2.75
2	2	1	0
3	1	1	0
3	2	0.6	15

--- LISTING IS FINISHED, YOU MAY DIRECT PROGRAM CONTROL -

1. RETURN TO MAIN DIRECTORY.
2. LIST MORE FILES.

SELECTED NUMBER : 1

At the end of the listing, the user chooses the direction of program control.

TITLE	ABSTRACT NO:
PRODUCTION CAPACITY FORECASTING MODEL	062-6715-01 Program 3

E. Updating Data Files

The updating routine allows the user to make corrections on existing data files. There are three possible types of corrections which may be performed:

- o Change or correct data values in a data file with no change in the number of products or processes.
- o Delete processes and/or products and their corresponding data values from a data file.
- o Add processes and/or products and their corresponding data values to a data field.

The first step is to enter the file name for the file you want to work on. In this example, we will work on the process time/yield rate file created in Section III.C.1.

Figure III.15 Prompt for Data File Correction Selection

```
==========================================================

                CORRECTION OF DATA FILES

==========================================================


THIS PROGRAM ALLOWS FOR EXPANDING,UPDATING, OR DELETING
RECORDS IN A FILE.THE FILE MUST EXIST FOR THIS ROUTINE TO RUN.

DO YOU WANT TO SEE A LISTING OF THE FILES ON DISK ? (Y/N) N

 SELECTED FILE NAME : D1.
```

TITLE	ABSTRACT NO:
PRODUCTION CAPACITY FORECASTING MODEL	062-6715-01 Program 3

Once the file name is entered, the file type is printed out, along with the number of products and processes on record. (These values are not printed out when looking at capacity or demand equation files.) The user then selects either the update routine or the expand/delete routine.

Figure III.16 Yield Rate File Correction

```
PROCESS TIME AND YIELD RATE FILE
--------------------------------

THE NUMBER OF PROCESSES ON FILE = 3

THE NUMBER OF PRODUCTS ON FILE = 2

YOU MAY SELECT FROM TWO CHOICES IN THE CORRECTION MODE --

  1. UPDATE THE FILE - CHANGE ONLY INFORMATION PERTAINING TO
     PRODUCTS/PROCESSES CURRENTLY ON RECORD.

  2. EXPAND/DELETE - ADD OR SUBTRACT TO NUMBER OF PROCESSES
     OR PRODUCTS ON DISK.

ENTER CHOICE: 1.
```

1. Update

   In this example, we have chosen to make corrections within the process time/yield rate file named "D1." To access the desired values within the file, we enter the appropriate process and product number. The computer will respond by printing out the values as they now

TITLE	ABSTRACT NO:
PRODUCTION CAPACITY FORECASTING MODEL	062-6715-01 Program 3

exist, and then print out the process and product numbers again. The user may now enter the correct values. If you do not want to change a value, you simply hit the return key. When you have finished your corrections, you may exit the routine by pressing user defineable key 3.

*NOTE: You may only exit the routine when a star is showing in the process number column.

Figure III.16 Yield Rate File Correction

PROCESS TIME AND YIELD RATE FILE UPDATE

THE COMPUTER WILL RESPOND WITH COLUMN TITLES AND A "*" FOLLOWED BY A BLINKING QUESTION MARK. THE USER SHOULD TYPE IN THE VALUES FOR THE RECORD TO BE CORRECTED AND PRESS THE "RETURN" KEY. THE RECORD WILL BE DISPLAYED AS IT NOW EXISTS ON THE DISK. CORRECTIONS CAN NOW BE MADE BY INPUTTING THE CORRECT VALUES AT THE APPROPRIATE LOCATION -- IF YOU DO NOT WANT TO CHANGE A VALUE - PRESS THE RETURN KEY.

* * * TO EXIT THE ROUTINE, PRESS USER DFN KEY 3. * * *

PROCESS #	PRODUCT #	NET YLD. RATE	PROD. RATE
* 1	2	0.54	1.6
1	2	0.54	1.6
* 2	1	0	2.75
2	1	.9	2.75
*			

The update shown above was to correct the mistake made on entry of the data to the file. The original file is shown in Section III.C.1. The corrected file is shown below.

TITLE	ABSTRACT NO:
PRODUCTION CAPACITY FORECASTING MODEL	062-6715-01 Program 3

Figure III.16 Yield Rate File Correction

PROCESS TIME /YIELD RATE FILE

PROCESS NUMBER	PRODUCT NUMBER	NET YIELD RATE	PROCESSIN(TIME
1	1	0.72	1.6
1	2	0.54	1.6
2	1	0.9	2.75
2	2	1	0
3	1	1	0
3	2	0.6	15

--- LISTING IS FINISHED. YOU MAY DIRECT PROGRAM CONTROL -

1. RETURN TO MAIN DIRECTORY.
2. LIST MORE FILES.

SELECTED NUMBER

The update routine for other file types is similar, except for the demand and capacity equation files. These files must be updated using specific line numbers, following the form shown in Section III.C.3 and III.C.4.

2. Expand/Delete

Here the selection of the expand/delete routine was made. The deletion of any products and/or processes is completed first, followed by the expansion or addition of products and/or processes. In the following example, we will first add two processes and one product, with their corresponding values.

TITLE	ABSTRACT NO:
PRODUCTION CAPACITY FORECASTING MODEL	061-6715-01 Program 3

a. Expansion

The expansion of a file takes place when new products and/or processes are accounted for in a production area. The first prompt asks for the number of processes and number of products you want to delete. Here, we do not want to make any deletions so we enter "zero" for each prompt.

Figure III.17 Yield Rate File Expand/Delete

```
PROCESS TIME AND YIELD RATE FILE EXPAND/DELETE
----------------------------------------------

YOU CAN ADD OR SUBTRACT PROCESSES OR PRODUCTS FROM THIS
FILE. NOTE-- YOU SHOULD SEE THE MANUAL FOR HOW NUMBERS ARE
ASSIGNED.

NUMBER OF PROCESSES YOU WANT TO SUBTRACT = 0.

NUMBER OF PRODUCTS YOU WANT TO SUBTRACT = 0.
```

We are next asked for the number of processes and products we want to add. The expansion of a file always begins where the last file ended. That is, the new processes and/or products are assigned sequential numbers beginning where the initial numbers left off. For our example we had a file with two products and three processes; the new processes will be assigned the numbers "4" and "5" and the new product will be called product number "3." You must enter values for all of the process and product number combinations given. The example below illustrates the expansion of a file:

TITLE	ABSTRACT NO:
PRODUCTION CAPACITY FORECASTING MODEL	062-6715-01 Program 3

Figure III.17 Yield Rate File Expand/Delete

THE EXPANSION OF A FILE BEGINS WHERE THE FILE LEFT OFF. YOU SHOULD BE SURE TO NOTE THE NUMBER ASSIGNED TO EACH PROCESS AND PRODUCT. BEFORE EXPANDING A FILE YOU MUST HAVE THE NECESSARY DATA FOR INPUT. (SEE THE INSTRUCTION MANUAL)

THE NUMBER OF PROCESSES YOU WISH TO ADD = 2 •

THE NUMBER OF PRODUCTS YOU WISH TO ADD = 1 •

PROCESS #	PRODUCT #	NET YLD RATE	PROD. RATE
1	3	.54 •	1.6 •
2	3	0 •	0 •
3	3	0 •	0 •
4	1	.9 •	2.75 •
4	2	0 •	0 •
4	3	1 •	0 •
5	1	1 •	0 •
5	2	0 •	0 •
5	3	.6 •	15 •

Once data entry is completed, the file is saved and the user may direct program control. These prompts are shown in Section III.E.3. A listing of our updated file is shown in Section III.E.2.b.

TITLE	ABSTRACT NO:
PRODUCTION CAPACITY FORECASTING MODEL	061-6715-01 Program 3

b. Deletion

Deletion involves taking out processes or products which are no longer used in a production area. The data corresponding to these processes and/or products is erased from the file. The file is then compressed so that all product and process numbers remain in sequential order. The file listed below shows our expanded version of file "D1" with five processes and three products.

Figure III.17 Yield Rate File Expand/Delete

PROCESS TIME /YIELD RATE FILE

PROCESS NUMBER	PRODUCT NUMBER	NET YIELD RATE	PROCESSIN TIME
1	1	0.72	1.6
1	2	0.54	1.6
1	3	0.54	1.6
2	1	0.9	2.75
2	2	1	0
2	3	0	0
3	1	1	0
3	2	0.6	15
3	3	0	0
4	1	0.9	2.75
4	2	0	0
4	3	1	0
5	1	1	0
5	2	0	0
5	3	0.6	15

--- LISTING IS FINISHED, YOU MAY DIRECT PROGRAM CONTROL -

1. RETURN TO MAIN DIRECTORY.
2. LIST MORE FILES.

SELECTED NUMBER :

TITLE	ABSTRACT NO:
PRODUCTION CAPACITY FORECASTING MODEL	062-6715-01 Program 3

In response to the prompts asking for the number of products and processes you want deleted, enter the appropriate information. Assume you want to delete two processes and one product. After entering the number of processes you want deleted, you must then enter the process numbers corresponding to the specific processes to be deleted. The same thing applies to deleting products. The example below illustrates:

Figure III.17 Yield Rate File Expand/Delete

```
PROCESS TIME AND YIELD RATE FILE EXPAND/DELETE
----------------------------------------------

YOU CAN ADD OR SUBTRACT PROCESSES OR PRODUCTS FROM THIS
FILE  NOTE-- YOU SHOULD SEE THE MANUAL FOR HOW NUMBERS ARE
ASSIGNED.

NUMBER OF PROCESSES YOU WANT TO SUBTRACT = 2 .


FOLLOWING EACH PROMPT, TYPE IN THE PROCESS OR PRODUCT
NUMBER YOU WANT DELETED. YOU MUST ENTER THE PRODUCT OR PRO-
CESS NUMBERS IN ORDER - BEGINNING WITH THE LOWEST NUMBER AND
WORKING TO THE HIGHEST NUMBER.

PROCESS/PROD. #
---------------
2.
3.

* CHECK YOUR INPUTS -- ARE THE CORRECT NUMBERS ENTERED
  AND ARE THEY IN THE CORRECT ORDER ? (Y/N) Y .

NUMBER OF PRODUCTS YOU WANT TO SUBTRACT = 1 .


FOLLOWING EACH PROMPT, TYPE IN THE PROCESS OR PRODUCT
NUMBER YOU WANT DELETED. YOU MUST ENTER THE PRODUCT OR PRO-
CESS NUMBERS IN ORDER - BEGINNING WITH THE LOWEST NUMBER AND
WORKING TO THE HIGHEST NUMBER.

PROCESS/PROD. #
---------------
2 .

* CHECK YOUR INPUTS -- ARE THE CORRECT NUMBERS ENTERED
  AND ARE THEY IN THE CORRECT ORDER ? (Y/N) Y .
```

TITLE	ABSTRACT NO:
PRODUCTION CAPACITY FORECASTING MODEL	062-6715-01 Program 3

Once the values are entered, the computer deletes all information dealing with the specific process and/or product numbers, compresses the numbers into sequential order, and saves the revised file on the disk. The revised file appears as shown below:

Figure III.17 Yield Rate File Expand/Delete

PROCESS TIME /YIELD RATE FILE

PROCESS NUMBER	PRODUCT NUMBER	NET YIELD RATE	PROCESSIN(TIME
1	1	Ø.72	1.6
1	2	Ø.54	1.6
2	1	Ø.9	2.75
2	2	1	Ø
3	1	1	Ø
3	2	Ø.6	15

--- LISTING IS FINISHED. YOU MAY DIRECT PROGRAM CONTROL -

1. RETURN TO MAIN DIRECTORY.
2. LIST MORE FILES.

SELECTED NUMBER :

* Note the new process and product numbers. After deleting processes two and three, the computer renamed the former process five as process three. The same type of compression follows with the product numbers. The key here is that the user understands this change in product and process numbers.

TITLE	ABSTRACT NO:
PRODUCTION CAPACITY FORECASTING MODEL	062-6715-01 Program 3

The computer next asks for the number of processes and products to be added. In this example, there are no products or processes to be added, so "zero" is entered following each prompt.

Figure III.17 Yield Rate File Expand/Delete

```
THE EXPANSION OF A FILE BEGINS WHERE THE FILE LEFT OFF. YOU
SHOULD BE SURE TO NOTE THE NUMBER ASSIGNED TO EACH PROCESS
AND PRODUCT. BEFORE EXPANDING A FILE YOU MUST HAVE THE
NECESSARY DATA FOR INPUT. (SEE THE INSTRUCTION MANU L)


THE NUMBER CF PROCESSES YOU WISH TO ADD = 0 .

THE NUMBER OF PRODUCTS YOU WISH TO ADD =  0 .
```

3. Following the completion of either the update routine or the expand/delete routine, the computer will prompt the user concerning program control. You should enter the appropriate number:

```
FILE UPDATE/EXPANSION IS COMPLETED --- RETURN CONTROL TO :

   1. MAIN DIRECTORY
   2. REMAIN IN CORRECTION MODE

SELECTED NUMBER =
```

TITLE	ABSTRACT NO:
PRODUCTION CAPACITY FORECASTING MODEL	062-6715-01 Program 3

F. Killing Data Files

This routine allows the user to completely erase a data file. This is particularly useful when the "FILES" disk has a lot of obsolete data files. These files may be erased, leaving room for more files.

* Once a data file is killed, the information that was in the file is unrecoverable. Be sure to enter the correct file name when you are killing a data file.

Figure III.18 Killing A File

```
==========================================================

                    KILLING A FILE

==========================================================


THIS PROGRAM SHOULD ONLY BE USED WHEN A FILE IS NO LONGER
NEEDED - DO YOU WANT TO DESTROY A FILE ? (Y OR N) Y .


DO YOU WANT TO SEE A LISTING OF THE FILES ON DISK ? (Y/N) N.


YOU WILL NOW BE ASKED FOR THE NAME OF THE FILE YOU WANT
TO KILL.  ALL INFORMATION WILL BE LOST, SO BE SURE TO USE
THE CORRECT FILE NAME.

FILE NAME : TRIAL .


---- THE FILE IS DESTROYED. ARE THERE MORE FILES YOU WANT
TO ERASE ? (Y/N) N .
```

TITLE	ABSTRACT NO:
PRODUCTION CAPACITY FORECASTING MODEL	062-6715-01 Program 3

G. Running the Model

1. The Run

The model is ready to run when the four required data files are created and on disk. The product name file and the process name/notes file are optional; you may have either one or both created and on disk. The following prompts are necessary to determine which files are to be used for the run.

Figure III.19 Running The Model

```
==========================================================

                RUN / SORT / LIST THE CAPACITY MODEL

==========================================================

THIS PROGRAM WILL TAKE THE DATA FILES YOU CREATED AND EVALUATE
THE INFORMATION.  THE FOLLOWING FOUR FILE TYPES ARE REQUIRED
TO RUN THE MODEL.

   1  PROCESSING TIME/NET YIELD RATE FILE.
   2. PRODUCTION TIME AVAIL./UTILIZATION FILE.
   3. DEMAND EQUATION PROGRAM/FILE.
   4. GENERALIZED CAPACITY EQUATION PROGRAM/FILE.

DO YOU HAVE THE NECESSARY FILES CREATED ? (Y OR N) Y.

DO YOU WANT TO SEE A LISTING OF FILES ON DISK ? (Y OR N) N.

DO YOU HAVE THE OPTIONAL PROCESS NAME/NOTES FILE ON
DISK ? (Y/N) Y.

DO YOU HAVE THE OPTIONAL PRODUCT NAME FILE ON DISK ? (Y/N)Y.
```

TITLE	ABSTRACT NO:
PRODUCTION CAPACITY FORECASTING MODEL	062-6715-01 Program 3

When asked about the process name/notes file and the product name file, the user should respond to the prompts concerning these files' existence. This tells the computer whether or not to account for these optional files during the run. After entering "yes" on the existence of the optional files, the computer then asks for the appropriate file names. This example is listed below.

Figure III.20 Entering the Data File Names

```
YOU SHOULD HAVE 6 FILE NAMES TO ENTER BELOW. PLEASE
ENTER THE FILE NAME FOLLOWING EACH PROMPT.

1. PRODUCTION AND YIELD RATE FILE.

FILE NAME ? D1 •

2. PRODUCTION TIME AVAILABLE AND UTILIZATION FILE.

FILE NAME ? D2 •

3. DEMAND EQUATION PROGRAM/FILE.

FILE NAME ? D3 •

4. GENERALIZED CAPACITY EQUATION PROG./FILE.

FILE NAME ? D4 •

5. PROCESS NAME/NOTES FILE.

FILE NAME ? D5 •

6 PRODUCT NAME FILE.

FILE NAME ? D6 •
```

TITLE	ABSTRACT NO:
PRODUCTION CAPACITY FORECASTING MODEL	062-6715-01 Program 3

Once the file names have been entered you will be asked to enter the number of processes and products involved in the production area. This is necessary for an error check which is performed before the model will run. This check involves comparing the number of processes and products actually in the files, with the values entered here by the user. If mistakes are found, the computer responds with error messages showing where the mistakes are. If no error messages appear, then the files check out and the model is ready to run.

Figure III.21 Prompt For File Error Checking

```
FOR THE PROGRAM TO RUN, THE FILES MUST BE CONSISTENT --
THEY SHOULD HAVE EQUAL NUMBER OF PROCESSES AND PRODUCTS
ON RECORD.  THE FILES WILL BE CHECKED FOR CONSISTENCY.

NUMBER OF PROCESSES = 3 .

NUMBER OF PRODUCTS = 2 .


HAVE ANY ERROR MESSAGES APPEARED ON THE SCREEN ? (Y OR N) N .
```

Once the run is complete, the user must select what form the output is to take. This involves selection of a sort routine and list routine. The various options are shown in the following section.

2. Output of Results

The various sort, list, and heading options provide flexibility in the output of the results. This helps in providing a meaningful output for the user. The first option available is the sort selection.

TITLE	ABSTRACT NO:
PRODUCTION CAPACITY FORECASTING MODEL	062-6715-01 Program 3

a. Sort Selection

The sort routine is used to order the processes in a meaningful way. The processes may be sorted in ascending order (from low to high values) looking at a given product's capacity or the time to reach capacity. You also have the option of not sorting the values and having the processes printed out in the order of their process numbers. If you choose the first sort routine, you may select which product you want to sort off of, and whether you want to go off of that product's net output at capacity or the gross starts at capacity. You may want to experiment with the different sort routines to find the one best suited for your output. The sort selection is shown below.

Figure III.22

Sort Method Selection

```
** THE RUN IS COMPLETE **

THE MODEL HAS BEEN RUN AND THE VALUES ARE READY FOR OUTPUT.
YOU WILL BE GIVEN CHOICES AS TO THE FORMAT OF OUTPUT. THESE
CHOICES INCLUDE -- METHOD OF SORTING AND LOCATION AND
STYLE OF OUTPUT.


                    SORT SELECTION
                    ==============

YOU MUST NOW SELECT THE METHOD OF SORTING THE OUPUT.
THE SORT IS USED TO ORDER THE PROCESSES IN A MEANINGFUL
WAY. THERE ARE THREE POSSIBLE WAYS TO PERFORM THE SORT:

   1. ASCENDING ORDER LOOKING AT THE PROCESS CAPACITIES
      FOR A GIVEN PRODUCT - YOU SELECT THE PRODUCT.

   2. ASCENDING ORDER KEYING OFF OF THE TIME TO REACH
      CAPACITY ON EACH PROCESS.

   3. NO SORT - LIST OUT THE VALUES IN THE ORDER OF THE
      PROCESS NUMBERS.

SORT FUNCTION = 3 .
```

TITLE	ABSTRACT NO:
PRODUCTION CAPACITY FORECASTING MODEL	062-6715-01 Program 3

b. Listing

Once your have selected a sort routine, you must decide how and where you want your output listed. You may list all of the products or one product of your choice. The output can be displayed at the screen or sent to the line printer (assuming the printer is available). These choices are shown in the example below.

Figure III.23 List Results

```
                LISTING OF OUTPUT
              ========================

YOU MUST NOW SELECT THE STYLE AND LOCATION FOR PRINTING
THE RESULTS. YOU HAVE  TWO OPTIONS FOR THE STYLE OF THE
OUTPUT --

   1. LIST ONE PRODUCT ONLY. (YOU SELECT WHICH PRODUCT)
   2. LIST ALL PRODUCTS.

SELECTED NUMBER = 2 .


YOU MAY LIST THE RESULTS AT THE LINE PRINTER OR CRT. YOU
MAY WISH TO PREVIEW YOUR RESULTS ON THE CRT. BEFORE PRINTING
AT THE LINE PRINTER.
   1. CRT.
   2. LINE PRINTER.

SELECTED NUMBER = 1 .
```

TITLE	ABSTRACT NO:
PRODUCTION CAPACITY FORECASTING MODEL	062-6715-01 Program 3

c. Title of Report

The title is displayed at the top of the output and provides information concerning the production area and the person responsible for the computer run. The example below shows the input of header information.

Figure III.24 Prompt For Title Selection

```
                TITLE OF REPORT
                ===============

YOU WILL NOW BE ASKED FOR INFORMATION TO BE USED ON THE
HEADING OF THE REPORT. YOU ARE LIMITED TO A MAXIMUM OF 50
CHARACTERS WHEN ENTERING EACH PART OF THE TITLE.

ENTER THE HEADING OF THE REPORT. (MAX. OF 50 CHARACTERS)
LARGE TUBE PRODUCTION AREA

ENTER THE PRODUCTION AREA BEING CONSIDERED.
LARGE SCREEN CRT.

ENTER YOUR NAME.
MARK LYMAN •

ENTER THE DATE.
29-SEPT.-80 •
```

TITLE	ABSTRACT NO:
PRODUCTION CAPACITY FORECASTING MODEL	062-6715-01 Program 3

d. Results

The example below shows the results of running the capacity forecasting model. The output follows from the selections made in the previous examples. The net output shows the number of good units which can be turned out at capacity. The gross starts show the number of units which must be started at the process in order to turn out the maximum net output. The time to capacity indicates the amount of time until the capacity at each process is reached. In our example, the time is measured in years, but other units may be required. (See Section II.C.1.a for information concerning units.)

Figure III.25 Sample Output

PAGE 1 OF 1

TITLE . LARGE TUBE PRODUCTION AREA
AREA . LARGE SCREEN CRT
NAME . MARK LYMAN
DATE . 20-SEPT.-82

MACHINE/PROCESS NAME	PRODUCT 19" CRT NET OUTPUT	PRODUCT 19" CRT GROSS STARTS	PRODUCT 25" CRT NET OUTPUT	PRODUCT 25" CRT GROSS STARTS	TIME TO CAPACITY (YRS.)	NOTES ON PROCESS
DRILL	668	928	264	406	45.95	ORDERED
VAPOR BLAST	371	412	0	0	24.72	EXISTING
DAC WASH	0	0	45	75	5.43	NEW

THE VALUES HAVE BEEN SORTED AND LISTED. SELECT FROM ONE OF THE TWO OPTIONS GIVEN BELOW.
1. PERFORM NEW SORT AND LIST.
2. RETURN TO MAIN DIRECTORY.

SELECTED NUMBER: 1 •

Once the listing is completed, you may elect to perform a new listing or return to the main directory. The program is complete and the output can be analyzed.

TITLE	ABSTRACT NO:
PRODUCTION CAPACITY FORECASTING MODEL	062-6715-01 Program 3

IV. PROGRAM DOCUMENTATION

The overall program hierarchy for the capacity model is shown below. Descriptions of the various data files are provided in the following sections.

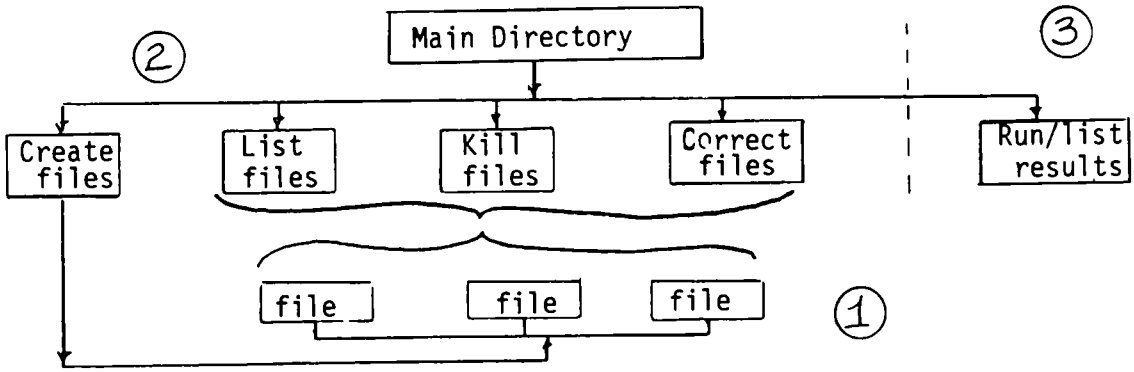
A. Program Structure

The hierarchy shown below can be related back to the three program functions given in the abstract:

1. Data storage
2. Data file management
3. Running the model

The location of each of these functions is indicated on the hierarchy.

Figure IV.1 Program Hierarchy Chart



B. Data File Structure

There are eight possible data files associated with any production area. These files may be divided into three categories:

- o Data Files required to run the program.
- o Optional data files, not essential to program execution.
- o Output data files where results from the program run are saved.

TITLE	ABSTRACT NO:
PRODUCTION CAPACITY FORECASTING MODEL	062-6715-01 Program 3

1. Required Data Files

The following data files are necessary for program execution. Each file type and its characteristics are listed below:

a. Process time/net yield rate file

- Numeric
- Binary
- Random Access
- 30 bytes/record
- First record holds three numeric values--number of products, number of processes, file type number 1.
- All succeeding records store two separate numeric values per record--process time and net yield rate
- Individual records accessed as in example given below:

Example:

Figure IV.2

Explanation of Record Numbering

Process # \ Product #	1	2	3
1	2	3	4
2	5	6	7
3	8	9	10

← Record numbers

For any given process number and product number, the following conversion will give the record number:

record # = (process # - 1)*(tot. # of products) + (product #) + 1

b. Production time available/machine utilization file

- Numeric
- Binary
- Random Access
- 30 bytes/record
- First record holds three numeric values--number of products, number of processes, file type number 2
- All succeeding records store two separate numeric values per record--the production time available and machine utilization factor.
- Individual records accessed using:

Record # = (Process # + 1)

TITLE	ABSTRACT NO:
PRODUCTION CAPACITY FORECASTING MODEL	062-6715-01 Program 3

c. Demand equations program/file

This file is a list of demand equations stored in a program format.

- Each equation must be in the proper form. (See Section II on data requirements.)
- Each equation must be assigned a corresponding line number (when creating a data file the line numbers must be between 7,000 and 10,000.
- The maximum length of one line is 72 characters.

d. Generalized capacity equations

This file is a list of the generalized capacity equations stored in program format.

- Each section of the equation must be in the proper form. (See Section II on data requirements.)
- Each section of the equation must have a corresponding line number (when creating a data file the line numbers must be between 7,000 and 10,000).
- The maximum length of one line is 72 characters.

2. Optional Data Files

The following files are not essential to running the program, but they do provide a better looking output.

a. Process name/notes file

- Alphanumeric
- Binary
- Random Access
- 45 bytes/record
- First record holds three numeric values--number of products, number of processes, file type number 3
- All succeeding records contain two separate alphanumeric strings per record--the process name and process notes
- Individual records accessed using:

  Record # = (Process #) + 1

TITLE	ABSTRACT NO:
PRODUCTION CAPACITY FORECASTING MODEL	062-6715-01 Program 3

b. Product name file

- Alphanumeric
- Binary
- Random Access
- 30 bytes/record
- Stores one alphanumeric string per record
- First record holds three numeric values--number of products, number of processes, file type number 4
- Individual records accessed using:

  Record # = (Product #) + 1

3. Output Data Files

The output data files hold the results of the program execution. This includes the time to reach capacity for each process and the output at capacity for each process/product combination. The files are kept on the "PROGRAMS" disk under the file names Captime and Capout, respectively.

a. Time to reach capacity

- Numeric
- Binary
- Random
- Temporary file name: "Captime"
- 10 bytes/record
- Stores one numeric value per record--time to capacity
- Individual records accessed using:

  Record # = Process #

b. Output at capacity

- Numeric
- Binary
- Random
- Temporary file name: "Capout"
- 20 bytes/record
- First record holds two numeric values--number of products, number of processes.

TITLE	ABSTRACT NO:
PRODUCTION CAPACITY FORECASTING MODEL	062-6715-01 Program 3

- All succeeding records store two numeric values per record--the net output and gross starts at capacity

- Individual records accessed according to process/ product number combination (see example in process time/yield rate file):

Record # = (process #-1)*(tot. # of products) + (product #) + 1

C. List of Programs/Variables

The list of programs and files given below are stored on the disk labeled "Programs". The files "Capout" and "Captime" serve to hold the output for the model before it is printed out.

Program Name	Program Function
Menu	Main directory
Newfile	Creates new files
List	Lists out data files
Kill	Deletes data file
Correct	Corrects data file
Run	Run the model
Sortlist	Sort/list results of the model

File Name	File Function
Capout	Holds results for output at capacity
Captime	Holds results for time to reach capacity

TITLE	PART NUMBER
4907 Multiple Volume Directory Lister	062-6715-01 Program 4
ORIGINAL DATE April 1982 / REVISION DATE	EQUIPMENT, OPTIONS AND SOFTWARE REQUIRED (INCLUDING PERIPHERALS AND HOST SYSTEM) 4907 File Manager (dual) Optional - 4641/3 Printer
AUTHOR Frits Handgraaf Tektronix, Inc. Europe	

ABSTRACT

Files: 5 Program
1 Data
Requires pre-MARKed data tape file

Statements: 841

The program provides a database of directories of different volumes of disks on one disk. In addition to the 4907 directory output, the program offers the ability to specify parts of the directory information (items) to be output in any order. The first program REMARK statement may be included in the output (as an extra item, provided it was specified with the CREATE function). All items may be sorted in ascending or descending order with the sort-key setting. The following functions are provided:

1. List created volume directory files

   Information is stored in two files, one contains the directory of the volume, the second contains the volume label, number of files and a flag indicating whether the first REMARK statement is included. Output may be to a specified output device.

2. Create volume directory file

   Stores the volume directories on the system disk, using the internal tape as an interim step, allowing the user to specify the first REMARK statement line to be included.

3. Sort volume directory file

   Provides a butterfly sort of the volume directory file using specified key settings in either ascending or descending order. The array and sorted information may be saved on the disk.

4. Output volume directory file

   Output of the directory (sorted or not sorted) in a format defined by the user to a specified device address.

HISTORY

The program was originally developed to provide a listing of 4907 based PLOT 50 program files together with their first included REMARK statement.

From the file-names it isn't always clear which routines reside in the program file(s). This makes it more difficult to locate error causes (if occuring in program flow). Another consideration for the development of the program is the lack of the 4907 to provide a directory of files in time/file name (alphabetical) order. This capability has been included in MVDL.

THE SOFTWARE

MVDL contains 1 start-up program, 4 function program overlays and one data file.

@MVDL/START - Starts up and initializes the system

@MVDL/OVL1
@MVDL/OVL2
@MVDL/OVL3
@MVDL/OVL4
- Contain the functions (1 thru 4), selectable from the menu which is presented after start up.

@MVDL/RECORDINFO - Contains record data: number of items in the record, item names, and start and end positions of the items in the record.

PROGRAM LOADING

To start up the system insert the disk in drive Ø and:

```
CALL "MOUNT",Ø,A$
OLD "@MVDL/START"
RUN
```

PROGRAM FUNCTIONS

List created volume directory files

Of each volume from which the directory is saved by MVDL (see CREATE below), information is stored in two files: MVDL/DIR/volume-name and MVDL/INFO/volume-name.

The first one contains the directory of the volume, the second contains volume label, number of files on the

volume and a flag indicating whether the first REMARK statement encountered in program files have been included in the database (for that volume). With the list function, a list of these data base files and/or their content can be output to a specified output device address. (See examples)

Create volume directory file

This function essentially stores volume directories on the MVDL system disk: it creates MVDL/DIR/volume-name and MVDL/INFO/volume-name and writes appropriate information into it (see List above as well).

The function requires a pre-MARKed tape file, large enough to contain DIR2, "@#" data. The internal tape of the 4050 is used for this purpose. The tape file serves as an intermedium. Within this function, the user can specify whether to include the first REMARK statement line encountered in a program file or not.

The create function is the only function within the package that requires two disk drives.

Sort volume directory file

This function allows a sort of the volume directory file with key settings, in ascending or descending order.

The sorting routine used is the butterfly sort. The outcome of the sort is kept in an index table (array) indicating the record numbers and order. This array, and sort information can be saved. The user has three characters available to name a sort file; the system keeps track of sorted files by supplying a file identifier which is identical to the directory file I.D. with the extension SXXX, so: MVDL/DIR/volume-name. SXXX data stored in such a file are item order of sort (array), a flag indicating whether line numbers and/or leading spaces* in the remark statement are included while sorting, the sorting table (array).

Output volume directory file

This function will output the directory of a volume (sorted or not sorted) in a format definable by the user:

* Note: in this case leading spaces are the possible blanks between REM (ARK) and the description/text i.e.:

```
100 REM   START
```

includes three leading spaces

Items and order of items, supplying item names or not, inclusion of REMARK statements (with line numbers and/or leading spaces), maximum number of lines/page (including one empty line between records) and output device address can be set.

The output includes the original volume label as header and will give the total number of records in the directory file.

VARIABLES

Variable	Used to Store
A$	Volume Information
B$	Directory Information
C$	DEL
D$	As B$
E$	REMark Information
F$	Name
I$ J$ Z$	Scratch
M$	Months
O$	OVL$
V$	As A$
X$ Y$	Named
C1	Filename Check
D5 D6 D7 D8	Function #
D9	Output Device
F1	EOF Flag
F5 F6 F7 F8 F9	Flags
I I1 J KØ K1 T1 T2 T3 T4	Counters
L1 L2	Record #
O	OVL #
O1	Sorted Index Array
PØ P1 P2	Position
RØ R1	Record Counters
S1 S2 S3	Record Information Locations
T9	Lines Per Page
U	Unit #
V1 V2	Sort Value

```
4907 MULTIPLE VOLUME DIRECTORY LISTER


                M E N U:


1. LIST CREATED VOLUME DIRECTORY FILES
2. CREATE       VOLUME DIRECTORY FILE
3. SORT         VOLUME DIRECTORY FILE
4. OUTPUT       VOLUME DIRECTORY FILE
5. END

   FUNCTION NUMBER REQUIRED: 2



BE SURE MVDL SYSTEM-DISC IS PLACED IN UNIT 0
      PLACE SOURCE-DISC IN DRIVE
         AND SPECIFY UNIT NUMBER (1 OR 2):1

BE SURE TAPE IS IN INTERNAL TAPE-UNIT
PRESS [RETURN] TO CONTINUE:
```

```
CREATE


SPECIFY TAPE-FILE # FOR DIRECTORY:6
DIRECTORY ---> TAPE-FILE:6
COUNTING NUMBER OF FILES:=2
Include REMARK statements (Y/N):Y
CREATING VOLUME INFO FILE
DIRECTORY ---> VOLUME DIRECTORY FILE
```

```
LIST


SPECIFY OUTPUT DEVICE FOR LISTING:32

LIST:
                    1. VOLUME DIRECTORY FILE(S)
                    2. SORTED VOLUME DIRECTORY FILE(S) (1/2):1

                    1. ALL FILES OF SPECIFIED TYPE, OR
                    2. ONE (TO BE NAMED) FILE ONLY (1/2):1

LIST TO CONTAIN FILE- :
                    1. NAMES (DIRECTORY 0, )
                    2. INFORMATION (DIRECTORY 2, )
                    3. CONTENT
                    4. COMBINATION OF 1 AND 3
                    5. COMBINATION OF 2 AND 3
LIST FORMAT NUMBER:4
```

```
LIST OF VOLUME DIRECTORY FILES

AS OF 07-APR-83 12:02:07


MVDL/DIR/APPLIB
ORIGINAL VOLUME:
4907          DEV ID   APPLIB     VOL ID   SCRATCH                    OWNER
   620032 FREE      630784 SIZE              0 LOST     256 BLK SIZE
06-APR-83 10:35 FORMATTED
NUMBER OF FILES ON VOLUME:2

(FIRST) REMARK STATEMENT ENCOUNTERED IN PROGRAM-FILES
ON VOLUME WITH VOLUME NAME:                APPLIB
INCLUDED IN VOLUME DIRECTORY FILE: @MVDL/DIR/APPLIB


MVDL/DIR/P50DDSYS
ORIGINAL VOLUME:
4907          DEV ID   P50DDSYS   VOL ID   TEK4054D08 Ver 2.1         OWNER
    16384 FREE      630784 SIZE              0 LOST     256 BLK SIZE
25-MAR-82 08:47 FORMATTED
NUMBER OF FILES ON VOLUME:184

(FIRST) REMARK STATEMENT ENCOUNTERED IN PROGRAM-FILES
ON VOLUME WITH VOLUME NAME:                P50DDSYS
INCLUDED IN VOLUME DIRECTORY FILE: @MVDL/DIR/P50DDSYS
```

```
SORT



WHICH VOLUME DIRECTORY FILE TO SORT
(VOLUME NAME ONLY):P501D

SAVE SORTED VOLUME DIRECTORY FILE (Y/N):Y
NAME:P501D.SNAM

FIELD ITEMS:
                    1: FILE IDENTIFIER
                    2: ATTRIBUTE A/B/H
                    3: ATTRIBUTE R/U
                    4: ATTRIBUTE S/C
                    5: ATTRIBUTE N/M
                    6: TIME CREATED
                    7: TIME LAST ALTERED
                    8: TIME LAST USED
                    9: ALLOCATED SPACE
                    10: USED FILE SPACE
                    11: RECORD LENGTH
                    12: REMARKS


SPECIFY ORDER OF SORT:
(END OF SPECIFICATION=0)

                  KEY-ITEM 1: 1
                  KEY-ITEM 2: 0

SORTING TO BE DONE IN ASCENDING(=1)
                      DESCENDING ORDER(=2) (1 OR 2):1

SORTING



SORT COMPLETED

SAVING SORTED FILE

PRESS [RETURN] TO CONTINUE:
```

```
OUTPUT


OUTPUT DEVICE #:32

NUMBER OF LINES PER PAGE:30

OUTPUT:
                    1. VOLUME DIRECTORY FILE(S)
                    2. SORTED VOLUME DIRECTORY FILE(S) (1/2):2

NAME OF VOLUME DIRECTORY FILE:P50ID
NAME:P50ID.SNAM

OUTPUT TO INCLUDE:
                    1. ALL ITEMS
                    2. SELECTABLE ITEMS IN ORDER OF SPECIFICATION (1/2):2

1. USE SAME PARAMATERS AS DEFINED FOR SORT, OR
2. DEFINE ITEM OUTPUT ORDER (1/2):2

ITEM-NAMES TO BE SUPPLIED WITH OUTPUT (Y/N):Y
```

```
SPECIFY ORDER OF ITEMS:
(END OF SPECIFICATION=0)

                    1.FILE IDENTIFIER
                    2.ATTRIBUTE A/B/H
                    3.ATTRIBUTE R/U
                    4.ATTRIBUTE S/C
                    5.ATTRIBUTE N/M
                    6.TIME CREATED
                    7.TIME LAST ALTERED
                    8.TIME LAST USED
                    9.ALLOCATED SPACE
                    10.USED FILE SPACE
                    11.RECORD LENGTH
                    12.REMARKS



                    ITEM 1:1
                    ITEM 2:7
                    ITEM 3:12          INCLUDE LINE NUMBERS (Y/N):Y
                                       INCLUDE LEADING SPACES (Y/N):Y
                    ITEM 4:0
```

```
4907          DEV ID   P50DDSYS   VOL ID   TEK4054D08 Ver 2.1        OWNER
    16384 FREE      630784 SIZE              0 LOST      256 BLK SIZE
25-MAR-82 08:47 FORMATTED


FILE IDENTIFIER: AUTOLOAD
TIME LAST ALTERED: 25-MAR-82 09:12
REMARKS: 1 REM*BPLOT 50 2-D DRAFTING

FILE IDENTIFIER: DD/D0/D0
TIME LAST ALTERED: 25-MAR-82 09:11
REMARKS: 50000 REM*BDRAFT_INIT D0/D0

FILE IDENTIFIER: DD/D0/D17
TIME LAST ALTERED: 25-MAR-82 09:10
REMARKS: 50000 REM*BWORKSET_ADD_ITEM D0/D17

FILE IDENTIFIER: DD/D0/D18
TIME LAST ALTERED: 25-MAR-82 09:12
REMARKS: 50000 REM*BWORKSET_REGION D0/D18

FILE IDENTIFIER: DD/D0/D19
TIME LAST ALTERED: 25-MAR-82 09:11
REMARKS: 50000 REM*BWORKSET_CHAIN D0/D19
```

```
FILE IDENTIFIER: DD/UTIL/U0/U5
TIME LAST ALTERED: 25-MAR-82 09:02
REMARKS: 50000 REM*BPLOT_ENVIR U0/U5

FILE IDENTIFIER: DD/UTIL/U0/U6
TIME LAST ALTERED: 25-MAR-82 09:02
REMARKS: 50000 REM*BGEN_SYMBOL_LIBRARY_DWG UTIL/U0/U6

FILE IDENTIFIER: DD/UTIL/U0/U7
TIME LAST ALTERED: 25-MAR-82 09:02
REMARKS: 50000 REM*BREAD_CONFIG_&_SEI_ENVIR U0/U7


NUMBER OF RECORDS IN @MVDL/DIR/P50DDSYS.SNAM :184
PRESS RETURN TO CONTINUE
```

4907 MULTIPLE VOLUME DIRECTORY LISTER

M E N U:

1. LIST CREATED VOLUME DIRECTORY FILES
2. CREATE VOLUME DIRECTORY FILE
3. SORT VOLUME DIRECTORY FILE
4. OUTPUT VOLUME DIRECTORY FILE
5. END

FUNCTION NUMBER REQUIRED: 5

TITLE	PART NUMBER
Local Language Translator	062-6715-01 Program 5
ORIGINAL DATE: June 1982 / REVISION DATE:	EQUIPMENT, OPTIONS AND SOFTWARE REQUIRED (INCLUDING PERIPHERALS AND HOST SYSTEM)
AUTHOR: Ruud Borstel — Tektronix, Inc. Europe	4907 File Manager (dual) Optional - 4641/3 Printer

ABSTRACT

Files: 2 Program

Statements: 252

Local language translator is an aid in changing the human interface (including screen messages and prompts) from the original language (in most cases English) into a local, other language.

The program offers two modes:

1. Reporting screen messages and prompts of all quoted parts via hard copy or printer.

2. Interactive changing of text strings and reporting the changes made.

NOTE: The original files to be translated must be ASCII files. The documentation contains a utility to convert host binary files to ASCII files.

THE SOFTWARE

The software remains on a system disk and is divided over two files:

@LANGUAGE/REPORT

@LANGUAGE/CHANGE

Both programs should be loaded separately; there is no monitor available.

NOTE: The original files to be translated should be ASCII files. Included in the documentation is a utility program to convert host binary files to ASCII files or vice versa.

THE PROGRAMS

@LANGUAGE/REPORT

This program will provide a report (on screen-hard copy or printer) of all quoted parts encountered in the original (ASCII) file. As the program cannot see the difference between text strings and 4907 filenames, the latter ones will be reported as well.

The program will underline the text parts in the output, to indicate the quoted parts encountered.

@LANGUAGE/CHANGE

This program allows you to interactively change encountered text parts. A report on the changes made can be output to the screen (hard copy) or printer.

In this mode, the system disk is mounted in UNIT Ø. The disk containing original (ASCII) files is mounted in another unit.

After specifying the name of the file on which translation is to be done, the software will create a destination file (filename.TRAN), and read in program lines from the original file one by one. If a quoted part is encountered, the user may replace it with his own text. If no change is desired simply press "RETURN".

After the change is made, the program will return the "new" line (as changed) and will ask whether the change made is correct (YES/NO/QUIT).

After copying the line (with changes, if made) to the file with extension TRAN, a new line will be read in, checked on text parts, etc.. Note that the lines with 4050 or 4907 commands are skipped. The length of the text replacing the original text may not exceed the length of the original text.

Utility to convert host binary files to ASCII files or vice versa.

Below you will find a short utility to convert the program files into ASCII files. You might want to convert whole libraries at a time (e.g., "@DD/SYS#") by supplying a file name using special characters.

The trick in the utility is the APPEND "Filename"; line #,Ø) statement. (line 65522, increment is zero)

To convert ASCII back to host binary, lines 65518, 65522 and 65524 should be changed.

```
65490 REM APPEND POINT1
65491 REM APPEND POINT2
65492 INIT
65493 DIM Z$(300)
65494 REM CONVERTER
65495 PRINT "LJMOUNT ORIGINAL DISK IN UNIT: ";
65496 INPUT U1
65497 CALL "MOUNT",U1,A$
65498 PRINT "DESTINATION DISK IN UNIT: ";
65499 INPUT U2
65500 CALL "MOUNT",U2,A$
65501 CALL "UNIT",U2
65502 SAVE "REM";65490,65491
65503 CALL "UNIT",U1
65504 PRINT "JNAME OF THE FILE TO BE CONVERTED: ";
65505 INPUT A$
65506 OPEN A$,"G";1,"R",Z$
65507 IF LEN(Z$)=0 THEN 65504
65508 Z$=SEG(Z$,189,LEN(Z$)-189)
65509 IF NOT(POS(Z$,"SCRATCHLIB",1)) THEN 65512
65510 Z$=REP("",1,11)
65511 GO TO 65517
65512 IF NOT(POS(Z$,"SYSLIB",1)) THEN 65516
65513 Z$=REP("",1,7)
65514 Z$="$"&Z$
65515 GO TO 65517
65516 Z$="@"&Z$
65517 CALL "FILE",U1,Z$,Y$
65518 IF POS(Y$,"H",6)<>6 THEN 65528
65519 REM DELETE LINES
65520 DELETE 1,65489
65521 CALL "UNIT",INT(U1+0*MEMORY)
65522 APPEND Z$;65490,0
65523 CALL "UNIT",U2
65524 SAVE Z$,"A";1,65489
65525 PRINT Z$;" ---> ASCII"
65526 CALL "UNIT",U2
65527 APPEND "REM";65491,0
65528 CALL "NEXT",1,Z$
65529 IF LEN(Z$)=0 THEN 65531
65530 GO TO 65508
65531 CALL "UNIT",U2
65532 KILL "REM"
65533 DELETE 1,65489
65534 END
```

```
Local Language Change Program

Do you want the changes reported on
          H - hardcopy unit
          P - line-printer
#P
Primary address of printer : 51

Be sure drive 0 contains the Local Language System Disc

Which drive is used for input (ASCII) programs ? 1


Which file to translate : @REPORT
file @REPORT already exist on drive 1. OK to overwrite?Y
```

```
130 PRINT "LLocal Language Report Program"
130 PRINT "L                              "
130 PRINT "LLocal Language Report Program"
CORRECT (Y/N/Q) ? Y

140 PRINT "JWhich drive is used for input (ASCII) programs ? ";
140 PRINT "J                                                 ";
140 PRINT "JWhich drive is used for input (ASCII) programs ? ";
CORRECT (Y/N/Q) ? Y

170 PRINT "JInsert disc with (ASCII) Programs in drive ";U1
170 PRINT "J                                           ";U1
170 PRINT "JInsert disc with (ASCII) Programs in drive ";U1
CORRECT (Y/N/Q) ? N
170 PRINT "J                                           ";U1
170 PRINT "JInsert disc with (ASCII) Programs in drive ";U1
CORRECT (Y/N/Q) ? Y

180 PRINT "press <RETURN>";
180 PRINT "              ";
180 PRINT "press <RETURN>";
CORRECT (Y/N/Q) ? Q
```

```
Local Language Report Program

Which drive is used for input (ASCII) programs ? 1

Insert disc with (ASCII) Programs in drive 1
press <RETURN>


Which file : @REPORT

Output to (S)creen or (P)rinter ? P

Primary address of printer : 51

END-OF-FILE
```

```
Report of file : @REPORT

130 PRINT "LLocal Language Report Program"

140 PRINT "JWhich drive is used for input (ASCII) programs ? ";

170 PRINT "JInsert disc with (ASCII) Programs in drive ";Ul

180 PRINT "press <RETURN>";

200 CALL "MOUNT",Ul,A$

210 IF POS(A$,"WRITE PROTECTED",186)<>0 THEN 240

220 PRINT "JDrive ";Ul;" must be WRITE PROTECTED_KK";

240 PRINT "JJJWhich file : ";

260 CALL "FILE",0,F$,A$

280 PRINT "J***>> file ";F$;" does NOT exist"

300 IF POS(A$,"A",6)=6 THEN 330

310 PRINT "J***>> file ";F$;" is not ASCII program"

330 PRINT "JOutput to (S)creen or (P)rinter ? ";

360 IF I$<>"P" THEN 390

370 PRINT "JPrimary address of printer : ";

390 PRINT @D1:"LReport of file : ";F$;"JJ"
```

```
400 CALL "UNIT",U1
          ____

410 OPEN F$;1,"R",A$
                 _

460 IF NOT(POS(B$,"""",1)) THEN 440

510 C$="                                          "
       ________________________________________

590 C$=REP("_",I,1)
              _

610 PRINT @D1:C$;"J"
                    _

640 PRINT "JJGGEND-OF-FILEGG"
             ________________

1080 I$="_H"&I$
          __
```

TITLE		PART NUMBER
DOC1		062-6715-01 Program 6
ORIGINAL DATE 4/19/82	REVISION DATE	EQUIPMENT, OPTIONS AND SOFTWARE REQUIRED (INCLUDING PERIPHERALS AND HOST SYSTEM) 4052/4054 with 64K 4907 File Manager (dual) Optional 4641/3 Printer
AUTHOR John McCrory, Jr.	USDA Forest Service Columbia, MO	

ABSTRACT

Files: 1 Program/1 Data
4 Data (examples)

Statements: 193

Based on the five level hierarchy of the 4907 File Manager, DOC1 maintains an inventory of tape/disks, their files, and documentation pertaining thereto.

A Table of Contents file contains all the first two-level libraries under which the Documentation files will be found. Each documentation file is stored under a user-specified file level name. It contains four sequential strings:

Program name (30)

Tape or disk volume # and file (72)

Short description of program (72)

Location of documentation (72)

The program allows data entry into the Table of Contents file and creation and data entry into the Documentation file, or a listing of any or all.

OPERATING INSTRUCTIONS

Program Loading

Mount disk in UNIT Ø, with backup in UNIT 1. Issue the following commands:

```
CALL "MOUNT",Ø,M$
CALL "MOUNT",1,M$
OLD "DOC1"
RUN
```

Program Execution

The program will ask if you want to enter data into the contents file "D", or list the table of contents file "L".

"D" - Will allow you to enter a data entry routine. It asks for the 3 other levels, the program name, volume and file of where located, and a short description of what the program does and the location of the documentation. Enter all these and respond to data OK?

"Y" - Yes, and the record is written and program starts over.

"N" - No, and the record is not written and program starts over.

"E" - End, and the record is written and the program stops.

"L" - Will list the categories in the table of contents file. This allows you to then enter a category by number and will select levels 1 and 2. You can then enter data "D", or list "L" the main program documentation file.

A list at this level will list out all the files containing the first two levels that have been entered into the main documentation file. You now may print out "ALL" the references under the first two headings or "RETURN" which will request the next three level names. Once these are entered, the only record printed will be the one specified.

DATA STRUCTURE

All data files are binary. The program file is ASCII. The program is loaded with:

```
CALL "MOUNT",Ø,M$
OLD "DOC1"
RUN
```

The files are:

```
"SCRATCHLIB/DOC1"    --   program (required)
"@TABLE/CONTENTS"    --   data (required)

"@GRAPH/SCATTER/DBH/CALSS/PRETTY"  --  data (example)
"@GRAPH/SCATTER/PLOT/P3DBH2/SPECIES"  --  data (example)
"@GRAPH/SCATTER/BASAL/AREA/GROWTH"  --  data (example)
"@GRAPH/SCATTER/USING/SANDER/DBH59"  --  data (example
```

The example files are for demonstration and to familiarize the user with the program. The last four files can be "KILL"ed when starting a new documentation data base.

"TABLE/CONTENTS" is required. It can be altered at any time to fit the needs of the particular installation. It is a random binary file and requires a record number to alter. Be sure not to overwrite any records that have files located beneath them in the file naming hierarchy. If you want to add files, simply make "TABLE/CONTENTS" large enough to accommodate additions past any current end-of-file.

"DOC1" can be altered to reflect file sizes desired by your shop. Just be consistent throughout the alterations.

Data is stored in "TABLE/CONTENTS" in one 23 byte long character string "S$". The program also requires a record number. "S$" is composed of the first level hierarchy followed by a "/" character as a separator followed by a second level hierarch name. These three are combined together to make "S$".

In the program documentation file the first two levels of the file identifier are drawn from "S$". Third, fourth, and fifth levels are entered by the user to construct the final file identifier which will label a single file containing documentation information to be accessed by either a group open or single open. The actual unique record is written into arrays D$(30), E$(72), F$(72), and G$(72).

Two disks are suggested -- the program one with "DOC1" and "TABLE/CONTENTS", and a backup. The program is designed to operate with two disks. This can be altered at the points needed.

METHODS

The whole point of this program is to create a series of files containing documentation information. Each file is named according to five levels of hierarchy. Once each file is created it can be opened either individually based upon all five levels of hierarchy, or in a group, based upon two levels given in a table of contents file. The documentation files can then be printed out according to a read statement.

The table of contents file is random access. It calls for a record number and calls next for the two upper level hierarchies needed. Data is entered and the record is written.

These two upper hierarchies are used to control the naming of the file to be created. The table of contents file describes all the possible two level headings that the documentation files will be named under.

The rest of the program calls for a data entry into character strings using combination commands to line up the strings into five level file identifiers. The data used in each file is asked for by the program and then written after the appropriate questions asked.

Contents of documentation files can be printed out using a group open or an individual open. Under a group open, the operator answers "ALL" at line 76Ø. Under an individual open, the operator is given a directory of the documentation files in line 72Ø. and lines 94Ø through 116Ø allow for the entry and printout of individual files.

REFERENCES

CARTER, JOHN, "GROUP OPEN ON THE 4907", TEKNIQUES, MAY 1, 1980, Vol. 4 No. 3, p. 14.

INTERNAL DATA STORAGE

Variable	Used to Store	Type
T$	Disk Patameters	String
S$	Table of Contents Record	String
D$	Program Name in Documentation File	String
E$	Tape/Disk Volume Numbers and File Numbers	String
F$	Short Description of Program	String
G$	Location of Documentation	String
I$	File Status Messages	String
U$	CALL "NEXT" File Status	String
Q$	Alphabetic Answer	String
I	Record Number in Table of Contents File	Simple
R$	User Library Name in Documentation File -- First Level Hierarchy	String
P$	Second Level Hierarchy	String
A	Table of Contents Record Selector	Simple
X1	Position of "#" Symbol in S$	Simple
X$	Third Level in Documentation File	String
Y$	Fourth Level in Documentation File	String
Z$	Fifth Level in Documentation File	String
X	X-Coordinate of Cursor During Data Entry	Simple
Y	Y-Coordinate of Cursor During Data Entry	Simple

Examples

The following examples show the beginning of the program, a prompting statement asking for "D" for data entry or "L" for listing of table of contents file. Output to each response, contents of each documentation file are printed after responding "ALL" to the second set of questions. Sample respones to all the questions asked.

```
DATA ENTRY (D) INTO TABLE OF CONTENTS FILE OR LIST (L)
TABLE OF CONTENTS FILE--YOU NEED TO LIST (L) BEFORE
YOU CAN ACCESS LEVELS THREE, FOUR AND FIVE
L

 1 FORT/PROGRAM
 2 GRAPH/BAR
 3 GRAPH/BOXPLT
 4 GRAPH/CIRCLE
 5 GRAPH/CIRCLESQ
 6 GRAPH/DISK
 7 GRAPH/FLEXO
 8 GRAPH/LINE
 9 GRAPH/SCATTER
 10 GRAPH/SLIDE
 11 GRAPH/STOCKCHT
 12 INPOUT/DISC
 13 INPOUT/PRINT
 14 INPOUT/TAPE
 15 INTERFACE/AMDAHL
 16 INTERFACE/TERMINAL
 17 JCL/BACKUP
 18 JCL/IEBGENER
 19 JCL/LOAD
 20 JCL/SAS
 21 MENU/DISC
 22 MENU/INTERFACE
 23 SAS/PROGRAM
 24 STAT/ANOVACOV
 25 STAT/COVARIANCE
 26 STAT/EXPECTATIO
 27 STAT/FLEXO
 28 STAT/NORMALDIST
 29 STAT/PRINT
 30 STAT/REGRESS
 31 STAT/SIMULATION
 32 STAT/TDIST
 33 UTILITY/DISC
 34 UTILITY/FILDIRTORY
 35 UTILITY/MSI

 SELECT ONE
 12
```

```
DO YOU WISH TO DO DATA ENTRY (D) INTO FULL PROGRAM FILE OR
TO LIST WHAT HAS BEEN SELECTED UNDER HEADING LIST (L)
D


FIRST TWO LEVELS ARE @INPOUT/DISC/
INPUT THIRD LEVEL
BINARY
INPUT FOURTH LEVEL
ASCII
ENTER FIFTH LEVEL
CONVERSION
ENTER PROGRAM NAME
BIN.ASCII.CONVERI_____________
ENTER TAPE/DISK VOLUME NUMBERS AND FILE NUMBERS
VOLUME_1109005==FILE_20_VOLUME_1109006==FILE_20_____________________
ENTER SHORT DESCRIPTION OF WHAT PROGRAM DOES
AND LOCATION OF DOCUMENTATION
THIS_PROGRAM_CONVERTS_BINARY_TO_ASCII_RECORDS_AND_STORES_THEM_ON_DISK___
DOCUMENTATION_IS_LOCATED_AFTER_THE_1109005==FILE_20._________________
DATA OK Y/N/E?
Y
```

```
DATA ENTRY (D) INTO TABLE OF CONTENTS FILE OR LIST (L)
TABLE OF CONTENTS FILE--YOU NEED TO LIST (L) BEFORE
YOU CAN ACCESS LEVELS THREE, FOUR AND FIVE
L


 1 FORT/PROGRAM
 2 GRAPH/BAR
 3 GRAPH/BOXPLT
 4 GRAPH/CIRCLE
 5 GRAPH/CIRCLESQ
 6 GRAPH/DISK
 7 GRAPH/FLEXO
 8 GRAPH/LINE
 9 GRAPH/SCATTER
 10 GRAPH/SLIDE
 11 GRAPH/STOCKCHT
 12 INPOUT/DISC
 13 INPOUT/PRINT
 14 INPOUT/TAPE
 15 INTERFACE/AMDAHL
 16 INTERFACE/TERMINAL
 17 JCL/BACKUP
 18 JCL/IEBGENER
 19 JCL/LOAD
 20 JCL/SAS
 21 MENU/DISC
 22 MENU/INTERFACE
 23 SAS/PROGRAM
 24 STAT/ANOVACOV
 25 STAT/COVARIANCE
 26 STAT/EXPECTATIO
 27 STAT/FLEXO
 28 STAT/NORMALDIST
 29 STAT/PRINT
 30 STAT/REGRESS
 31 STAT/SIMULATION
 32 STAT/TDIST
 33 UTILITY/DISC
 34 UTILITY/FILDIRTORY
 35 UTILITY/MSI

 SELECT ONE
 33
```

```
DATA ENTRY (D) INTO TABLE OF CONTENTS FILE OR LIST (L)
TABLE OF CONTENTS FILE--YOU NEED TO LIST (L) BEFORE
YOU CAN ACCESS LEVELS THREE, FOUR AND FIVE
D


INPUT RECORD #
9
INPUT FIRST LEVEL HIERARCHY
GRAPH
INPUT SECOND LEVEL HIERARCHY
SCATTER
DATA OK Y/N/E?
Y
```

```
DO YOU WISH TO DO DATA ENTRY (D) INTO FULL PROGRAM FILE OR
TO LIST WHAT HAS BEEN SELECTED UNDER HEADING LIST (L)
L
UTILITY/DISC/CONVERT/ASCII/BINARY
UTILITY/DISC/MERGE/COPY/OH441
UTILITY/DISC/FLAGSHIP/CONVERT/PROGRAM
UTILITY/DISC/FORMAT/BEFORE/USEDISK
UTILITY/DISC/CREATE/DIFFERENT/FILES
UTILITY/DISC/PRINT/FIRST/THREE
UTILITY/DISC/THIRD/IEBGENER/CONVERT
UTILITY/DISC/FOURTH/IEBGENER/CONVERT
UTILITY/DISC/SIXTH/IEBGENER/CONVERT
UTILITY/DISC/SEVENTH/IEBGENER/CONVERT
UTILITY/DISC/EDIT/IN/FILES
UTILITY/DISC/COPY/FULL/DISK
UTILITY/DISC/BINARY/ASCII/CONVERT
UTILITY/DISC/SECOND/IEBGENER/CONVERT
UTILITY/DISC/FIFTH/IEBGENER/CONVERT
UTILITY/DISC/EIGHTH/IEBGENER/CONVERT
THESE ARE THE PROGRAMS UNDER HEADING @UTILITY/DISC#
ENTER "ALL" FOR AN ALL PRINTOUT OR ENTER NOTHING
TO GO TO LAST THREE LEVELS--TO BE ENTERED BY YOU SHORTLY
ALL
COPY.DBH59N
VOLUME 1109005--FILE 14  VOLUME 1109006--FILE 14.
TO COPY IVAN'S SHORT-LEAF PINE DATA INTO A BINARY FORMAT.  DOCUMENTATION
IS LOCATED AFTER TAB 1109005--FILE 14.

MERGE.COPY
TAPE VOLUME 1109005--FILE 15  TAPE VOLUME 1109006--FILE 15.
TO MERGE A NEW RECORD INTO FILE OH441.  DOCUMENTATION IS LOCATED AFTER
TAB 1109005--FILE 15.

CONVERT.IEBGENER
VOLUME 1109005--FILE 45  VOLUME 1109006--FILE 45.
```

TO CONVERT UMC IEBGENER RECORDS TO TEKTRONIX DATA. DOCUMENTATION IS AFTER VOLUME 1109005--FILE 45.

DISC.FORMAT
VOLUME 1109001--FILE 5 VOLUME 1109002--FILE 5.
TO FORMAT FLOPPY DISK ON 4907 FILE MANAGER. DOCUMENTATION IS LOCATED AFTER TAB 1109001--FILE 5.

MOUNT.CREATE
VOLUME 1109003--FILE 8 VOLUME 1109004--FILE 8.
TO MOUNT TWO DISCS AND CREATE FILES. DOCUMENTATION IS LOCATED AFTER TAB 1109003--FILE 8.

PRINT.THREE
VOLUME 1109005--FILE 40 VOLUME 1109006--FILE 40.
PRINTS OFF FIRST THREE RECORDS OF A FILE. DOCUMENTATION IS AFTER VOLUME 1109005--FILE 40.

CONVERT.IEBGENER.2
VOLUME 1109005--FILE 48 VOLUME 1109006--FILE 48.
TO CONVERT DATA TO DISK FILES. DOCUMENTATION IS LOCATED AFTER TAB 1109005--FILE 48.

CONVRT.IEBGENER.3
VOLUME 1109005--FILE 49 VOLUME 1109006--FILE 49.
TO CONVERT DATA TO DISK FILES. DOCUMENTATION IS LOCATED AFTER TAB 1109005--FILE 49.

CONVERT.IEBGENER.5
VOLUME 1109005--FILE 51 VOLUME 1109006--FILE 51.
TO CONVERT TAPE FILES TO DISK FILES. DOCUMENTATION IS AFTER TAB 1109005--FILE 57.

CONVERT.IEBGENER.6
VOLUME 1109005--FILE 52 VOLUME 1109006--FILE 52.

TO CONVERT TAPE FILES TO DISK FILES. DOCUMENTATION IS AFTER TAB 1109005--FILE 52.

DISC.EDIT
VOLUME 1109001--FILE 8 VOLUME 1109002--FILE 8.
TO CORRECT, DELETE OR INSERT RECORDS IN A RANDOM TEKTRONIX DISC.
DOCUMENTATION IS LOCATED AFTER TAB 1109001--FILE 8.

MOUNT.COPY
VOLUME 1109003--FILE 10 VOLUME 1109004--FILE 10.
TO DO A FULL COPY FROM DISK UNIT 0 TO UNIT 1. DOCUMENTATION IS LOCATED AFTER TAB 1109003--FILE 10.

BIN.ASCII.CONVERT
VOLUME 1109005--FILE 29 VOLUME 1109006--FILE 29.
TO CONVERT FILE P3DBH2 INTO ASCII CHARACTERS FOR STORAGE ON TAPE.
DOCUMENTATION IS LOCATED AFTER TAB 1109005--FILE 29.

CONVERT.IEBGENER.1
VOLUME 1109005--FILE 47 VOLUME 1109006--FILE 47.
TO CONVERT UMC DATA TO TEKTRONIX DATA. DOCUMENTATION IS BEHIND TAB 1109005--FILE 47.

CONVERT.IEBGENER.4
VOLUME 1109005--FILE 50 VOLUME 1109005--FILE 50.
TO CONVERT TAPE FILES TO DISK FILES. DOCUMENTATION IS LOCATED AFTER TAB 1109005--FILE 50.

CONVERT.IEBGENER.7
VOLUME 1109005--FILE 53 VOLUME 1109006--FILE 53.
TO CONVERT JCL/IEBGENER TO DISK FILES. DOCUMENTATION IS AFTER 1109005--FILE 53.

```
DO YOU WISH TO DO DATA ENTRY (D) INTO FULL PROGRAM FILE OR
TO LIST WHAT HAS BEEN SELECTED UNDER HEADING LIST (L)
L
GRAPH/SCATTER/DBH/CLASS/PRETTY
GRAPH/SCATTER/PLOT/P3DBH2/SPECIES
GRAPH/SCATTER/BASAL/AREA/GROWTH
GRAPH/SCATTER/USING/SANDER/DBH59
THESE ARE THE PROGRAMS UNDER HEADING @GRAPH/SCATTER#
ENTER "ALL" FOR AN ALL PRINTOUT OR ENTER NOTHING
TO GO TO LAST THREE LEVELS--TO BE ENTERED BY YOU SHORTLY
ALL
DBH.CLASS.1978.80.GROWTH
VOLUME 1109005--FILE 21  VOLUME 1109006--FILE 21.
TO PLOT 1978 DBH CLASS AND 78-80 GROWTH.  DOCUMENTATION IS LOCATED AFTER
TAB 1109005--FILE 21.


PLOT.1978.80.GROWTH
VOLUME 1109005--FILE 20  VOLUME 1109006--FILE 20.
TO PLOT 1978 DBH VERSUS 1978-80 GROWTH.  DOCUMENTATION IS LOCATED AFTER
TAB 1109005--FILE 20.


BASAL.AREA.GROWTH
VOLUME 1109005--FILE 22  VOLUME 1109006--FILE 22.
PLOTS 1978 BASAL AREA VS. 1978-80 BASAL AREA GROWTH.  DOCUMENTATION IS
LOCATED AFTER TAB 1109005--FILE 22.


SCAT.DIAG
VOLUME 1109003--FILE 7  VOLUME 1109004--FILE 7.
TO DRAW SCATTER DIAGRAMS.  DOCUMENTATION IS LOCATED AFTER TAB 1109003--
FILE 7.
```