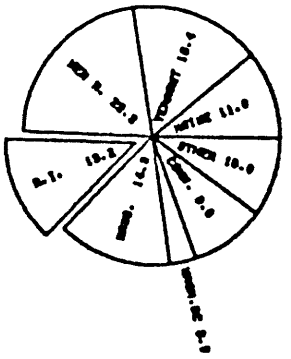
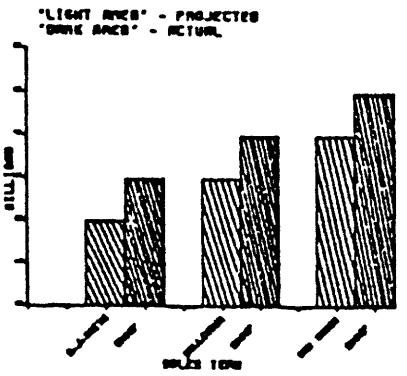
PLOTWARE-Z

ENGINEERING AND BUSINESS GRAPHICS FOR MICRO, MINI, AND MAINFRAME COMPUTERS

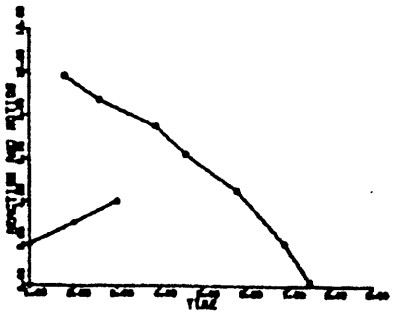
EASTERN AREA SALES



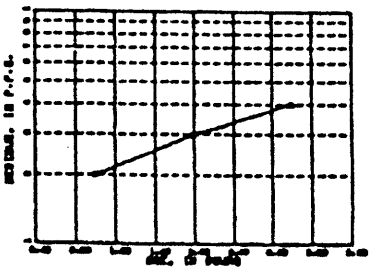
SALES PROFILE - FIRST QUARTER
PROJECTED AND ACTUAL



REACTION LEVEL AND
RESIDUAL MOTION
VERSUS TIME



PARTICULATE RESIDUE
BY AGE OF COMPOUND



THE ENERCOMP COMPANY

P.O. BOX 28014
LAKEWOOD,CO. 80228

DRAWN BY PLOTWARE-Z
ON A NIPLOT DRT-4
NO HUMAN DRAFTING
NO PHOTO. REDUCTION

REVISION 1.4

SERIAL#

(C) 1982

INTRODUCTION

Welcome to the world of COMPUTER GRAPHICS. The PLOTWARE-z® system consists of this manual, the software on the disks you received with your system, and your imagination. There are many ways to use the PLOTWARE-z®. You can draw charts and graphs of almost unlimited complexity. You can become artistic and draw complete pictures by using data from many sources. You can compose slides with the (optional) ornate fonts, by using everything from textbook quality letters and Old English letters to many special characters. You can also attach PLOTWARE-z® to your own programs.

If you are an experienced computer user, and wish at this time to begin drawing graphs you may "LOAD UP", enter "CHART" on your console and get started. Remember to press the ESC button to get out of trouble or to get helpful hints.

If you prefer to peruse this manual before proceeding, we will begin with explanations of the construction of the PLOTWARE-z® MANUAL.

A: INSTALLATION SECTION:
This section explains the first steps of getting started, and customizing your system to a special graphics peripheral or a special system environment.

B: INTERFACE SECTION:
There are many possibilities in approaching the use of PLOTWARE-z®. This section tells more about how you can use it and helps you choose the approach which will best suit your needs. You may find it beneficial to come back to this section later, as you get more proficient in the world of computer graphics. You will probably discover new, more effective ways to do what your graphics dispalys.

C: MENU DRIVEN SECTION:
This is the quickest way to display your data. You probably will find a chart here among the many charts available that will suit your own needs. It all starts with the DRAW command.

D: "GRAPHIT"SECTION:
This is a question and answer method of making charts. If you like to "FILL IN THE BLANKS", use GRAPHIT. GRAPHIT contains a data editor and tutorial hints which are built into most of the major GRAPHIT functions. It all starts with the GRAPHIT command in the CHART program.

E: EXPANDING THE MENU:
You will eventually want to expand the menu of the menu-driven approach if you use menus extensively. This lets you build specialized graphs, in whatever complexity you desire, and retain the easy approach to the PLOTWARE-z® system which is inherent in this MENU-DRIVEN portion. Make this the key to your very own special graphics system.

F: COMMAND FILE SECTION:
This section acquaints you with COMMAND FILES, the heart of the PLOTWARE-z® system. It talks of how they work, and shows how to build them.

G: COMPILER LIBRARY SECTION:
This is a "must" for any person who wishes to attach the system directly to their program. If you are just beginning in computers, avoid this section until you become familiar with one of the traditional compiler languages.

H: DEVICE SPECIFICATION SECTION:
If you need to attach a new plotting device (plotter, word processing printer, CRT, camera, etc.), or if you wish to modify the way the built-in software handles, all of these types of devices (programmers often refer to this as "drivers" or "handlers") use this section. It is also a good way to just "fiddle around" with the inner-most workings of PLOTWARE-z®. This section includes an example of how to build a new device into the software. This section should probably be avoided if you are just beginning in computers.

I: DEFERRED DEVICE SECTION:
This section is similar to the DEVICE SPECIFICATION SECTION, except that it is strictly for "deferred devices" like dot matrix printers. Here the plot must be prepared for drawing with just one trip through the printer, no matter how complex your drawing has been. This is often referred to as a "vector to raster scan conversion". It is the secret to attaching this type of device to the system. It includes the source code and examples you will need to make this type of addition.

J: OPERATION SECTION:
This is the nitty-gritty information about how PLOTWARE-z® works, operationally. This section includes details about disk files, overlay files, and the like. Although this may bore you, remember that computers must be properly groomed, and the necessary details for this grooming are usually found by understanding the relationship between the software and your own particular computer system. As with any software package, PLOTWARE-z® may appear to emit certain errors which are undefined in this manual. This is because some simple rule of the computer has been violated and the error is actually being displayed by some "support software" such as the operation system of your machine. A good example would be found if you tried to store your data on a disk which was write protected. Each computer system usually has a unique way to show errors of this nature. Your knowledge of this section combined with your knowledge of your own unique system and the knowledge of the data you wish to display, will lead to the answer for these "undocumented problems".

K: TECHNICAL REFERENCE SECTION:
This is a detailed description of each module of PLOTWARE-z*. This section is sometimes referred to as the SYNTAX DESCIPTION of the system. It explains the "grammar rules" for all of the modules available in the command files or in the compiler library. This section is a must for the programmer making new command files or writing a new program which will access the compiler library. It might also be referred to by main-frame programmers as a "programmer's reference".

L: INSTALLATION DEPENDENT SECTION:
By now, it should be apparent that the author of PLOTWARE-z* made this unique graphics program easy to modify and simple to expand. As you build the system into the package that suits you, take special care to place all of the listings that come from INSTALL or the other sections into this section. It should hold the directory content listing of your "work disks", including all items pertaining to your own customized system. This is YOUR section. It is in your best interest to keep it up to date.

INSTALLATION SECTION

PLOTWARE-z is designed to be a universal tool to draw on all graphics devices that may be attached to the micro-computer. It is not directed specifically at the scientist or engineer or at the financial planner or computer programmer. Hence, it has many different ways to meet the needs of each group of users. Because of this variety of uses, it is suggested that you do these two things before going into the actual installation and usage of your software.

1. **Install your graphics hardware and verify that it works.** Most devices have simple check-out software with them.

2. **Select the manner in which you will use this package.** You may, of course, use all of it to its maximum capacity, as any single part does not exclude the use of any other part. Usually, one will use one approach in preference to others. This will make the system requirements somewhat less when he makes up his "work disks" since the parts not used may be omitted.

First, before proceeding any further, be sure to archive your system disks. **NEVER USE THE ORIGINAL DISKS AS WORK DISKS!** Besides the possibility of Murphy's law showing up and ruining your only copy of the software, ENERCOMP requires that you return the original, unmodified disks when periodic updates are distributed.

If your dealer has already installed this package into your own particular system, skip this next part and go directly to the INTERFACE SECTION.

INSTALLING YOUR HARDWARE

(This part presents installation requirements)

First consider some basic facts. These are pertinent to all systems.

1. A certain amount of hardware and software knowledge is essential : how your operating system works, how each hardware device is attached, how the corresponding operating system "connection" is made via software. REMEMBER - NO SOFTWARE, NOT EVEN PLOTWARE-z , WILL OPERATE ON AN IMPROPERLY INSTALLED SYSTEM.

2. You must have at least **ONE GRAPHICS DEVICE** in order to use PLOTWARE-z .

3. You **may install as many graphics devices as you like.** There is no practical limit other than any limit your hardware may impose on you. The actual limit is devices PLOTWARE-z will consider is about 64,000 - attached at any one time.

4. DEVICES are grouped into two categories: DEFERRED DEVICES and REAL-TIME DEVICES.

5. DEFERRED DEVICES include all hardware which must print or display all of the picture in one step as the media (such as paper in a printer) is passed in one direction only. This means that the picture is drawn in "passes" by some device such as a print head of scanning electron beam in a regular pattern from right to left or left to right as the media is passed from top to bottom. In no case will the media be "backed up" to make more impressions on it. These devices are typically called "raster scan devices" and include dot matrix printers, certain graphics camera devices, and the like.

6. REAL-TIME DEVICES actually draw the picture as it is presented. That means they seem to actually draw a line, by some means peculiar to them, and therefore have the freedom of moving in any direction on the "page" or media used to display the picture. These include: GRAPHICS CRTs, PEN PLOTTERS, WORD PROCESSING PRINTERS, VECTOR FILM CAMERAS, RASTER SCAN CAMERAS with accessible raster pages or "smart controllers", and DISK FILE INTERMEDIATE STORAGE.

7. All devices used are associated with a device TYPE. The more popular TYPES are installed in PLOTWARE-z* as you receive it. You may choose to install these in a way that is more convenient to you. As you eliminate devices that don't pertain to your system, you also make the disk space requirements smaller.

8. Device TYPE 0,1,2, and 3 are reserved for DEFERRED DEVICES which means, of course that you may have a maximum of four of these types of devices at any given time. It also meansthat you may have four "modes" of operation of any one device at any given time instead of four devices. This usually is used to define density of pictures as you display them. For instance, TYPE 0 is "super dense" mode for the OKIDATA 84 printer and TYPE 3 is "LEAST DENSE" (hence fastest) mode for that printer.

9. TYPE 4 and all higher numbers up to about 64000 are reserved for REAL-TIME devices. The ENERCOMP COMPANY suggests that you use the following convention.

TYPE 4	--	Your pen plotter.
TYPE 5	--	Your graphics CRT.
TYPE 6	--	Your auxiliary plotter, usually a word processing printer

DEVICE "TYPE" MAY BE SELECTED BY YOU when you exercise the program INSTALL.

DEVICE SPECIFICS

(This part presents specific requirements for device types)

Here are some specific device requirements that will assist you in determining the suitability of a device for graphics work. In general, if the device does not meet these requirements, the device is not a graphics device. In particular, that device will not work with PLOTWARE-z®. This also specifies any specific requirements for devices to be used.

1. **DEFERRED DEVICES** - Dot matrix printers must be assigned to the "PRINTER" device of your computer if they are to use the standard software without modification or further installation. This is usually referred to as the "CP/M LIST DEVICE" in CP/M or as LPT1 in MSDOS or PCDOS. It can, of course, be any other possible device if your system can logically assign it to the "PRINTER" via some "IO BYTE" capability such as STAT in CP/M or MODE in MSDOS or PCDOS.

   In multi-processing systems such as MP/M, the printers should be specifically specified via the appropriate statement before beginning your graphics session.

2. **DEFERRED DEVICES** - Dot matrix printers must be one of the available devices supported if no additional programming is done. A partial list is shown here:

   ANADEX
   CITOH
   DATASOUTH DS180
   EPSON with GRAFTRAX
   OKIDATA
   NEC
   STAR GEMINI

   RUN INSTALL TO INSERT THE PROPER PRINTER TYPE PRIOR TO USING A DEFERRED DEVICE.

   The software includes the NAME of the device as part of its necessary install sequence. Be sure that you know the brand name of your device prior to installation.

3. **DEFERRED DEVICES** - Dot matrix printers and similar devices **not specifically supported must be included in your system by installing a special "replacement subroutine" as an ASSEMBLY language code module.** This is a very simple procedure (for people familiar with ASSEMBLY language of their machine) as the skeleton code is included on your disk software. The devices must have the following characteristics to be installed via this method:

   a. They must be "dot addressible" by placing one or more "dots" evenly spaced across the path of the carriage, and be able to do this by manipulating from one to

eight of the "print head wires" or "dot driving devices" to do this. Multiple passes, wherein a second row of dots are laid in between the first row of dots may be made to generate super-dense graphics pictures. This is sometimes called "interlaced" operation. It prints the even dots on one pass of the print head, then prints the odd dots on the second pass. This allows the printer to overlap the dots and produce a very smooth line.

b. The device must have a consistent "dot density" both in the vertical and horizontal directions. These densities need not be identical.

c. The device must be able to perform a "graphic carriage return and line feed" or some similar sequence that will serve to begin an new uniform entry of "dots" on the next segment of the page or media.

4. REAL TIME devices must be connected to your computer as one of the standard operating system peripherals unless you install them specially with the INSTALL program, for specific IO port addresses, or by modification of the plotter source code in PLOTSPEC.SRC.

5. Some devices may be connected only to a few of the available devices. For instance, the screen of the TELEVIDEO 1603 or 803 must only be connected to the DISPLAY CONSOLE. Such restrictions are dependent on your own unique hardware, of course.

6. Each device supported must have the following characteristics in order to be "custom installed" by you.

a. It must be able to draw a line (vector drawing) or be able to "turn on" a dot within its matrix on command.

b. The grid on which the dots are located or on which the lines are drawn must be consistent in both the X and Y directions. The X and Y may be different in measure. This means that a consistent "steps per inch" or "dots per inch" must be available for the device.

c. REAL-TIME device extensions in addition to the ones required above may be included by using the NEWPEN statement. These usually include such extensions as COLOR, LINE STYLE often specified in the CORE GRAPHIC standards, and special FILL with PATTERN commands and similar commands.

INSTALLATION FILES

(This part describes disk files that are used by installation program : INSTALL)

Now that you have patiently read through all of the ground rules, you can proceed into the installation procedure for your work disks.

If you can get all of PLOTWARE-z® onto your work disk, do so. You can remove the excess later.

If not, place these files onto your first work disk:

INSTALL.COM
INSTALL1.OVR
INSTALL2.OVR
INSTALL3.OVR
INSTALL4.OVR

Then you may enter INSTALL as a CP/M command.

If even these will not fit on your disk, include INSTALL.COM on a separate disk, assign the .OVR files to your default disk, and initiate INSTALL from the separate disk (for example : B:INSTALL).

After the INSTALL command is issued to CP/M, the program will talk you through the rest of the procedure. Some miscellaneous details of this program follow:

a. **The CUSTOMIZE procedure lets you select specific devices.**

b. **The PLOTCOMP procedure compiles the results of "a".**

c. **"a" must be followed by "b" to completely customize your devices.** This is by design, as it allows you to modify the file "PLOTSPEC.DAT" between the two steps if you desire to enhance the operation of any of the devices for your own particular reasons.

d. **The MODIFY program allows you the ability to directly modify parameters within the system.** It is usually not required in most installations, but is included to allow the user to avoid such clumsy software as the DDT package.

INSTALLATION DETAILS

(This part explains "difficult" errors and describes operating details of the INSTALL procedure)

Skip this unless you are curious about the workings of the install procedure. Read it if you have problems.

Normally the INSTALL program will ask you questions, install the system, give you specific information, give you specific directions, then terminate.

Certain errors may occur, however. Usually these are self explanatory, however, such as :

"INSTALL1.OVR not on the disk."

Other errors may occur that are not explained here, such as :

"BDOS ERROR ON A:R/O"

which means to the experienced computer user: "YOU PROBABLY CHANGED DISKS DURING THIS PROCESS".

These are operational errors and must be resolved by you, since they pertain to your own computer system and all possible computer systems can not be covered in this manual.

Here are a few possibilities that you should consider:

You did something wrong, just start over, and try again.

You had insufficient disk space. You need 5 to 10 K of space on the "default disk" to run INSTALL.

There were improper files on disk. Follow the directions, and re-do the procedure.

There was a "write protect" device on the disk. Correct this and re-do the procedure.

The primary purposes of INSTALL are as follows:

1. The generation of a disk file called PLOTSPEC.DAT which is the "brains" behind the operation of PLOTWARE-z® on your own particular system. You MUST have this file present whenever PLOTWARE-z® is used. It will be very small unless you have many plotter devices installed in your system.

2. The generation of the optimal disk configuration for your own computer system. This will allow the most efficient operation of the package on your system.

3. Inform you of any "last minute" operation notes. Any hints, any manual changes, or any explanation of vague areas in the documentation (which exist in all systems, of course) will be found here. When users call in hints, or ideas, or when new devices are implemented in special ways, these notes will be included in the INSTALL program so that you may have the very latest information on your system when it arrives.

When PLOTSPEC.DAT is built, several things go into it:

a. A concise description of your deferred devices.

b. A complete description of your REAL-TIME devices.

c. A "snapshot" of your computer system (memory size, etc.) as needed for efficient operation of overlays, etc.

NEVER MODIFY THE FILE PLOTSPEC.DAT -- it could result in system failure. ALWAYS modify only those parameters needed to build this file (in PLOTSPEC.SRC or by INSTALL, for instance).

INSTALL builds this file from several sources:

1. Your responses to the INSTALL questions.

2. The file PLOTSPEC.SRC used to describe real-time devices. This file is specifically made available to the user in order to allow him the freedom of modifying current devices or installing new devices in whatever manner he chooses. This is sometimes calles "device drivers" or "device handlers" by main-frame and minicomputer programmers. Remember to incorporate any desired changes into this file prior to using the "PLOTCOMPPROCEDURE"in INSTALL, if you wish to make changes.

3. The file which handles your dot matrix printer(s). These go by different names and may be modified if you have special dot-matrix printer requirements. This information in PLOTSPEC.DAT includes routine information items such as

   a. Type of interface.

   b. Dots-per-inch in both directions.

   c. The kinds of "resolution density" available - if your printer has this feature.

INSTALLING THE FILE PLOTSPEC.DAT ON YOUR WORKING DISKS

(This describes, in summary, the building of the file onto a working disk)

These are the normal steps used to modify the file PLOTSPEC.DAT and implement the changes.

The process for **REAL-TIME devices:**

1. **Run INSTALL and select the "CUSTOMIZE ..." option in order to obtain the file PLOTSPEC.SRC. You must have the file INSTALL1.OVR on your default disk when this is done.**

2. If you wish to enhance the standard devices in any way, or if you must do special things to implement your device, use your favorite text editor or word processing program to modify PLOTSPEC.SRC. BEWARE OF "HIGH-BIT" errors. Do not use a text editor that sets the "high bit" (bit 7) of the ASCII data. Most processors find that this means something very special - it ignores any item with this bit set. A common mistake is found when WORDSTAR is used in

the "D" or document mode instead of the "N" or non-document mode to modify this file.

3. Run INSTALL and specify the "PLOTCOMP PROCEDURE" to process the file PLOTSPEC.SRC and compile it into PLOTSPEC.DAT. You will have a chance to get a listing of the code, if desired.

4. Copy the new file PLOTSPEC.DAT onto the working disk. This is the only information that you need on that disk from this installation process.

5. Store the PLOTSPEC.SRC file in your archives and include the PLOTSPEC.SRC listing in the "INSTALLATION DEPENDENT" section of this manual.

The process for deferred devices:

1. Use INSTALL to install the desired dot matrix printer device. If the device is in the standard software, this step is sufficient to install it.

2. Your file PRINTMAP.COM must be on the working disk. This file can be modified in the areas specified as available to you for modification. After modification, place the new PRINTMAP.COM onto your working disk.

3. If the dots-per-inch specification has changed during this process, you must re-run the INSTALL program to re-specify the dots-per-inch specifications to the system.

4. If you have changed printers, you must re-run INSTALL to obtain the proper printer environment.

Now you can make any necessary changes in your work disk as directed by INSTALL. The only files affected by this procedure are PLOTSPEC.DAT, CHART.COM, and PRINTMAP.COM.

CHECKING OUT THE INSTALLATION:

(This part describes a simple way to verify that your hardware works in the proper fashion. It is used after "custom installations" as well as after standard menu selection installations.)

For PLOTWARE-z® packages:

Verify installation by performing the following steps:

YOU ENTER	THE COMPUTER RESPONDS
CHART	COMMAND:
PLOTS TYPE n	

(use your own value for n)

COMMAND:

PLOT 3 6 2 3 0 2 0 0 2 END

COMMAND:

At this time, if a real time device was selected, a right triangle should have been drawn, with a three inch leg and a six inch leg. If you used a device that shared the computer console with you (like an APPLE II screen) some conflict may have been encountered with the graphic screen activity versus the text activity.

EXIT

At this time the program will terminate. If you used a deferred device, the right triangle will be drawn before the program terminates.

For USER PROGRAMS (all other systems - not PLOTWARE-z®)

Run the program GRAFTEST. It will prompt you to answer the TYPE question, then it will allow you to run the same "triangle test" as well as a simple test of the character generator and a few other things.

AND IF IT DOESN'T WORK? Try these remedies:

1. You may have a hardware error or an improperly installed device. Correct this.

2. You may have incorrectly installed the software. Be sure there are no missing disk files. Correct any errors.

3. Perform the test until it works.

IF THE TRIANGLE IS DISTORTED: The legs may not measure 3 inches by 6 inches. If so, continue:

1. For REAL-TIME devices: Often one uses a different size device. For example, on a CRT or graphics card, one may install a variety of monitors, any of which may have a different "ASPECT RATIO" than another. You may also have a plotter with METRIC stepping motors instead of INCH stepping motors. Your plotter may have a switch that allows you to choose different step sizes. There is a simple solution to this in the file PLOTSPEC.SRC. Refer to

the DEVICE SPECIFICATION SECTION for information on the "MECHANICAL" specification of this file. A standard practice in ENERCOMP has been to "size" the CRT screen to a 10 inch height in order to allow a compatibility with most hard-copy devices. This obviously makes the triangle smaller, but is usually acceptable to those who want this compatibility. A side effect of this is that it usually makes the screen appear to be 12 or 13 inches "wide".

2. DEFERRED DEVICES. Surprisingly enough, ENERCOMP has found that some dot-matrix printers will vary in the step size, or "dots per inch" between supposedly similar models. This is usually only in the dots-per-inch across the page. It usually results from a different "micro-code" program in the printer, and can be explained by the manufacturer's revised program. The quickest way to fix this is to measure the triangle and form the ratio needed to correct the error. For example, if the 6 inch leg measured 5 inches when drawn, the ratio would be 6/5. Now run INSTALL again. When you choose the printer type, enter a "dots-per-inch in the Y direction" (which is across the page) that is 6/5 of the value previously entered.

   If the triangle is distorted (just has lines not connected or not straight), you may have one of the following deferred device problems:

1. A HARDWARE PROBLEM. Usually a handshake problem. This occurs most frequently with older EPSON printers with GRAFTRAX software which had "bugs" in it.

2. You are not sending enough bits to the device. On EPSON and STAR GEMINI and similar printers, you must send all 8 bits of the byte in order to correctly make the graph. There is a switch on most interfaces that allows the "eighth bit to be grounded". This switch will create problems in graphics mode, if set.

For real-time devices:

1. You may have made an error while "customizing the system". Correct it.

2. You may need to specify a different "step size". Do so.

INTERFACE SECTION

There are several different ways to use PLOTWARE-z. Each way is called an "interface method". Each method described here is directed at a specific approach to using PLOTWARE-z.

APPROACH	INTERFACE METHOD
1. Computer asks questions. User answers questions.	"DRAW"
2. Computer shows "screen". User "fills in the blanks".	"GRAPHIT"
3. User prepares commands into disk file. Computer reads "COMMAND FILE" as a "program".	"COMMAND FILE"
4. User writes traditional "program". "COMPILES" it and"LINKS" it with PLOTWARE-z library.	"COMPILER LIBRARY"

Each of these interface methods will be described separately. Use this section to build a basic familiarity with each way and to become familiar with the basic computer operating details of each.

DRAW:

Draw is the easiest way to get started with PLOTWARE-z. It asks questions, then produces graphs and charts from your answers. The "standard graphs" are all built into the system when it is delivered. However, if you wish to add to these "menus", it is very easy to do so. A basic understanding of the COMMAND FILES is required and some means by which to re-arrange the menus is necessary. This re-arrangement may be done with any text editor such as WORDSTAR. Completely new menus may be included and existing menus may be customized specifically for your needs.

Examples of using this approach are found in the "MENU DRIVEN SECTION (C)". Explanation of the process of building new menus and modifying existing ones is found in the "EXPANDING THE MENU SECTION (E)".

GRAPHIT:

This program is meant to build "generic plots" and to prepare a command file in order to allow further enhancements of the chart. In addition to this, graphit contains a handy data editor which allows one to edit the data from a variety of sources into the necessary disk file format. The format of this approach is to present screens for "filling in the blanks", and to produce the necessary command files to generate charts based on these screens. All menus have explanations about their function. All

menus which produce graphs or charts include a pictorial representation of the final graph together with all available options. This does not require a graphics screen to function. Examples of this approach are included in the "GRAPHIT SECTION (D)".

COMMAND FILE

All PLOTWARE-z operations except the compiler library begin or end in this section. It is the most important feature of PLOTWARE-z from an operational viewpoint. All other interface methods (except the compiler library) generate the COMMAND FILE language that is discussed here. All other processes also leave this language available to you in a source file format so that you may modify or enhance it in any way you choose. This allows you the total flexibililty of dynamically altering the files to give your results "that custom look". The process consists of generating a "COMMAND FILE" by using a text editing program or some other program and then allowing the program "CHART".COM" to process this file and yield the requested chart. The command file consists of a kind of "language" with simple, English-like verbs which have modifiers to further explain the command. A complete explanation of this language is found in the "TECHNICAL REFERENCE SECTION (K)" as well as examples for using command files.

This approach can also be used by programs written in BASIC or other interpretive languages and compiled programs. The process consists of building the command file from the program and building any data files which the command file may reference. (The data may be included in the command file if desired, however). The program then "chains" to the "CHART.COM" program which processes the command file. The CHART sequence may then restart the original program, if so desired. There is an example of this approach later in this section showing a partial derivative and directional derivative function described in a 3-D manner through PLOTWARE-z.

This entire "chaining" process revolves around a base file called "PLOTWARE.OVR" and if the user has files named by this name or if the file resides on some disk improperly, very strange things may occur. These can be corrected by removing any file by the name "PLOTWARE.OVR" which is on the default disk. This file is built when needed and erased later in normal operations.

When the user generates "PLOTWARE.OVR" to control chaining BACK to his original program after PLOTWARE-z processes his (generated) files, the user needs only to generate a file with at least one record in it. The first byte of this record is a binary designation of the disk on which his program resides (0 = default disk, 1= "A", etc.) and positions 2-12 define the program name which is to be restarted. The remainder of the file should be used to house any common data necessary to his program. The PLOTWARE-z system will leave this data intact. The command file should be named COMMON.CMD and the file PLOTWARE.OVR should be erased after the original program regains control of the system.

Another feature of this language is its ability to automatically "RESTART" after some catastrophic (to the computer system) event. If the computer experiences a power failure during the PLOTWARE-z process, the user needs only to restart the phase program (such as CHART or PRINTMAP) that was last active. The entire process will then continue as though nothing occurred. This is an effective way to run the lengthy processes sometimes necessary an retain a level of system confidence.

This feature uses tne file PLOTWARE.OVR and if the (interrupted) process is NOT to be restarted, this file must be manually removed to avoid the automatic restart of the LAST PLOTWARE-z process whenever a PLOTWARE-z phase again executes in the machine. This is especially true if logic or data errors (such as feeding negative numbers to a LOG-LOG chart) result in a fatal error in a PLOTWARE-z phase program. The error occurs and leaves the restart file intact so that you can correct the logic or data error (remove the negative numbers in the above example, for example) and then restart the process without the necessity for all of the preliminary "set-up" exercises.

All of the COMMAND FILES used to illustrate the TECHNICAL REFERENCE SECTION (E) are listed to allow you to see how each works. These are at the end of that section.

COMPILER LIBRARY

This method allows the programmer to directly attach PLOTWARE-z to his program. This is normally done by the process of "LINKING" his program to the necessary pieces of the compiler library. Thisprocess is normally explained in the appropriate compiler manual, but will be covered here by example.

The graphic process usually begins by the statement:

```
CALL PLOTS (A,B,ITYPE)
```

and ends by the statement:

```
CALL PLOT (0., 0., 999)
```

which terminates that graphics program normally.

The syntax associated with the calls is found in the"TECHNICAL REFERENCE SECTION (K)".

The library makes external calls for arithmetic subroutines. This allows the programmer to include common subroutines for his programs as well as PLOTWARE-z and to take advantage of any special hardware he might have available. Input/output subroutines used by PLOTWARE-z are included and external routines are not needed.

EXAMPLE : BASIC AND PLOTWARE-z

This example shows the use of a BASIC program written in interpretive MICROSOFT basic using PLOTWARE-z as an auxiliary program. The program generates a 3-D data file and then generates the command file to draw the picture.

```
1 IPEN=3:GOSUB 2000
3 XMAX=-9999:YMAX=-9999:ZMAX=-9999:XMIN=9999:YMIN=9999:ZMIN=9999
5 YIS=-8:XIS=8:YIS2=8:XIS2=-9:NEW=(YIS2-YIS)/(XIS2-XIS)
10 FOR X=14 TO -18 STEP -1:FLAG=0
20 REM YNEW=YIS+(X-XIS)*NEW
30 REM FOR Y=YNEW TO 17 STEP .5
31 FOR Y=-13 TO 17 STEP .5
40 GOSUB 1000
50 PRINT#1,X;Y;Z;IPEN
52 IF X<XMIN THEN XMIN=X
53 IF X>XMAX THEN XMAX=X
54 IF Y<YMIN THEN YMIN=Y
55 IF Y>YMAX THEN YMAX=Y
56 IF Z<ZMIN THEN ZMIN=Z
57 IF Z>ZMAX THEN ZMAX=Z
58 IPEN=2
60 NEXT Y
62 PRINT X
65 IPEN=3:NEXT X
66 PRINT#1,XMIN;YMIN;ZMIN;" 3":PRINT#1,XMAX;YMAX;ZMAX;" 3":CLOSE #1
67 PRINT XMIN;YMIN;ZMIN;XMAX;YMAX;YMAX
68 PRINT #2,"EXIT":CLOSE #2
69 IF NA1$="COMMON.CMD" THEN GOTO 3000:REM IF NAMED "COMMON.CMD" THEN CHAI
70 STOP
1000 WX=3*EXP(-.1*((Y+10)^2)-.02*((X+10)^2))
1010 RX=4*EXP(-.1*((X+10)^2)-.3*((Y-6)^2))
1020 SX=3*EXP(-.1*((Y+8)^2)-.02*((X-10)^2))
1030 UX=-6*EXP(-.09*(X-5)^2-.09*(Y-7)^2)
1040 Z=UX+RX+WX+SX
1050 RETURN
2000 INPUT "INPUT COMMAND FILE NAME ('name'.CMD)";NA1$
2005 NA1$=NA1$+".CMD"
2010 INPUT "INPUT DATA FILE NAME ('name'.DAT)";NA$
2011 Q=CHR$(34)
2015 NA$=NA$+".DAT"
2020 INPUT"INPUT PLOT DEVICE (NUMBER)";PD
2022 OPEN"O",#1,NA$:OPEN "O",#2,NA1$
2025 PRINT #2,"PLOTS TYPE ";PD
2030 PRINT #2,"PLOT 6 5 -3 END"
2035 PRINT #2,"FACTOR .5"
2040 PRINT #2,"ZPERSX 80 22.9183 71.6198 25 0"
2045 PRINT #2,"FONT ROMAN"
2050 PRINT #2,"FONTFQ 1 1.2"
2055 PRINT #2,"FONTFQ 2 0"
2060 PRINT #2,"FONTFQ 3 0"
2065 PRINT #2,"FONTFQ 4 1"
2070 PRINT #2,"FONTFQ 5 0"
2075 PRINT #2,"FONTFQ 6 0"
```

```
2080 PRINT #2,"SMBOLC 23.5 5 2.8 'ENERCOMP' 90 99"
2085 PRINT #2,"PLOT (-1 0 0 0)";Q;NA$;Q
2100 RETURN
3000 I=40000!:REM BEGIN CHAINING PROGRAM AT 40000
3010 FOR J=1 TO 61
3020 READ K:POKE I,K:I=I+1:NEXT J
3030 FOR J=1 TO 24
3040 POKE I,0:I=I+1:NEXT J
3050 DEF USR0=40000!
3060 A=USR0(A)
3070 DATA 17,113,156,14,15,213,205,5,0,209,254,255,202,0,0,33,0,1
3080 DATA 235,213,229,14,26,205,5,0,225,209,1,128,0,235,9,229,213
3090 DATA 14,20,205,5,0,209,225,183,194,0,1,195,82,156,0,67,72,65
3100 DATA 82,84,32,32,32,67,79,77
```

EXAMPLE - COMPILER LIBRARY

This program illustrates the use of the COMPILER library. It is written in FORTRAN with Microsoft's F80 compiler.

```
      LOGICAL L(255)
      CALL MAINR
C- THIS MUST ALWAYS HAVE "ROOM" AT THE LOWER PART OF MEMORY FOR
C- THE TRANSIENT REAL-TIME DEVICE ROUTINES WHICH BEGIN AT 145H
      END
      SUBROUTINE MAINR
C- THIS IS THE ACTUAL PROGRAM
C- ALWAYS COMPILE YOUR PROGRAMS LIKE THIS, OR LINK A "DUMMY"
C- ROUTINE TO BEGIN WITH THAT HAS ENOUGH ROOM FOR THIS.
      WRITE(5,1)
    1 FORMAT(' Enter the index of your device ("TYPE") :')
      READ(5,2)I
    2 FORMAT(I4)
      CALL PLOTS(0.,0.,I)
C- THIS INITIALIZES THE DEVICE
      CALL PLOT (0.,0.,3)
      CALL PLOT (4.,0.,2)
      CALL PLOT (4.,6.,2)
      CALL PLOT (0.,0.,2)
C- THIS JUST DREW A 4 INCH BY 6 INCH TRIANGLE
      CALL SMBOLL(5.,0.,.5,'ABCDEF123456',90.,12)
C- THIS JUST PLOTTED 12 CHARACTERS 1/2 INCH HIGH AT 90 DEGREES
      CALL PLOT (0.,0.,999)
C- THIS TERMINATES
      END
```

The following submit file will compile and link the program

```
F80 =TESTLIBR
L80 TESTLIBR,PLOTLIB/S,FORLIB/S,TESTLIBR/N/E
```

To execute the program (the standard S.E.T.S test), enter:

```
TESTLIBR
```

MENU DRIVEN SECTION

The menu driven section allows the person to make direct question and answer graphs. There is very little to explain about this except to show how to begin the session:

Enter:

CHART

When prompted to respond, enter:

DRAW

The computer will then be in the MENU DRIVEN SECTION.

You may enter the ESC command to look at directories, get general help information, or perform other tasks, as indicated.

You may enter the ? code to get specific help about certain options of this format.

To take full advantage of this section, use it as you would use any software package, with one exception:

a. Make a note to yourself of EVERY little thing you would change if you could do so.

b. After compiling all of these notes, or at least enough of them to make the process worthwhile, don't curse the PLOTWARE-z authors as you have other authors that have left you limited to the "standard options" of typical software. Instead, include these changes into the menus that you have found that you use the most.

c. Familiarize yourself with the COMMAND FILE language. You will probably already have accomplished this by this time.

d. Use the "EXPANDING THE MENU SECTION" as a guide to effect your changes.

e. Now you can use your OWN CUSTOMIZED GRAPHICS SOFTWARE!

THE "GRAPHIT" SYSTEM

The "GRAPHIT" system consists of a separate program call GRAPHIT.COM with its corresponding overlay file called OVERLAYG.OVR, and the necessary code to "chain" to the CHART system to process the command files that are built in GRAPHIT.

The system consists of two different types of solutions. One is a collection of "screen menu" routines that allow you to specify the necessary items to build a chart or graph for all of the common types of charts and graphs. It is very fast because all you ever need to enter is the name of the file which houses your data.

The other type of routine is a set of "editing" routines that allow you to take data from other disk files or from the computer console keyboard and build data files for charting and graphing.

In both cases, no manual is needed to run the system as it presents sufficient information for its "screen menus" and gives a compact set of editing rules whenever any error is made, allowing the user to correct the error.

BUILDING DATA for the GRAPHIT system.

The GRAPHIT system produces graphs and charts from a varietyof data sources.

a. Data entered manually.

b. Data from electronic "spread sheet" programs.

c. Data from disk files created by BASIC programs or other programs.

d. Other ASCII data files.

Data taken from other files must be from files having these characteristics:

a. The data must be in the ASCII format.

b. The data must be numeric, without embedded commas, dollar signs, asterisks, or other special characters.

c. The data must be in "columnar" format. For each record or line of data, the data must occur in the same position each time. The data does not actually have to occur in the same PHYSICAL columns, but must always be in the same position with respect to the other numbers. For instance, the user may always refer to the THIRD number or column in the record as the X value in a graph, and the FIFTH number or column as the Y value, when drawing line graphs.

d. The data may be included with certain alphabetic annotation or other items which need to be removed from consideration. This would include ALL items which are not numeric in nature. This data may be "blanked out" before the data is editted into the users data file. The positions within the line must be known to do this. EXAMPLE : Take the following data extracted from a printed listing from a "spool" file:

May, 1981	14.7	12.3	residual: 2.1
June, 1981	12.4	11.1	residual: 1.7

Here the dates and the item "residual:" must be "blanked out" from the rest of the numeric data, leaving:

14.7	12.3	2.1
12.4	11.1	1.7

This is done by specifying where the "blanked out" data resides, withrespect to the character position with the record.

In the above case, position 1-17 would blank the data (with a couple of extra spaces for longer months), and 41-49 would blank out the "residual:" item. These are also referred to as "columns" (which should not be confused with "columns" of numbers. The first column of numbers contains 14.7 and 12.4, the second column of numbers contains 12.3 and 11.1, etc.)

The process is as follows:

1. Select the "BUILD GRAPH DATA FILE" entry from the menu. (enter a "5").
2. Enter a name for your data file.
3. Enter a title for your data file.
4. Enter a short description for the file.

The optional items 3 and 4 have been included in each file to allow for more detailed information to be stored in each file. This will make future use of the file more simple since the data is adequately identified.

The GRAPHIT program itself will explain each response requested. If any typical errors are made, such as selecting a file name which already exists on disk, you will be told about this and your options will be explained.

You must then tell what kind of file you wish to build:

1. A file for X-Y graphs (each record holds 2 numbers)

2. A file for simple bar charts or for pie charts (each record holds a bar or segment title and a number.)

3. A file for "clustered" or "stacked bars" (2 high in this case). Each record holds a bar title and from 2 to 10 numbers. Multiple stacked bars are handled in another way.

You must then select how you will enter the data (manually or from a disk file). Manual entry is done by the data editor.

When you select the automatic extraction from a disk file source, you will need to know these things:

1. How many records you want to "skip" at the beginning of the file. These are usually titles.

2. The exact location of fields to be blanked out, as shown above.

3. The relative occurrence of the columns of numbers in the data file with respect to how you will use the data. For instance, you may choose to select the third column of the data file for your "X" values (which is your FIRST column), and the second column of the data file for your "Y" (which is your SECOND column). Don't be confused by which column is the data file and which is yours.

These responses must be given in the above order.

You may then decide just how you will process each record from the data file:

a. You may look at each record and decide whether it is to be processed or bypassed.

b. You may process all remaining records in the file.

c. You may process only a certain number of records in the file.

If you select option "a" or "c", you may decide again at the end of the records processed how to continue processing. You can go back to the beginning of the file, you can skip more records, you can redefine the "blank areas" or you can redefine now your "column structure" will match the data file's "column structure". The system explains this when you get there.

When the data is extracted (either by reading through the entire data file, or by your choice to stop extracting data), the process begins the "data editting phase".

EDITTING EXISTING DATA FILES

The GRAPHIT system expects its data to be in a format that includes not only the data but certain items about the data structure (is it an X-Y data file, a PIE chart data file, etc.) as well as the extra information about the data that you may have

supplied when the data file as originally built.

The editor (when selected originally by menu item "4") must be supplied with a name of the data file. If the data file is not in order, a message will be displayed. This is not necessary if you have elected to "BUILD" a data file through the menu item number "5". It already has your data file name.

You then may do one of the following five things:

a. List your data (1 or more lines.)

b. Print your data on the printer (1 or more lines)

c. Delete a line of your data.

d. Insert a new line into your data.

e. Replace a line of data with a new line.

f. "AUTO" insert data into the data file.

g. Save the edited data file.

The experienced user will see a remarkable similarity between this approach and the editors used in the more common BASIC interpreters. This is designed to eliminate the need to learn more and more editors and their peculiarities.

The data will have a "line number" which begins with 1 and proceeds upward. All operations must refer to this line number (except the SAVE operation).

Only the first letter of each operation is needed when editting (as explained by the program when it runs), and if an error is made, the editor re-displays the rules of editting.

The only exception to the single letter command is found in the "SAVE" command. You must first enter an "empty" command (RETURN key only), and then enter S or SAVE. This is to avoid the accidental "saving" of a file and the subsequent need to re-edit it, due to the relative close proximity of the "S" key to the "D" and "R" keys.

The file must be "SAVED" to exit this procedure (unless you press the control-c key, which will cause your data file to remain as it was at the last "SAVE" command, and will exit to the operating system.)

An example:

NOTE THAT ALL COMPUTER "VERBAGE" IS IN DARKER LETTERS.
NOTE: ALL OPERATOR ENTRIES ARE UNDERLINED!!!!

Let's enter some data manually into the system.

We begin by starting GRAPHIT:

```
A>GRAPHIT

PLOTWARE-z 1.3GI COPYRIGHT (C) 1982 EMERCOMP CO.
DATA STORAGE AVAILABLE (NUMBERS):  8263
```

It responds with a menu:

```
   ****************************
   **   GRAPH  GENERATION    **
   ****************************

ALL FUNCTIONS SHOW GRAPH AND OPTIONS ON SCREEN

SELECTION     FUNCTION
 NUMBER       TO BE DONE

   1        X-Y LINE GRAPHS
   2        BAR CHARTS
   3        CIRCLE CHARTS (PIE CHARTS)
   4        EDIT GRAPH DATA
   5        BUILD GRAPH DATA FILE
   6        STOP THE PROCESS AND EXIT

SELECT A FUNCTION (BY NUMBER) : 5
```

After selecting option "5", some explanation appears:

```
   ******************************************
   **  MAKING A NEW GRAPH/CHART DATA FILE  **
   ******************************************

THIS OPTION ALLOWS YOU TO MAKE A NEW DATA FILE, ON YOUR
DISK, AND FILL THIS FILE WITH DATA FROM ANOTHER DISK FILE
OR FROM YOUR KEYBOARD - BY ENTERING IT MANUALLY. YOU MUST
FIRST ENTER A FEW BASIC ITEMS REQUIRED FOR ALL DATA FILES.

WHAT WILL BE THE NAME OF YOUR DATA FILE ON DISK? TESTFILE.DAT
```

here we choose the title:

```
NOW ENTER A TITLE FOR YOUR DATA FILE (YOU CAN USE
UP TO 20 CHARACTERS): This is the test case for the Manual Example.
```

Here we enter some verbage so future users will know about this data:

```
NOW ENTER ANY "NOTES" THAT YOU WISH TO INCLUDE IN THIS
FILE AND WHICH MAY BE HELPFUL TO YOU IN THE FUTURE.
Manual test case example data.
```

Let's select our data type (we want an X-Y line graph):

```
NOW ENTER THE TYPE OF DATA YOU WILL BE ENTERING.
 "1" FOR X - Y DATA.
 "2" FOR CHART DATA (PIE OR BAR WITHOUT COMPARISON BARS.
 "3" FOR BAR CHART DATA WITH 2 TO 9 BARS BEING COMPARED.
TYPE: 1
```

The system then responds with an explanation of the data editor:

```
THE DATA MUST NOW BE ENTERED.  THIS CAN HAPPEN IN TWO
DIFFERENT WAYS.  YOU CAN ENTER IT DIRECTLY, OR YOU CAN
EXTRACT IT FROM A DISK FILE, THEN GO INTO THE "EDIT"
OPTION AND MAKE ANY CHANGES NECESSARY.

  "1" TO ENTER IT MANUALLY.
  "2" TO EXTRACT DATA FROM DISK FIRST.

METHOD:1
```

We choose "method 1"

```
YOU CAN NOW ENTER YOUR DATA WITH THE "A" OPTION OF THE EDIT ROUTINE.
       ************************************
       **   EDITING EXISTING GRAPH DATA   **
       ************************************

THIS OPTION ALLOWS YOU TO EDIT GRAPH OR CHART DATA THAT
HAS ALREADY BEEN CREATED.  YOU EDIT THE DATA BY REFERRING
TO THE "LINE NUMBER" OF THE DATA.  THIS IS LISTED AT THE
FAR LEFT COLUMN OF THE SCREEN.  YOU MAY DELETE THAT LINE,
INSERT ANOTHER LINE JUST BEFORE THAT LINE, OR REPLACE THAT
LINE. ALL DATA IS ENTERED "FREE FORM" - THAT IS, SEPARATED
BY COMMAS OR SPACES. ENTER "D" TO DELETE, "I" TO INSERT,
OR "R" TO REPLACE DATA, FOLLOWED BY THE LINE NUMBER, E.G.

    D  1234

WHICH WILL DELETE THE LINE NUMBERED "1234". OTHER OPTIONS:
```

"L" (line #) TO LIST THE DATA, "P" (line #) TO PRINT THE
DATA, "A" TO AUTO-INSERT DATA. NO NUMBER AFTER THE OPTION
STARTS THE PROCESS WITH THE TITLE INFORMATION. WHEN FINISHED,
A NULL ENTRY (RETURN KEY ONLY) WILL SAVE THE (NEW) DATA.
ANY "OPTION ERROR" WILL RE-DISPLAY THIS EXPLANATION.

As suggested, we choose the "auto" means to enter the data. Note upper case letters will be used in the system. Only the "A" is needed.

ACTION:A

PRESS "RETURN" KEY ONLY TO EXIT AUTO-INSERT MODE.
AUTO-INSERT LINE# 1
ENTER 2 NUMBERS:1.23 3.44
2:1.25 4.21
3:1.271 4.97
4:1.284 65.62
5:1.296 5.6
6:1.31 4.9
7:

Note that the long explanations occur only on the first entry. Subsequent entries are prompted only by the line number.

Let's list the data:

ACTION:LIST

DISK FILE NAME: :TESTFILE.DAT, HAS 12 VALUES, 6 LINES:X - Y DATA
TITLE:This is the test cas
NOTE:Manual test case example data.

1	1.23000	3.44000
2	1.25000	4.21000
3	1.27100	4.97000
4	1.28400	5.62000
5	1.29600	5.60000
6	1.31000	4.90000

It seems that we left a couple of data elements out of the file--between lines 1 and 2. Let's insert them.

ACTION:I 2

ENTER 2 NUMBERS:1.234 3.52

The list confirms the action:

```
ACTION:L

DISK FILE NAME: :TESTFILE.DAT, HAS    14 VALUES,     7 LINES:X - Y DATA
TITLE:This is the test cas
NOTE:Manual test case example data.
     1        1.23000        3.44000
     2        1.23400        3.52000
     3        1.25000        4.21000
     4        1.27100        4.97000
     5        1.28400        5.62000
     6        1.29600        5.60000
     7        1.31000        4.90000
```

One more time to feel secure:

```
ACTION:A 3

PRESS "RETURN" KEY ONLY TO EXIT AUTO-INSERT MODE.
AUTO-INSERT LINE#     3
ENTER 2 NUMBERS:1.236 3.57
     4:1.241 4.02
     5:_
```

It worked:

```
ACTION:LIST

DISK FILE NAME: :TESTFILE.DAT, HAS    18 VALUES,     9 LINES:X - Y DATA
TITLE:This is the test cas
NOTE:Manual test case example data.
     1        1.23000        3.44000
     2        1.23400        3.52000
     3        1.23600        3.57000
     4        1.24100        4.02000
     5        1.25000        4.21000
     6        1.27100        4.97000
     7        1.28400        5.62000
     8        1.29600        5.60000
     9        1.31000        4.90000
```

Can we insert data on the front of the file? (before line 1?):

```
ACTION:AUTO 1

PRESS "RETURN" KEY ONLY TO EXIT AUTO-INSERT MODE.
AUTO-INSERT LINE#     1
ENTER 2 NUMBERS:1.21 3.41
     2:1.23 3.477
     3:_
```

It works here also:

```
ACTION:LIST

DISK FILE NAME: :TESTFILE.DAT, HAS   22 VALUES,   11 LINES:X - Y DATA
TITLE:This is the test cas
NOTE:Manual test case example data.
    1        1.21000        3.41000
    2        1.23000        3.47700
    3        1.23000        3.44000
    4        1.23400        3.52000
    5        1.23600        3.57000
    6        1.24100        4.02000
    7        1.25000        4.21000
    8        1.27100        4.97000
    9        1.28400        5.62000
   10        1.29600        5.60000
   11        1.31000        4.90000
```

How about insertion at the end of the file?:

```
ACTION:A 12

PRESS "RETURN" KEY ONLY TO EXIT AUTO-INSERT MODE.
AUTO-INSERT LINE#   12
ENTER 2 NUMBERS:1.32 4.83
   13:1.33 4.67
   14:_
```

Looks right here too:

```
ACTION:L

DISK FILE NAME: :TESTFILE.DAT, HAS   26 VALUES,   13 LINES:X - Y DATA
TITLE:This is the test cas
NOTE:Manual test case example data.
    1        1.21000        3.41000
    2        1.23000        3.47700
    3        1.23000        3.44000
    4        1.23400        3.52000
    5        1.23600        3.57000
    6        1.24100        4.02000
    7        1.25000        4.21000
    8        1.27100        4.97000
    9        1.28400        5.62000
   10        1.29600        5.60000
   11        1.31000        4.90000
   12        1.32000        4.83000
```

It appears that line 2 was entered erroneously. Let's replace it with some proper data:

ACTION:R 2

ENTER 2 NUMBERS:1.22 3.43

A list shows that it worked:

ACTION:L

```
DISK FILE NAME: :TESTFILE.DAT, HAS   26 VALUES,   13 LINES:X - Y DATA
TITLE:This is the test cas
NOTE:Manual test case example data.
    1       1.21000        3.41000
    2       1.22000        3.43000
    3       1.23000        3.44000
    4       1.23400        3.52000
    5       1.23600        3.57000
    6       1.24100        4.02000
    7       1.25000        4.21000
    8       1.27100        4.97000
    9       1.28400        5.62000
   10       1.29600        5.60000
   11       1.31000        4.90000
   12       1.32000        4.83000
```

Since line 5 is very similar to line 4, let's delete it, for appearance sake, on the graph:

ACTION:D 5

List it:

ACTION:L

```
DISK FILE NAME: :TESTFILE.DAT, HAS   24 VALUES,   12 LINES:X - Y DATA
TITLE:This is the test cas
NOTE:Manual test case example data.
    1       1.21000        3.41000
    2       1.22000        3.43000
    3       1.23000        3.44000
    4       1.23400        3.52000
    5       1.24100        4.02000
    6       1.25000        4.21000
    7       1.27100        4.97000
    8       1.28400        5.62000
    9       1.29600        5.60000
   10       1.31000        4.90000
   11       1.32000        4.83000
   12       1.33000        4.67000
```

Before we save the file, let's print it on the printer:

```
ACTION:PRINT

HOW MANY LINES TO PRINT:999

DISK FILE NAME: :TESTFILE.DAT, HAS   24 VALUES,   12 LINES:X - Y DATA
TITLE:This is the test cas
NOTE:Manual test case example data.
    1:        1.21000        3.41000
    2:        1.22000        3.43000
    3:        1.23000        3.44000
    4:        1.23400        3.52000
    5:        1.24100        4.02000
    6:        1.25000        4.21000
    7:        1.27100        4.97000
    8:        1.28400        5.62000
    9:        1.29600        5.60000
   10:        1.31000        4.90000
   11:        1.32000        4.83000
   12:        1.33000        4.67000
```

To save it, first press "return" only, then enter "S" or "SAVE" (followed by "return"):

```
ACTION:_

ENTER "SAVE" TO SAVE FILE, OR ANOTHER ACTION REQUEST:SAVE
```

Now let's see what kind of graph we get:

```
A>GRAPHIT

PLOTWARE-z 1.3GI COPYRIGHT (C) 1982 ENERCOMP CO.
DATA STORAGE AVAILABLE (NUMBERS):  8263
```

Now let's build a data file from another data file. An easy way is to take your printed report, whether it is from BASIC, PL1, or ..., and direct its output to "SPOOLED PRINT" as the following list was done. An editor then extracted page 17 for our use.

```
A>TYPE TITANIUM.DAT
                TITANIUM SHIPMENTS  -      PAGE 17

          Shipping Depot : ANSLOVILLE WAREHOUSE
```

LINE # 1	LINE # 2	LINE # 3	LINE # 4	LINE # 5	DAY SAMPLE
236	341	213	298	305	JUNE 1
247	347	236	259	277	JUNE 2
245	365	209	227	323	JUNE 3
231	349	218	240	276	JUNE 4
215	340	235	283	307	JUNE 5
207	360	206	304	302	JUNE 6
213	358	223	279	279	JUNE 7
229	339	232	237	323	JUNE 8
244	352	204	228	274	JUNE 9
248	364	228	262	310	JUNE 10
237	344	227	300	300	JUNE 11
221	343	204	296	281	JUNE 12
208	363	232	255	322	JUNE 13
209	354	222	226	273	JUNE 14
222	338	206	243	312	JUNE 15
239	357	235	286	297	JUNE 16
248	362	217	304	283	JUNE 17
243	341	210	276	321	JUNE 18
227	347	237	235	272	JUNE 19
212	365	212	229	314	JUNE 20
207	349	214	266	295	JUNE 21
216	340	236	301	285	JUNE 22
233	361	208	294	320	JUNE 23
246	358	219	252	272	JUNE 24
247	339	234	225	316	JUNE 25
234	352	205	246	292	JUNE 26
217	364	225	288	287	JUNE 27
207	344	231	303	319	JUNE 28
211	343	204	273	272	JUNE 29
226	364	229	233	318	JUNE 30

After selecting "BUILD DATA..", and saying that it would be an automatic process, we enter a title, short description and the decision for "bar chart data".

```
*****************************************
**  MAKING A NEW GRAPH/CHART DATA FILE  **
*****************************************
```

THIS OPTION ALLOWS YOU TO MAKE A NEW DATA FILE, ON YOUR DISK, AND FILL THIS FILE WITH DATA FROM ANOTHER DISK FILE OR FROM YOUR KEYBOARD - BY ENTERING IT MANUALLY. YOU MUST FIRST ENTER A FEW BASIC ITEMS REQUIRED FOR ALL DATA FILES.

WHAT WILL BE THE NAME OF YOUR DATA FILE ON DISK? TITANIUM.002

NOW ENTER A TITLE FOR YOUR DATA FILE (YOU CAN USE UP TO 20 CHARACTERS): TITANIUM, ANSLOVILLE

NOW ENTER ANY "NOTES" THAT YOU WISH TO INCLUDE IN THIS FILE AND WHICH MAY BE HELPFUL TO YOU IN THE FUTURE.
SHIPMENT STUDY, ANSLOVILLE, FOR DAYS 7-11 LINES 1,2,3 AND 5

```
NOW ENTER THE TYPE OF DATA YOU WILL BE ENTERING.
 "1" FOR X - Y DATA.
 "2" FOR CHART DATA (PIE OR BAR WITHOUT COMPARISON BARS.
 "3" FOR BAR CHART DATA WITH 2 TO 9 BARS BEING COMPARED.
TYPE: 3
```

We will chart four bars, for line 1, 2, 3, and 5:

```
NOW ENTER HOW MANY BARS WILL BE COMPARED ON YOUR CHART:4
```

The system requests "where will you get your data?":

```
THE DATA MUST NOW BE ENTERED.  THIS CAN HAPPEN IN TWO
DIFFERENT WAYS.  YOU CAN ENTER IT DIRECTLY, OR YOU CAN
EXTRACT IT FROM A DISK FILE, THEN GO INTO THE "EDIT"
OPTION AND MAKE ANY CHANGES NECESSARY.

  "1" TO ENTER IT MANUALLY.
  "2" TO EXTRACT DATA FROM DISK FIRST.

METHOD:2

ENTER THE NAME OF THE DISK FILE FROM WHICH THE DATA
WILL BE EXTRACTED: TITANIUM.DAT
```

Examination of the printed report shows that seven lines of title information need to be skipped:

```
YOU MAY "SKIP" SOME RECORDS AT THIS POINT. HOW MANY? 6

**SKIPPED:                    TITANIUM SHIPMENTS          PAGE 17
**SKIPPED:
**SKIPPED:             Shipping Depot : ANSLOVILLE WAREHOUSE
**SKIPPED:
**SKIPPED:  LINE # 1     LINE # 2     LINE # 3     LINE # 4     LINE # 5
 DAY SAMPLE
**SKIPPED:
```

For this case, all alphabetic data is beyond our 4 columns, so we will not discard any data:

```
NOW THE DATA WILL BE EXTRACTED.  IT IS EXPECTED TO BE
IN "COLUMNS" IN THE DISK FILE.  THESE ARE COLUMNS OF
OF NUMBERS THAT ARE SEPARATED BY SPACES OR COMMAS.
YOU MAY ALSO IGNORE A POSITION OR GROUP OF POSITIONS
AS YOU EXTRACT THE DATA.  THIS MUST BE SPECIFIED AS
A BEGINNING CHARACTER POSITION, THROUGH AN ENDING
CHARACTER POSITION (IN ORDER TO DISCARD "TRASH")
```

BEGINNING POSITION (OR NOTHING, IF NOT DESIRED): _

The bar titles are taken from the date - here we say it is in position 70-72:

YOUR DATA FILE WILL HAVE 4 COLUMNS OF DATA IN IT.
THIS DATA FILE HAS DATA WHICH IS ALSO IN COLUMNS.
YOU MUST NOW INDICATE HOW THE "COLUMN STRUCTURE"
OF THIS DATA FILE WILL MATCH UP TO YOUR DATA FILE.

YOUR BAR OR SEGMENT TITLE (UP TO 6 CHARACTERS) CAN BE
TAKEN FROM ANYWHERE IN THIS FILE (AND LATER BLANKED).
WHAT IS THE BEGINNING COLUMN OF THE TITLE? 74

AND THE ENDING COLUMN? 76

We now let the system know that our data (for bars # 1, 2, 3, and 4) will be taken from columns 1, 2, 3, and 5 respectively:

FOR COLUMN 1 OF YOUR DATA FILE, ENTER THE COLUMN
NUMBER OF THIS DATA FILE WHICH WILL CORRESPOND: 1

FOR COLUMN 2 OF YOUR DATA FILE, ENTER THE COLUMN
NUMBER OF THIS DATA FILE WHICH WILL CORRESPOND: 2

FOR COLUMN 3 OF YOUR DATA FILE, ENTER THE COLUMN
NUMBER OF THIS DATA FILE WHICH WILL CORRESPOND: 3

FOR COLUMN 4 OF YOUR DATA FILE, ENTER THE COLUMN
NUMBER OF THIS DATA FILE WHICH WILL CORRESPOND: 5

Here we will examine every record as it is presented and choose only the first five.

YOU MAY PROCESS THIS DATA FILE IN ONE OF THREE WAYS TO
EXTRACT YOUR DATA.

A. PREVIEW EVERY RECORD BEFORE PROCESSING (ENTER "0")
B. PROCESS ALL REMAINING RECORDS IN THIS FILE (ENTER "9999")
C. PROCESS ONLY A CERTAIN NUMBER OF RECORDS (ENTER THAT NUMBER)

RESPOND: A

*** NEXT RECORD ***
236 341 213 298 305
JUNE 1-
ENTER "Y" TO PROCESS, "N" TO NOT PROCESS, "S" TO STOP PROCESSING THIS WAY:N

```
*** NEXT RECORD ***
   247          347          236          259          277
   JUNE  2
ENTER "Y" TO PROCESS, "N" TO NOT PROCESS, "S" TO STOP PROCESSING THIS WAY:N

*** NEXT RECORD ***
   245          365          209          227          323
   JUNE  3
ENTER "Y" TO PROCESS, "N" TO NOT PROCESS, "S" TO STOP PROCESSING THIS WAY:N

*** NEXT RECORD ***
   231          349          218          240          276
   JUNE  4
ENTER "Y" TO PROCESS, "N" TO NOT PROCESS, "S" TO STOP PROCESSING THIS WAY:N

*** NEXT RECORD ***
   215          340          235          283          307
   JUNE  5
ENTER "Y" TO PROCESS, "N" TO NOT PROCESS, "S" TO STOP PROCESSING THIS WAY:N

*** NEXT RECORD ***
   207          360          206          304          302
   JUNE  6
ENTER "Y" TO PROCESS, "N" TO NOT PROCESS, "S" TO STOP PROCESSING THIS WAY:N

*** NEXT RECORD ***
   213          358          223          279          279
   JUNE  7
ENTER "Y" TO PROCESS, "N" TO NOT PROCESS, "S" TO STOP PROCESSING THIS WAY:Y

    1:  7
    1:    213.00000     358.00000     223.00000     279.00000
*** NEXT RECORD ***
   229          339          232          237          323
   JUNE  8
ENTER "Y" TO PROCESS, "N" TO NOT PROCESS, "S" TO STOP PROCESSING THIS WAY:Y

    2:  8
    2:    229.00000     339.00000     232.00000     323.00000
*** NEXT RECORD ***
   244          352          204          228          274
   JUNE  9
ENTER "Y" TO PROCESS, "N" TO NOT PROCESS, "S" TO STOP PROCESSING THIS WAY:Y

    3:  9
    3:    244.00000     352.00000     204.00000     274.00000
*** NEXT RECORD ***
   248          364          228          262          310
   JUNE 10
ENTER "Y" TO PROCESS, "N" TO NOT PROCESS, "S" TO STOP PROCESSING THIS WAY:Y

    4:  10
    4:    248.00000     364.00000     228.00000     310.00000
*** NEXT RECORD ***
```

```
   237          344          227          300          300
   JUNE 11
ENTER "Y" TO PROCESS, "N" TO NOT PROCESS, "S" TO STOP PROCESSING THIS WAY:Y

    5: 11
    5:     237.00000     344.00000     227.00000     300.00000
*** NEXT RECORD ***
   221          343          204          296          281
   JUNE 12
ENTER "Y" TO PROCESS, "N" TO NOT PROCESS, "S" TO STOP PROCESSING THIS WAY:Y

    6: 12
    6:     221.00000     343.00000     204.00000     281.00000
*** NEXT RECORD ***
   208          363          232          255          322
   JUNE 13
ENTER "Y" TO PROCESS, "N" TO NOT PROCESS, "S" TO STOP PROCESSING THIS WAY:S

ENTER "Y" TO PROCESS MORE DATA FROM THE FILE,
      "B" TO GO BACK TO BEGINNING OF THIS FILE,
      "N" TO EDIT THIS DATA NOW:N
```

After saying "N" to further extraction, we automatically fall into the "EDIT MODE":

Let's list the file before saving it, just to be sure:

```
**************************************
**   EDITING EXISTING GRAPH DATA    **
**************************************

THIS OPTION ALLOWS YOU TO EDIT GRAPH OR CHART DATA THAT
HAS ALREADY BEEN CREATED.  YOU EDIT THE DATA BY REFERRING
TO THE "LINE NUMBER" OF THE DATA.  THIS IS LISTED AT THE
FAR LEFT COLUMN OF THE SCREEN.  YOU MAY DELETE THAT LINE,
INSERT ANOTHER LINE JUST BEFORE THAT LINE, OR REPLACE THAT
LINE. ALL DATA IS ENTERED "FREE FORM" - THAT IS, SEPARATED
BY COMMAS OR SPACES. ENTER "D" TO DELETE, "I" TO INSERT,
OR "R" TO REPLACE DATA, FOLLOWED BY THE LINE NUMBER, E.G.

    D  1234

WHICH WILL DELETE THE LINE NUMBERED "1234". OTHER OPTIONS:
"L" (line #) TO LIST THE DATA, "P" (line #) TO PRINT THE
DATA, "A" TO AUTO-INSERT DATA. NO NUMBER AFTER THE OPTION
STARTS THE PROCESS WITH THE TITLE INFORMATION. WHEN FINISHED,
  NULL ENTRY (RETURN KEY ONLY) WILL SAVE THE (NEW) DATA.
ANY "OPTION ERROR" WILL RE-DISPLAY THIS EXPLANATION.
ACTION:LIST

DISK FILE NAME: :TITANIUM.002, HAS   36 VALUES,     6 LINES:BAR CHART : COMPARE
TITLE:TITANIUM, ANSLOVILLE
```

NOTE:SHIPMENT STUDY, ANSLOVILLE, FOR DAYS 7-11 LINES 1,2,3 AND 5

1:	7	213.00000	358.00000	223.00000	279.00000
2:	8	229.00000	339.00000	232.00000	323.00000
3:	9	244.00000	352.00000	204.00000	274.00000
4:	10	248.00000	364.00000	228.00000	310.00000
5:	11	237.00000	344.00000	227.00000	300.00000
6:	12	221.00000	343.00000	204.00000	281.00000

ACTION:_

ENTER "SAVE" TO SAVE FILE, OR ANOTHER ACTION REQUEST:SAVE

Now let's look at the chart:

(SEE CHART AT END OF SECTION)

Now for one more time, just to see two differences:

1."BLANKING" out the data area (we will use the 6th column - the day - for our "X")

2. Reading an entire group of data (30 days worth) from a file for graph data editing.

****
** MAKING A NEW GRAPH/CHART DATA FILE **
****

THIS OPTION ALLOWS YOU TO MAKE A NEW DATA FILE, ON YOUR DISK, AND FILL THIS FILE WITH DATA FROM ANOTHER DISK FILE OR FROM YOUR KEYBOARD - BY ENTERING IT MANUALLY. YOU MUST FIRST ENTER A FEW BASIC ITEMS REQUIRED FOR ALL DATA FILES.

WHAT WILL BE THE NAME OF YOUR DATA FILE ON DISK? TITANIUM.003

NOW ENTER A TITLE FOR YOUR DATA FILE (YOU CAN USE UP TO 20 CHARACTERS): TITANIUM, ANSLOVILLE

NOW ENTER ANY "NOTES" THAT YOU WISH TO INCLUDE IN THIS FILE AND WHICH MAY BE HELPFUL TO YOU IN THE FUTURE.
INDIVIDUAL DATA FOR SHIPMENTS OF TITANIUM FOR LINE 4

NOW ENTER THE TYPE OF DATA YOU WILL BE ENTERING.
"1" FOR X - Y DATA.
"2" FOR CHART DATA (PIE OR BAR WITHOUT COMPARISON BARS.
"3" FOR BAR CHART DATA WITH 2 TO 9 BARS BEING COMPARED.
TYPE: 1

THE DATA MUST NOW BE ENTERED. THIS CAN HAPPEN IN TWO
DIFFERENT WAYS. YOU CAN ENTER IT DIRECTLY, OR YOU CAN
EXTRACT IT FROM A DISK FILE, THEN GO INTO THE "EDIT"
OPTION AND MAKE ANY CHANGES NECESSARY.

"1" TO ENTER IT MANUALLY.
"2" TO EXTRACT DATA FROM DISK FIRST.

METHOD: 2

ENTER THE NAME OF THE DISK FILE FROM WHICH THE DATA
WILL BE EXTRACTED: TITANIUM.DAT

YOU MAY "SKIP" SOME RECORDS AT THIS POINT. HOW MANY? 6

**SKIPPED: TITANIUM SHIPMENTS PAGE 17
**SKIPPED:
**SKIPPED: Shipping Depot : ANSLOVILLE WAREHOUSE
**SKIPPED:
**SKIPPED: LINE # 1 LINE # 2 LINE # 3 LINE # 4 LINE # 5
DAY SAMPLE
**SKIPPED:
NOW THE DATA WILL BE EXTRACTED. IT IS EXPECTED TO BE
IN "COLUMNS" IN THE DISK FILE. THESE ARE COLUMNS OF
OF NUMBERS THAT ARE SEPARATED BY SPACES OR COMMAS.
YOU MAY ALSO IGNORE A POSITION OR GROUP OF POSITIONS
AS YOU EXTRACT THE DATA. THIS MUST BE SPECIFIED AS
A BEGINNING CHARACTER POSITION, THROUGH AN ENDING
CHARACTER POSITION (IN ORDER TO DISCARD "TRASH")

BEGINNING POSITION (OR NOTHING, IF NOT DESIRED): 70

ENDING POSITION : 73

DATA WILL BE CONSIDERED AS "BLANK" FROM 70 TO 73
NEXT BEGINNING POSITION (OR NOTHING TO STOP) :

YOUR DATA FILE WILL HAVE 2 COLUMNS OF DATA IN IT.
THIS DATA FILE HAS DATA WHICH IS ALSO IN COLUMNS.
YOU MUST NOW INDICATE HOW THE "COLUMN STRUCTURE"
OF THIS DATA FILE WILL MATCH UP TO YOUR DATA FILE.

FOR COLUMN 1 OF YOUR DATA FILE, ENTER THE COLUMN
NUMBER OF THIS DATA FILE WHICH WILL CORRESPOND: 6

FOR COLUMN 2 OF YOUR DATA FILE, ENTER THE COLUMN
NUMBER OF THIS DATA FILE WHICH WILL CORRESPOND: 4

YOU MAY PROCESS THIS DATA FILE IN ONE OF THREE WAYS TO
EXTRACT YOUR DATA.

A. PREVIEW EVERY RECORD BEFORE PROCESSING (ENTER "0")
B. PROCESS ALL REMAINING RECORDS IN THIS FILE (ENTER "9999")
C. PROCESS ONLY A CERTAIN NUMBER OF RECORDS (ENTER THAT NUMBER)

RESPOND: 30

1:	1.00000	298.00000
2:	2.00000	259.00000
3:	3.00000	227.00000
4:	4.00000	240.00000
5:	5.00000	283.00000
6:	6.00000	304.00000
7:	7.00000	279.00000
8:	8.00000	237.00000
9:	9.00000	228.00000
10:	10.00000	262.00000
11:	11.00000	300.00000
12:	12.00000	296.00000
13:	13.00000	255.00000
14:	14.00000	226.00000
15:	15.00000	243.00000
16:	16.00000	286.00000
17:	17.00000	304.00000
18:	18.00000	276.00000
19:	19.00000	235.00000
20:	20.00000	229.00000
21:	21.00000	266.00000
22:	22.00000	301.00000
23:	23.00000	294.00000
24:	24.00000	252.00000
25:	25.00000	225.00000
26:	26.00000	246.00000
27:	27.00000	288.00000
28:	28.00000	303.00000
29:	29.00000	273.00000
30:	30.00000	233.00000

ENTER "Y" TO PROCESS MORE DATA FROM THE FILE,
"B" TO GO BACK TO BEGINNING OF THIS FILE,
"N" TO EDIT THIS DATA NOW:N

```
**************************************
**   EDITING EXISTING GRAPH DATA    **
**************************************
```

THIS OPTION ALLOWS YOU TO EDIT GRAPH OR CHART DATA THAT HAS ALREADY BEEN CREATED. YOU EDIT THE DATA BY REFERRING TO THE "LINE NUMBER" OF THE DATA. THIS IS LISTED AT THE FAR LEFT COLUMN OF THE SCREEN. YOU MAY DELETE THAT LINE, INSERT ANOTHER LINE JUST BEFORE THAT LINE, OR REPLACE THAT LINE. ALL DATA IS ENTERED "FREE FORM" - THAT IS, SEPARATED BY COMMAS OR SPACES. ENTER "D" TO DELETE, "I" TO INSERT, OR "R" TO REPLACE DATA, FOLLOWED BY THE LINE NUMBER, E.G.

```
    D  1234
```

WHICH WILL DELETE THE LINE NUMBERED "1234". OTHER OPTIONS: "L" (line #) TO LIST THE DATA, "P" (line #) TO PRINT THE

DATA, "A" TO AUTO-INSERT DATA. NO NUMBER AFTER THE OPTION
STARTS THE PROCESS WITH THE TITLE INFORMATION. WHEN FINISHED,
A NULL ENTRY (RETURN KEY ONLY) WILL SAVE THE (NEW) DATA.
ANY "OPTION ERROR" WILL RE-DISPLAY THIS EXPLANATION.
ACTION:_

ENTER "SAVE" TO SAVE FILE, OR ANOTHER ACTION REQUEST:SAVE

MAKING LINE GRAPHS

In order to make line graphs, the system presents a menu which can be filled out as needed. Only one item is needed. That is the title of a data file for the X and Y data values. It must have previously been created by the GRAPHIT system. You may enter up to 10 data files.

All other information is for the sake of making a more professional appearance with the graph. The following items are available for these purposes:

1. Three title lines across the top of the graph.

2. Five sub-title lines which may be placed in one of the four corners of the graph.

3. Titles for the X and Y axis.

Each of these items may be entered or changed by first entering the option number associated with it and then responding to the question asked by PLOTWARE-2.

LINEAR and LOGARITHMIC plots.

The system will graph both linear and log plots, in any combination. The type of plot is established during the AXIS option, while the axis title is being specified.

GRIDS on the plots.

GRIDS may be placed on any plot in a variety of ways. The grid is always associated with the axis with which it is perpendicular, that is, the X axis grid will have vertical lines and the Y axis grid will have horizontal lines. The specifications about the vertical lines forming our grid will be established when you select the X axis options. The Y axis grid will be specified in the same manner.

Each grid set must have specification for the kind of lines desired. If the line is on either end of the corresponding axis, the line will be solid (not dashed). The other lines may be dashed if so desired. In addition to having dashed lines, a solid line may be established at regular intervals. If you

choose a dashed line, you must also specify the distance between dashes (which is also the actual length of one dash).

SCALING your data.

The data presented to the X-Y graph routine does not have to reflect any special units. It must be consistent in some form, of course, to be validly displayed, although PLOTWARE-2 will graph trash just as readily as it will graph your treasures. The system will SCALE your data to an appropriate set of values for the axis. It will always come up with a multiple of 10 or 2 or 5 or something similar (or this value times a power of ten for large or small numbers) and display the data properly. If a power of ten is required in the scaling, the power of ten used will be added to your axis title in order to reflect the corresponding axis annotation used.

For log scaling, a similar process occurs. The axis title will not reflect the units displayed. Instead, the units will be drawn between the annotations on the tic marks and the axis title, slightly larger than the annotaton letters and numbers, yet slightly smaller than the axis title.

POSITIONING THE SUB-TITLE LINES

The sub-title lines, as specified by option item 4 thru 8, may be placed at any one of four different locations on the graph. The location will probably be chosen based on the data which you display. The points at which this data may be placed are labeled on the small representation of the graph, on the menu, as "A", "B", "C", and "D".

SAVING THE GRAPH

The graph may be saved for later use in "composing", using electronic "cut and paste" techniques, or for later display and possible enhancement over that which has already been entered. This option is selected by the "SAVE" menu entry. The system will first ask for a file name to be used for saving the data. If you are going to use it for composing, be sure to use ".CMD" as the extension name, unless you plan to re-name it later. If you save it for a later plotting session, use any name you want.

The "composing" version will be an ASCII file that can be readily editted. It will NOT have the first required command file entry (a PLOTS statement) or the last entry (an EXIT statement). This is to facilitate the merging of several of these files, if so desired. This option is selected by an "A" entry when requested by the computer.

The version "saved for later plotting sessions" is a binary file whic holds certain data values and program overlay information which allows the GRAPHIT program to immediately begin right where you left off. This option is selected by a "P" entry when requested by the computer. The process of re-starting it is as

follows:

GRAPHIT filename.ext

where filename.ext is the file name you choose when saving it.

MAKING A GRAPH

The X-Y line graph will be made on the appropriate device when this option is selected. Refer back to the chart.

An example of using GRAPHIT to build an X-Y line graph:

```
A>GRAPHIT

        ****************************
        **   GRAPH  GENERATION    **
        ****************************

ALL FUNCTIONS SHOW GRAPH AND OPTIONS ON SCREEN

SELECTION     FUNCTION
 NUMBER       TO BE DONE

   1         X-Y LINE GRAPHS
   2         BAR CHARTS
   3         CIRCLE CHARTS (PIE CHARTS)
   4         EDIT GRAPH DATA
   5         BUILD GRAPH DATA FILE
   6         STOP THE PROCESS AND EXIT

SELECT A FUNCTION (BY NUMBER) : 1
```

We select option 1 and get the X-Y line graph menu:

```
ONLY ONE "GRAPH DATA" NAME IS REQUIRED, OTHER ITEMS ARE OPTIONAL.
                <1>                                      "GRAPH DATA"
                <2>                                       FILE  NAME
                <3>                                         TO USE
<11>  ! <A>                              <D>
      !  <4>                                      :          .
      !  <5>                                      :          .
      !  <6>                                      :          .
      !  <7>                                      :          .
      !  <8>                                      :          .
      ! <B>                              <C>      :          .
      !_____________________________________      :          .
                      <10>                        :          .
   <BLANK> SPECIFY "GRAPH DATA" FILE NAME         :          .
   <1 - 3> CENTERED GRAPH TITLE                   :          .
   <4 - 8> "CORNER NOTATION" ON THE GRAPH         :          .
   <9>     PLACE <4-8> AT <A>,<B>,<C>,<D>         :          .
```

REVISION 1.4.....(6-83)

```
   <10>    "X" TITLE, TYPE, GRID:
   <11>    "Y" TITLE, TYPE, GRID:
   <12>    MAKE THE GRAPH NOW.                "CORNER NOTATION" NOW AT : A
   <13>    SAVE THE GRAPH (BY NAME).          "X" AXIS IS LINEAR
   <14>    EXIT - ABANDON THIS GRAPH          "Y" AXIS IS LINEAR
```

We wish to graph the data file "TEST.DAT" so we press return:

```
 MAKE A SELECTION (BLANK, OR 1 THRU 14):_
```

```
ENTER THE NAME FOR THE GRAPH DATA FILE (LINE # 1)
TEST.DAT
```

```
ONLY ONE "GRAPH DATA" NAME IS REQUIRED, OTHER ITEMS ARE OPTIONAL.
            <1>                                        "GRAPH DATA"
            <2>                                         FILE  NAME
            <3>                                           TO USE
<11>  ! <A>                          <D>
      !  <4>                                          :TEST    .DAT
      !  <5>                                          :        .
      !  <6>                                          :        .
      !  <7>                                          :        .
      !  <8>                                          :        .
      ! <B>                          <C>              :        .
      !_____________________________                 :        .
                    <10>                             :        .
   <BLANK> SPECIFY "GRAPH DATA" FILE NAME            :        .
   <1 - 3> CENTERED GRAPH TITLE                      :        .
   <4 - 8> "CORNER NOTATION" ON THE GRAPH            :        .
   <9>     PLACE <4-8> AT <A>,<B>,<C>,<D>            :        .
   <10>    "X" TITLE, TYPE, GRID:
   <11>    "Y" TITLE, TYPE, GRID:
   <12>    MAKE THE GRAPH NOW.                "CORNER NOTATION" NOW AT : A
   <13>    SAVE THE GRAPH (BY NAME).          "X" AXIS IS LINEAR
   <14>    EXIT - ABANDON THIS GRAPH          "Y" AXIS IS LINEAR

 MAKE A SELECTION (BLANK, OR 1 THRU 14):11
```

Notice that the file name is now present, after having been verified as present on disk by the system. There is still room for nine more data files to be plotted if we so choose:

We select option 11, in order to specify a title for the Y axis. We also must tell what kind of Y data we have, so we indicate that it is linear. We also indicate that no grid is desired on this plot.

```
ENTER THE TITLE FOR THE Y AXIS
Y AXIS DATA
```

```
ENTER (FOR THIS AXIS ONLY) :
  1   TO PROVIDE LOGARITHMIC DATA DISPLAY
  2   TO PROVIDE LINEAR DATA DISPLAY (WITH GRID LINES)
  3   TO PROVIDE LOGARITHMIC DATA DISPLAY (WITH GRID LINES)
  <RETURN KEY> TO PROVIDE LINEAR (STANDARD) DATA DISPLAY :_
```

Notice that the menu reflects our choices up to now. We could continue to add more and more to the graph if we desire.

```
ONLY ONE "GRAPH DATA" NAME IS REQUIRED, OTHER ITEMS ARE OPTIONAL.
            <1>                                          "GRAPH DATA"
            <2>                                           FILE  NAME
            <3>                                             TO USE
<11>  ! <A>                              <D>
      !  <4>                                             :TEST    .DAT
      !  <5>                                             :        .
      !  <6>                                             :        .
      !  <7>                                             :        .
      !  <8>                                             :        .
      ! <B>                              <C>             :        .
      !_____________________________________             :        .
                      <10>                               :        .
   <BLANK> SPECIFY "GRAPH DATA" FILE NAME                :        .
   <1 - 3> CENTERED GRAPH TITLE                          :        .
   <4 - 8> "CORNER NOTATION" ON THE GRAPH                :        .
   <9>     PLACE <4-8> AT <A>,<B>,<C>,<D>                :        .
   <10>    "X" TITLE, TYPE, GRID:
   <11>    "Y" TITLE, TYPE, GRID: Y AXIS DATA
   <12>    MAKE THE GRAPH NOW.              "CORNER NOTATION" NOW AT : A
   <13>    SAVE THE GRAPH (BY NAME).        "X" AXIS IS LINEAR
   <14>    EXIT - ABANDON THIS GRAPH        "Y" AXIS IS LINEAR

 MAKE A SELECTION (BLANK, OR 1 THRU 14):12
```

Let's now make a graph:

```
ON WHAT DEVICE IS THIS TO BE DRAWN?
 0 : DOT MATRIX PRINTER "TYPE A" (EG. MX80 DENSE MODE)
 1 : DOT MATRIX PRINTER "TYPE B" (EG. MX80 FAST MODE)
 2 : DOT MATRIX PRINTER "TYPE C" (EG.MX100 DENSE MODE)
 3 : DOT MATRIX PRINTER "TYPE D" (EG.MX100 FAST MODE)
 4 : PEN PLOTTER
 5 : GRAPHICS CRT
 6 : WORD PROCESSING PRINTER
 7 - ? : SPECIAL DEVICE.          ENTER SELECTION:2
```

We choose item "2" as our output device (C.ITOH 8510 is used here):

And we get this plot:

(SEE CHART AT END OF SECTION)

BUILDING A BAR CHART

A tutorial of this is not needed as it proceeds exactly as did the X-Y line graph. The following points are necessary, however, as they reflect differences or items of importance.

BARS:

a. Bars may be placed vertically or horizontally.

b. Bars may be "STACKED" one on top of another. This is done by using two values on your input data file. After selecting the "STACKED" option, the 1st value is the total value of the bar, the 2nd is the amount of the smaller (sub-bar). If the 2nd value is ZERO, no shading is done on the bar. USE THIS OPTION FOR INSTANCES WHERE YOU DO NOT DESIRE SHADING.

c. Bars may be multiply stacked. Here you first choose "simple" bar charts. Then for each STACKED bar, the next data record MUST have a "-" (hyphen) as its ONLY title. The value on this data record is the TOTAL height of the bar at that time. Continue as needed. For instance, we need 4 bars, each stacked 3 high:

title	amount
ITEM 1	8
-	10
-	13
ITEM 2	6
-	12
-	15
ITEM 3	9
-	10
-	11
ITEM 4	-4
-	-9
-	-11

Note that this will appear as 12 bars, but will actually be graphed as 4 bars, each with 3 elements.

d. Bars may have negative values. When this happens, the reference axis is "floated" away from its annotation and title to allow room for the negative values to be plotted.

e. Bars may be entirely negative. As in "d", the bar is "floated" to the highest (or furthest right - depending on whether horizontal or vertical bars are used) point possible.

BUILDING PIE CHARTS

A tutorial of this is not needed as it proceeds exactly as did the X-Y line graph. The following points are necessary, however, as they reflect differences or items of importance.

SEGMENTS:

a. To color a segment select the COLOR option. Use only the color available to your devices.

b. To "explode" a segment (remove it slightly from the pie): Use a negative color (use -1 for black on white, as with a dot matrix printer).

SEGMENT ANNOTATION

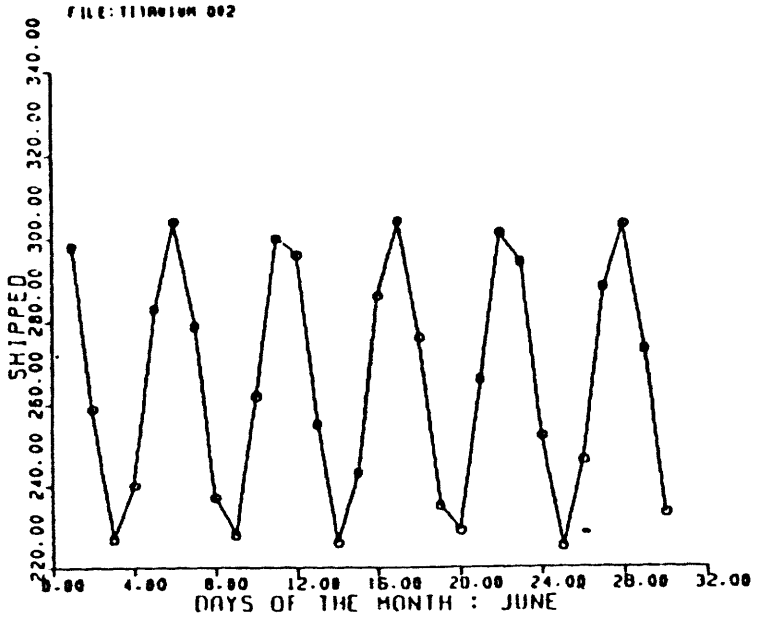
Annotation is centered in the segment, unless the segment is too small for this.

Annotation outside of the segment is also angled to the center of the segment and is justified (either right or left) to the segment depending on the "direction" of the printing.

DATA VALUES

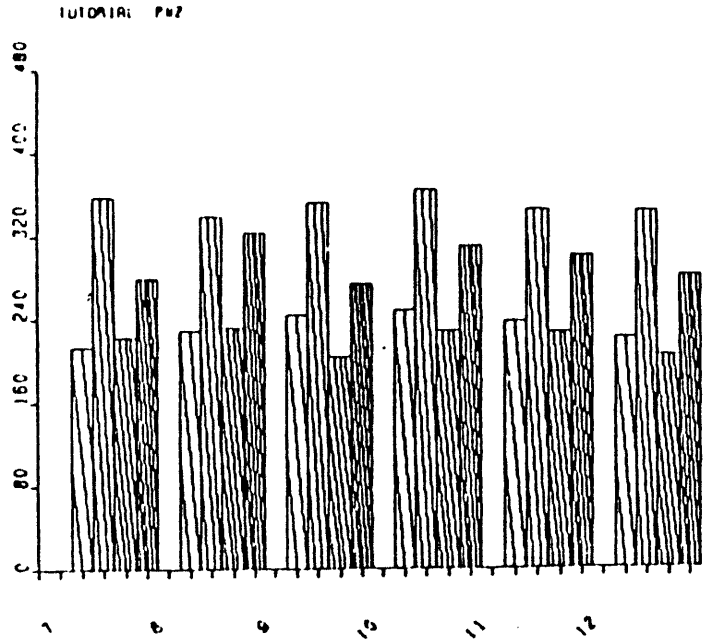
DATA VALUES may be any reasonable value. They will automatically be scaled to percentages.

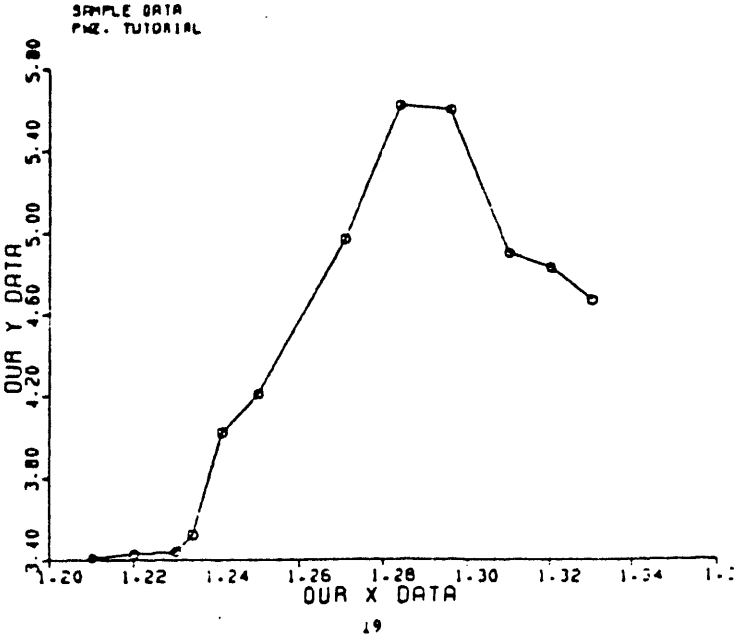
A SAMPLE



TESTFILE.DAT IS NOW GRAPHED

TEST BAR CHART





EXPANDING THE MENUS

In using the DRAW command, one may soon desire to add new features to the menus which he finds in the DRAW module of the CHART processing program. This is always the case when one finds that certain "nice items" like logos, special sizes, special combinations of graphs, etc. will make his graphs assume a very "customized" appearance. This is usually approached by starting a major programming effort and spending many months of intense efforts building menus, file processing subroutines, and all of the other typical things needed by a major system.

The DRAW command is not implemented in that fashion, however. It instead will access a set of instructions which have been specified in a very simple language that is a subset of the standard BASIC language all computer users learn in the beginning of their experience with micro-computers. This particular language is so simple that a person who is not familiar with BASIC will be able to make new menus after a couple of hours of study in this section of the manual.

BASIC does not have a convenient way to "format" data into an orderly appearance on the screen or in the disk files. This is due mostly to the purpose of the original authors of BASIC. It would be a simple language for the SCIENTIST and ENGINEER who would usually scoff at frills in output appearance and who, instead, wanted his answers quickly. Consequently, this BASIC implemented as the PLOTWARE-z language will use an extension to allow the proper formatting of the data to be passed to the PLOTWARE-z processing program.

The subset of BASIC chosen for this purpose is called DATAWARE-z and will be referred to as this for the remainder of this manual. DATAWARE-z functions in the following manner:

1. A DATAWARE-z file consisting of the BASIC statements is generated by some convenient word processing program or test editor just as any other text file would be generated. This file must be called : "DATAWARE.SRC".

2. The INSTALL program is invoked and the "DATAWARE-z COMPILER" is selected from the menu. This allows you to get a listing of the program along with any warning or error messages which may have been produced.

3. The DATAWARE-z COMPILER will generate a code file called "DATAWARE.DAT". This file must be on the default disk when the DRAW verb is invoked. It will then be accessed to present the included menus to the operator of PLOTWARE-z.

The file DATAWARE.SRC, as used in the distribution disk, may be obtained from the INSTALL program as a selection from its menu.

SYNTAX OF DATAWARE-z

VERBS AVAILABLE IN DATAWARE-z:

ACCEPT - Get an item from a screen structure.
ASC - Convert a numeric item to a character item.
CLOSE - Used to terminate access of a disk file or data base.
ENTRY - Marks the location the DATAWARE-z program begins.
FIELD - Used to define a data item.
FOR n = m TO mm - Used to define and control loops.
GET - Used to access a data base item.
GOPROG - Initiates separate program - with return privelege.
GOSUB - Transfers control to external subroutine.
GOTO - Used to transfer control to a new location (label)
IF - Used to selectively transter control to a label.
INPUT - Get data from the operator console.
INPUT#n - Get data from a file or another user partition.
NEXT - Used to terminate a loop and/or continue its action.
OPEN - Used to begin access of a disk file or data base.
PRINT - Used to display items on the operator console.
PRINT#n - Used to display items into disk files or partitions.
PUT - Used to insert an item into a data base.
RETURN - Transfers control back to the host program.
SCREEN - Defines a screen to be used for data entry.
SELECT - Repositions data base pointer (as applicable).
VAL - Converts character data item to numeric.

Other syntax items:

assignment statement:

VARIABLE = STATEMENT

label:

nnn

PROGRAM STRUCTURE:

The DATAWARE-z program is different from BASIC programs because it must always specify any fields which are used in the program. This is in keeping with the stated purpose of making FORMATTING of the data simpler. All "variables" used in the program must therefore be declared by a FIELD statement, (which may occur anywhere in the program), by a SCREEN statement and its accompanying variable declarations (at the beginning of a DATAWARE-z program) or by implicit reference to a data base item through the OPEN statement.

The DATAWARE-z program cannot have multiple statements on a single line, separated by the ":", as allowed by some BASIC interpreters, but it can have multiple statements on separate lines. This means that it is NOT NECESSARY that each line have a statement number. In fact, it is recommended that only the lines be numbered which are specifically referenced by a IF or a GOTO

statement, as this speeds the execution and as only 255 labels are allowed in a program.

Arithmetic statements are allowed only in ASSIGNMENT statements and may not be used in other ways. If an arithmetic statement is needed, in an IF statement for example, the assignment statement should precede the other statement to perform the arithmetic procedure. The resulting quantity can then be used as a "variable".

Variables are specifically given a type by their respective defining verbs. String variables are specifically given a length and that length is always constant. Variables may not be REDEFINED by the encounter of another conflicting statement.

All other differences are seen as one of the two following items:

a. Since DATAWARE-z is a subset of BASIC, some extensions of some of the verbs may be omitted.

b. DATAWARE-z has specific verbs associated with external multiple tasks, data base structures, and CRT screen concepts. Therefore certain verbs are found in DATAWARE-z that are not in BASIC.

SYNTAX OF THE DATAWARE-z VERBS:

ACCEPT

Syntax:

ACCEPT variable

This causes the cursor to position itself at the screen position defined by the SCREEN statement(s) and begin to assemble the item referenced by the variable name. The variable name must be specified by the SCREEN statement as one of the variables of the screen.

EXAMPLE : ACCEPT NAME01

ASC

Syntax;

ASC variable

This effects the conversion of a numeric item (specified by the variable) into a character (string) item. This is normally used to transfer numeric quantities, which would normally be unreadable in a "standard ASCII" data format into a format of that type. It affects the TOGGLE variable in the following way:

TOGGLE = 0 when a normal conversion occurs.
TOGGLE= 1 when a conversion occurred with overflow ofthe

receiving field.

EXAMPLE : DATE01 = ASC (DATEX)

CLOSE

Syntax:

CLOSE n (standard file format)

CLOSE name (data base format)

This closes an input disk file. It closes and saves an output disk file, with any updates noted in the directory. It must specifically be executed for any output disk file which you wish to save for other uses. It terminates usage of a data base file and allows other tasks and/or programs to access the updated file structures (as applicable).

EXAMPLES: CLOSE 2 CLOSE "MYDATA.DBS"

ENTRY

Syntax:

ENTRY n

DATAWARE-z functions as a "subroutine" to other software structures. The normal use of DATAWARE-z allows the program to "call" the dataware structure and to direct it to begin at a specific point. DATAWARE-z may choose to assemble certain data and then return to the calling program to allow it to process that data. On subsequent times, the host program may then return to DATAWARE-z in the same way, but with the intention of proceeding from a different point. This point is specifically stated by the ENTRY statemnt.

This statement is not normally used unless the complexity of the calling structure and/or the needs of the host program require it.

EXAMPLE: ENTRY 3102

FIELD

Syntax:

FIELD n variable defaultvalue format

Where:

"n" is 0 for the pass/return array to the host program.
"n" is 1-5 for disk files, or n = 6 for the data base.

"variable" is a well-defined variable name, not previously

encountered.

"defaultvalue" is a default value that the variable assumes. This is a convenient way to specify "CONSTANTS".

"format" specifies the beginning location in the respective file or area where the variable begins. It is followed by a dash and another number indicating that the field extends through that position. This is true only on character (string) variables and only the beginning position is needed for integers and reals.

This allows the program to subsequently use that name as a variable, as specified. The following convention is used when the "format" character option is omitted (hence specifying a NUMERIC item):

If the variable begins with I,J,K,L,M, or N it is an INTEGER

Otherwise it is a REAL

Caution should be exercised with the use of integers and reals as most "canned software" accepts only character (string) data. The usual place where Integers and reals are found is in the pass/return array for use by the host program, or for use as "scratch-pad" items, or in disk files where speed and data compaction is necessary. Integers and reals are stored in the standard "mainframe" format of two bytes per integer and four bytes per real.

EXAMPLE : FIELD 2,MYINTG,45,(15)

MYINTG is an integer in file 2, occupying 2 bytes of memory. It begins with a value of 45.

FIELD 3,MYSTRG,"ABCdef123456",(21-32)

MYSTRG is a character (string) variable occupying positions 21 through 32 of the area of file 3. It has an initial value of "ABCdef123456".

PROGRAMMING HINT: Since it is rare to need all five possible files during a DATAWARE-z data entry subroutine, it is usually convenient to select one of the files (for example #5) to use as a "work" or "scratchpad" area. The work variables will then always occupy that file's area.

FOR

Syntax:

FOR n = m TO mm

This allows "loops" in the standard sense of BASIC programming. It sets the variable "n" to the quantity "m" and procedds through

the loop until it encounters a "NEXT" statement. It then increments the quantity "n" by 1 and begins the next trip through the loop if the quantity "n" is not greater than "mm". Hence it always makes one trip through the loop no matter what values "m" and "mm" assume.

The variables "n", "m", and "mm" must be assigned as integers and referenced as such throughout the "program".

EXAMPLE:

```
FIELD 5,N,0,(10)
FIELD 5,M,12,(12)
FIELD 5,MM,20,(14)
  ...
FOR N = M TO MM        (loops from 12 through 20)
```

GET

Syntax:

GET data-base-specification.

This is used to obtain the item specified in the target data base. The section accompanying the data base of your choice must be consulted to reference the proper syntax for this verb.

GOTO

Syntax:

GOTO n

The label "n" must exist in the program and must be in the proper range for a label

EXAMPLE: GOTO 100

IF

Syntax:

IF variable-1 ltest variable-1 THEN n

"variable" is a properly formed variable and both are compatible in type.

"ltest" is one of the following:

=
<
>
<>
<=
>=

"n" is a valid label.

This allows the program to make logical program control transfers.

EXAMPLE: IF A = B THEN 200

INPPUT

Syntax:

```
INPUT                   (input from the console)
INPUT#n                 (from a disk file or user partition)
```

This does not get variables from the designated location. It instead inputs a RECORD from the location, which will, or course, contain the values to be associated with the variables. It then the responsibility of the FIELD statements to get the values to their associated variables. It is suggested that one use the INPUT statement on single variables only as this simplifies the handling of direct console items. If the console is to accept more than one variable at a time, probably following the presentation of a single question to the user, the best way to accomplish this is to use the SCREEN verb to set up a CRT screen.

```
EXAMPLE:        INPUT#2   (gets next record from file 2)
                INPUT#3(IKEY) (RANDOM FORM : record IKEY)
```

NEXT

Syntax:

```
NEXT
```

This is the proper way to end a "loop". It processes the loop currently in effect.

NOTE : There is no provision in DATAWARE-z to handle "nested loops" as there is in BASIC. Care should be taken to avoid these constructs.

OPEN

Syntax:

```
OPEN n                      (data file format)
OPEN "DATABASE.NAM"         (data base format)
```

This attaches a data file to the program for subsequent usage via INPUT#n or PRINT#n.

The syntax for data base usage depends on the data base attached. See the appropriate section for your preferred data base.

The proper action for data files is as follows:

a. For established sequential files, the next references should be INPUT#n only.

b. For new files (output files), the first reference will cause any established file to be removed.

c. For random files, the first access of the file must be an INPUT#n. Thereafter reads and writes may be interspersed.

It should be noted that there is no inherent record limit in DATAWARE-z as there is in BASIC and other systems. If you wish to write or read a 750 character record, you may do so. There is a realistic limit however, as imposed by the total memory on your system. If this is exceeded by abnormally large file sizes, you may possibly experience a system halt with an accompanying message about the resultant lack of memory space.

```
EXAMPLE:    OPEN 3 "B:DATAFILE.DAT"  (OPENS FILE 3 as
                                      DATAFILE.DAT on "B:")
```

PRINT

Syntax:

```
PRINT              (Console format)
PRINT#n            (DISK FILE or DATA BASE format)
```

This allows the display of the designated fields onto the appropriate item. This is the only way to effect a physical transfer of the data which you have specified through the use of your FIELD statements. A record of length equal to that necessary to handle all of your specified fields will be transferred to the designated place. The values of 1-5 are valid for disk files. The value of 6 may be valid for your data base. See the data base section appropriate to your preferred data base format.

```
EXAMPLE:          PRINT#3
```

PUT

Syntax:

```
PUT variable
```

This allows the direct entry of a data base item into the specified location. See the section on your preferred data base.

RETURN

Syntax:

```
RETURN
```

This construct allows the return of control to the host program.

In PLOTWARE-z, it should be used only after the file "COMMON.CMD" and the appropriate data file(s) have been gathered. This is the proper way to begin processing the data gathered by the DATAWARE-z subroutine and the command file built during that process.

SCREEN

Syntax:

```
SCREEN
   (records : one for each line to display on the CRT)
   ...
   FIELD
   ...
```

This allows an easy way to specify a CRT screen to be built. The necessary "help" information and room for each input field (denoted by the underline symbol(s)) precedes, then a FIELD statement for each field follows. There is one major difference between this FIELD statement and the standard FIELD statement. There is no "format" part as each field is necessarily character data (do not attempt to enter binary data on an ASCII terminal!) and the position of input is already defined. The total length of the screen input field is the sum of the separate fields within the SCREEN.

EXAMPLE :

```
SCREEN
Your NAME          ____________________
Your SSN           _________
The DATE           ________
FIELD 0,NAMEIN,"                      "
FIELD 0,SSNIN,"          "
FIELD 0,DATEIN,"  -  -  "
```

This defines a three line screen with three fields. The total record length is the sum lengths of the fields. When displayed on the screen, the first two fields will be blank and the DATE field will have dashes in the proper positions for month, day, and year placement.

SELECT

This construct allows the proper placement of the input-output process for certain data base constructs. It effects a specified record and file selection as necessary. See the section on your preferred data base.

VAL

Syntax:

```
VAL(variable)
```

This is the means to convert character (string) data into numeric format. It sets the TOGGLE variable as follows:

```
TOGGLE = 0      (after successful conversion)
TOGGLE = 1      (after conversion on non-numeric data)
```

This allows the editing of numeric data fields.

```
EXAMPLE:        X = VAL (MYSTRG)
```

This construct, like the ASC construct, is valid only in the assignment statement.

label

There is provision for labels in the program. The label must occur as the first item on that program line, and must be in the range of 1 to 255 (inclusive)

```
EXAMPLE:        40 X = A + B
```

This shows a label used together with an assignment statement.

assignment statement

This allows the inclusion of simple computations. Since DATAWARE-z is designed for data input and not computations, the assignment statement is limited to the following forms:

```
variable1 = variable2 arithop variable3      or
variable1 = function ( variable2)
```

arithop may be : "+","-","*","/",or "**"
The second form is seen with VAL and ASC operations (see above).

USING THE COMMAND FILE SYSTEM - OPERATING DETAILS

When using the command file system, it is assumed that the user is familiar with the following items:

a. The COMMAND FILE language.

b. The INSTALLATION of his HARDWARE.

c. Any items unique to his system (device TYPE, etc.)

If you have not done these, return to the proper sections and do so. This section is designed to provide any specific details necessary to the operation of the system which were not covered in those other sections. However, this section will be complex and vague to those not familiar to some degree with the above items.

OPERATION:

The COMMAND FILE system revolves around three different files. These must always be accessible to the system when running:

a. CHART.COM

b. PLOTSPEC.DAT

c. OVERLAYS.OVR

CHART is the base program which processes the command files. It is started by the operator or by the running system (see the BASIC example in the INTERFACE section for an example of this) and will not be needed again.

Chart is not in the format of the usual PASCAL, COMPILED BASIC, FORTRAN, or similar object code files. It instead consists of many linked assembly language modules which contain many different data and object code areas. These areas may be or may not be modified in the process of running the system. Some of these areas are specifically scrambled to facilitate the security system of PLOTWARE-z. However, the program CHART.COM will not be modified in any way once it is loaded. Therefore, it may be STARTED, by the operator or by any system, from any disk desired. This will allow the user to manage his disk storage facilities in whatever manner is convenient to the occasion. Instead, all modified code and overlay code and data that is modified will be placed into the file PLOTSPEC.DAT and/or PLOTWARE.OVR. Generally speaking, one must never perform input or output to PLOTSPEC.DAT for that file is for the internal usage of PLOTWARE-z routines. The file PLOTWARE.OVR may be used as a link and chain file (see the INTERFACE SECTION) and can be freely modified in the first 8 logical (128 byte) records. The format of the first 12 bytes of the first record is fixed (see the INTERFACE SECTION), and records beyond the first eight records will most likely be modified if they are included into this file. If 1012 bytes of

common disk storage is not enough for your program requirements outside of the COMMAND FILE routines, use another disk file and maintain it in some acceptable fashion.

The CHART program first reads the PLOTSPEC.DAT file and analyses this file with respect to the code file in memory (CHART.COM). If the code file matches that expected from the serialized distribution disk, as expected via a special serial number "hatch algorithm", the program proceeds. The first 262 bytes of memory are always used for this purpose. This is called the "initialize phase". Other things proceed during this phase:

a. The handling of "3 pass" dot matrix color printers.

b. The handling of "n pass" single pen plotters.

c. The establishing of further "serial number hatch and slash" routines. These always occur periodically and will use the first 262 bytes of memory, obtain their code from the OVERLAYS.OVR file and access PLOTSPEC.DAT to find necessary overlay space for long routines.

d. The structure of the OVERLAY SKELETON for the CHART program.

It should be noted that the overlay structure of PLOTWARE-z is not like that used in BASICS and PASCAL compilers found elsewhere. It is specifically designed to be a multi-level overlay scheme and must not be modified by the user. Modification of the .OVR files or of the file PLOTSPEC.DAT will probably result in a "systems crash" and the user may find that the disk directories will be destroyed. LEAVE THESE FILES ALONE! If you are overly concerned about this problem occurring - perhaps by the inadvertent encounter of Murphy's Law while you are debugging some external program (those assembly language programs will be candidates for this sort of thing), feel free to use STAT to make these file READ-ONLY. PLOTWARE-z is sufficiently intelligent to modify the status of the file(s) prior to writing to them in the overlay process.

SUMMARY OF DISK FILES

GRAPHIT.COM (menu driven graphics) - Uses CHART.COM

CRT's, Plotters:

a. GRAPHIT.COM (on default disk)
b. OVERLAYG.OVR (may be placed on any disk - prefer default)
c. CHART.COM (on default disk)
d. OVERLAYS.OVR (may be placed on any disk - prefer default)
e. PLOTSPEC.DAT (on default disk)

DOT MATRIX PRINTERS:

REVISION 1.4.....(6-83)

a. GRAPHIT.COM (on default disk)
b. OVERLAYG.OVR (may be placed on any disk - prefer default)
c. CHART.COM (on default disk)
d. OVERLAYS.OVR (may be placed on any disk - prefer default)
e. PRINTMAP.COM (on default disk)
f. PLOTSPEC.DAT (on default disk)

CHART.COM (for COMMAND FILE plotting without GRAPHIT, or by the USER program as written)

CRT's, Plotters:

a. CHART.COM (on default disk)
b. OVERLAYS.OVR (may be placed on any disk - prefer default)
c. PLOTSPEC.DAT (on default disk)

DOT MATRIX PRINTERS:
a. CHART.COM (on default disk)
b. OVERLAYS.OVR (may be placed on any disk - prefer default)
c. PRINTMAP.COM (on default disk)
d. PLOTSPEC.DAT (on default disk)

PRINTMAP.COM (used atomatically for others - for dot matrix)

Reference files:

a. PLOTWARE.DAT (holds last "picture" from any program) (required)
b. PLOTWARE.OVR (holds return overlay information for auto-restart of calling (chaining) program. Program name in 1st record, pos 2-12.) restarts only if this file is present.

PLOTCOMP.COM (Used to generate binary file of plotter routines.) (This routine is found in the INSTALL section. Use INSTALL to run it from the INSTALL overlay structure - level 3)

1. PLOTSPEC.SRC (the input file to the compiler PLOTCOMP)

2. PLOTSPEC.DAT (the output file from the compiler)

MODIFY.COM

files as indicated in menu : during execution.

This part of the INSTALL program allows the specific direction of certain parameters to the PLOTSPEC.DAT file and to the CHART.COM overlay structure. This is the way one must select specific options for multiple dot matrix printers or multiple dot matrix printer modes.

YOU DO NOT NEED THE PLOTCOMP AND MODIFY SECTIONS PAST INSTALLATION. Remove these once the system is installed.

USING THE COMPILER LIBRARY SYSTEM - OPERATING DETAILS

When using compiler library file system, it is assumed that the user is familiar with the following items:

a. The language to be used as a "host program" language.

b. The INSTALLATION of his HARDWARE.

c. Any items unique to his system (device TYPE, etc.)

d. The "linking process" associated with the host language.

If you have not done these, return to the proper sections or documents from your language vendor and do so. This section is designed to provide any specific details necessary to the operation of the system which were not covered in those other sections. However, this section will be complex and vague to those not familiar to some degree with the above items.

OPERATION:

The COMPILER LIBRARY system revolves around three different files. These must always be accessible to the system when running:

a. The USER PROGRAM.

b. PLOTSPEC.DAT

c. OVERLAYS.OVR

The USER PROGRAM is the base program which processes the graphic command subroutines. It is started by the operator or by some running system which can intelligently select control of the computer to some other program. The necessary code overlays and data overlays will be taken from available memory of the computer after the program performs its first mandatory call to the PLOTWARE-z system (via a call to PLOTS) and the original code file will not be needed again.

The PLOTWARE-z routines which require overlay of program and/or data structures are not in the format of the usual PASCAL, COMPILED BASIC, FORTRAN, or similar object code files. Therefore, it will not interfere with any of these types of overlay calls. It instead consists of many linked assembly language modules which contain many different data and object code areas. These areas may be or may not be modified in the process of running the system. Some of these areas are specifically scrambled to facilitate the security system of PLOTWARE-z. However, the program users program on the disk will not be modified in any way once it is loaded. Therefore, it may be STARTED, by the operator or by any system, from any disk desired. This will allow the user to manage his disk storage facilities in whatever manner is convenient to the occasion. Instead, all modified code and overlay code and data which is modified will be placed into the file

PLOTSPEC.DAT and/or PLOTWARE.OVR. Generally speaking, one must never perform input or output to PLOTSPEC.DAT for that file is for the internal usage of PLOTWARE-z routines. The file PLOTWARE.OVR may be used as a link and chain file (see the INTERFACE SECTION) and can be freely modified in the first 8 logical (128 byte) records. The format of the first 12 bytes of the first record is fixed (see the INTERFACE SECTION), and records beyond the first eight records will most likely be modified if they are included into this file. If 1012 bytes of common disk storage is not enough for your program requirements, use another disk file and maintain it in some acceptable fashion.

The user program will call PLOTS which will first read the PLOTSPEC.DAT file and analyse this file with respect to the code file in memory (linked PLOTWARE-z parts only). If the code file matches that expected from the serialized distribution disk, as expected via a special serial number "hatch algorithm", the program proceeds. The first 262 bytes of memory are always used for this purpose. This is called the "initialize phase". Other things proceed during this phase:

a. The handling of "3 pass" dot matrix color printers.

b. The handling of "n pass" single pen plotters.

c. The establishing of further "serial number hatch and slash" routines. These always occur periodically and will use the first 262 bytes of memory, obtain their code from the OVERLAYS.OVR file and access PLOTSPEC.DAT to find necessary overlay space for long routines.

d. The structure of the OVERLAY SKELETON for the CHART program.

Be sure to always allow this "low memory" area in your user programs. See the INTERFACE SECTION for an example of this in FORTRAN.

It should be noted that the overlay structure of PLOTWARE-z is not like that used in BASICS and PASCAL compilers found elsewhere. It is specifically designed to be a multi-level overlay scheme and must not be modified by the user. Modification of the .OVR files or of the file PLOTSPEC.DAT will probably result in a "systems crash" and the user may find that the disk directories will be destroyed. LEAVE THESE FILES ALONE! If you are overly concerned about this problem occurring - perhaps by the inadvertent encounter of Murphy's Law while you are debugging some external program (those assembly language programs will be candidates for this sort of thing), feel free to use STAT to make these file READ-ONLY. PLOTWARE-z is sufficiently intelligent to modify the status of the file(s) prior to writing to them in the overlay process.

SUMMARY OF DISK FILES - COMPILER LIBRARY "RUN TIME"

CRT"s Plotters:

a. User's program (on any disk)
b. PLOTSPEC.DAT (on default disk)

DOT MATRIX PRINTERS:
c. PRINTMAP.COM (on default disk)
d. PLOTSPEC.DAT (on default disk)

PRINTMAP.COM (used atomatically for others - for dot matrix)

Reference files:

a. PLOTWARE.DAT (holds last "picture" from any program) (required)
b. PLOTWARE.OVR (holds return overlay information for auto-restart of calling (chaining) program. Program name in 1st record, pos 2-12.) restarts only if this file is present.

PLOTCOMP.COM (Used to generate binary file of plotter routines.) (This routine is found in the INSTALL section. Use INSTALL to run it from the INSTALL overlay structure - level 3)

1. PLOTSPEC.SRC (the input file to the compiler PLOTCOMP)

2. PLOTSPEC.DAT (the output file from the compiler)

MODIFY.COM

files as indicated in menu : during execution.

This part of the INSTALL program allows the specific direction of certain parameters to the PLOTSPEC.DAT file and to the CHART.COM overlay structure. This is the way one must select specific options for multiple dot matrix printers or multiple dot matrix printer modes.

YOU DO NOT NEED THE PLOTCOMP AND MODIFY SECTIONS PAST INSTALLATION. Remove these once the system is installed.

In attaching new devices, the user must be aware of the simple language of PLOTCOMP. This is processed by a part of the INSTALL program and will produce the necessary code to drive the plotters, CRT's, cameras, and such which are typically assigned to graphics peripheral service.

Most items do not need assembly language support, but if you do need it or just prefer to use this approach, it is available. The code must reside from 145H and cannot extend past 200H. See the SUBR command and the CALL command.

MODIFYING THE PLOTWARE PARAMETERS

The PLOTWARE-z software can be modified in all of the areas which specify the hardware to be used as graphics output. These areas are held in "COMMON" storage, in most cases and are accessible via the COMMON specification in either M80 assembler, or by the COMMON specification in F80 Fortran, or similar compilers. In the case of any standalone programs which access these variables, a separate program, "MODIFY.COM" (found as structure "1" of the INSTALL program) will change these parameters to the desired values. Some changes are able to be made only by external program means, such as DDT.COM, but these are all associated with such things as replacing an assembler driver routine and are not generally used. Other modifications are possible through the INSTALL program.

The explanation of each of these variables follows, with the variables grouped into the following categories:

1. Pen plotter and word processing printer parameters. Includes discussion of the plotter command parameters.

2. Dot Matrix printer parameters for plot generation - as common with the REAL TIME device data areas.

3. Device "driver" addresses - used for placing user - installed routines for other hardware devices.

The parameters are described in the following fashion:

PARAMETER NAME (variable NAME in common)

PARAMETER description.

When using the COMMON specification, it should be noted that all parameters MUST be of the proper type and usage. For this reason, all parameters named in common follow this convention:

All REAL variables are named with a beginning letter of A-H or O-Z.

All 16 bit integers are named with a beginning letter of I-K or

M-N.

All 8 bit integers are named with a beginning letter of L.

When updating the stand-alone programs, only the name used in parentheses need be specified. Usage will be accounted for by the MODIFY program.

PEN PLOTTER and WORD PROCESSING PRINTER parameters.

STEP SIZE OF PLOTTER (STEPS)

The step size of the plotter is the effective distance between each step of the controlling motor drivers. This is usually expressed in "steps per inch" and must be entered into the software in this fashion. If the step size is expressed in a fractional value (e.g. .005) take the inverse of the value. The default step size is 200.0 - the effective steps per inch as given by the Houston Instruments Hiplot series plotters.

THE MULTIPLICATIVE FACTOR FOR X VALUES (XFACT)

This is a value which can be used to adjust the plotter software for any strange step size that might be encountered, for instance - the X step size may be only 1/2 the size of the Y step size. Since there is only one STEP value, the user must "weight" each X value by a factor of 2 in order to avoid a "squashed" appearance in his plot.

THE MULTIPLICATIVE FACTOR FOR Y VALUES (YFACT)

This is a corresponding value for the Y factor of the plot. It functions exactly as XFACT does for X values.

THE MAXIMUM NUMBER OF X STEPS PER PLOT COMMAND (MAXXS)

The software will calculate the best path between the points of the plot based on an algorithm that includes the necessary integer approximation to that path. It must calculate the relative X and Y distances for the move, and must know what is the maximum number of X steps which it can take with one single command to the plotter. In the case of Word Processing printers, this is usually 1 since the printer must then pause to print a period, if the "pen" is down. It may be limited by the electronics of your plotter. It is set to 999 in the standard system, for the "smart" controller of the HIPLOT plotters.

THE MAXIMUM NUMBER OF Y STEPS PER PLOT COMMAND (MAXYS)

This is a corresponding maximum for the Y steps. It functions exactly as does MAXXS.

THE "STEPS / INCH" FOR WORD PROCESSING PLOTTERS (WPXSTP)

The word processing printers must calculate a conversion for each

"plotter move" in terms of their available "step size". The distance that the platen moves the paper in the line feed or reverse line feed mode is called WPXSTP. This is delivered in the standard system as 48.0.

THE "STEPS / INCH" FOR WORD PROCESSING PLOTTERS (WPYSTP)

This is a corresponding value for the movement of the daisy wheel or thimble across the page. It is standard as 60.0.

THE TABLE OF "PSEUDO COMMANDS" (LPTABL)

This is the relative number for the code table for all plotter COMMANDS. It exists in the machine as the file "PLOTSPEC.DAT" and each plotter type must be contained in 512 bytes or less. For example, the HIPLOT series of intelligent plotters is 0, the COMPLOT series is 1, and so on. The relative position is determined by the relative call numbers found in the PLOTS specification. In the standard system, this table contains three sets of parameters - the HIPLOT series (intelligent), the COMPLOT series (DMP 4,5,6), and the QUME word processing printer.

See the explanation at the end of this section for further explanation on the LPTABL and how to write new code tables for it.

DOT MATRIX PRINTER PARAMETERS common with the REAL TIME area.

The dot matrix printers supported by the standard system are all controlled by a set of parameter tables that define the necessary protocol for obtaining graphics on the selected printer(s). Many of these devices have several forms of graphics, some of which are "dense", and gives a good approximation to a pen plotter graphics; and some of which are "fast", giving quicker results since the dots are printed farther apart. The density of the different modes give rise to four sets of parameters for all of the dot matrix commands. If the user is changing the software for another printer which has only one form of graphics display, he need change only one of the types of parameters and specify that corresponding "PRINTER CALL" when he is using the device.

The PRINTER graphics are made in the following way:

1. For the TYPE 1,2 and 3 mode, the print head makes one pass across the page and puts all of the dots into place. No overlap of dots occurs, and the individual dots can be seen.

2. For the TYPE 0 mode, the first pass proceeds at maximum dot density and each adjacent dot overlaps the previous dot by part of a dot. The head then returns and begins the second pass. It first advances just enough to pass the second row of dots between the previously made dots. This lets the dots overlap about one half a dot. This results in far better resolution in this direction.

3. For the TYPE 2 mode, the first pass proceeds as did the

previous mode, however, there is no second pass made by the print head. This is done in order to allow a "medium dense mode". This gives less resolution than is available with the previous mode, yet it is still very good resolution. This mode is especially useful for those tasks which may take longer than you need, yet which still require very good resolution.
The parameters are explained as follows:

THE NUMBER OF STEPS/INCH, PAPER TRAVEL DIRECTION (X1FACT)

This describes the number of "dots / inch" that will be placed onto the paper, in the paper travel direction. This is a function of the distance between the matrix wires in the print head, and can be doubled if the print head can make two passes across the line, offsetting the second pass so as to print the second row of dots between the first row. This value here reflects the density when NO second pass is made. In the standard system, this is 72.0.

THE NUMBER OF STEPS/INCH, PAPER TRAVEL DIRECTION (X2FACT)

This describes the density when TWO print head passes are made. In the standard system, this is 144.0.

THE NUMBER OF STEPS/INCH, "ACROSS" THE PAGE (Y1FACT)

This is a corresponding value for the "across the page" density. It reflects the dots per inch, when the head is traveling at the "normal" rate of speed. It is 60.0 in the standard system.

THE NUMBER OF STEPS / INCH, "ACROSS" THE PAGE (Y2FACT)

This is a corresponding value for the "across the page" density when the print head is traveling at maximum resolution, or is printing (plotting) in the "compressed" mode. It is 120.0 in the standard system.

THE EFFECTIVE WIDTH OF THE PRINT PATH OF THE PRINT HEAD

This number reflects the print width of the print head pass, from its extreme "left" to its extreme "right". It is used to allocate storage for the vector to raster scan conversion and to produce an effective "window" for all plotting to be done on your device. It is 8.0 in the standard system.

THE EFFECTIVE WIDTH OF THE PRINT PATH OF THE PRINT HEAD

This number reflects the print width of the print head pass. It is intentionally set at 15.0 (wider than the actual travel of the wide printers) so that future printers may be included. It is of no real concern, for most printers will throw away the extra values.

THE MAXIMUM LENGTH OF THE PICTURE IN THE X DIRECTION (XLDFLT)

This value reflects the maximum length of the "page" for the picture to be drawn on the dot matrix printer. It is set at 10.0 for the standard system, and is adjustable (dynamically) for each picture, by use of the XYSIZE statement. This value is the DEFAULT value only, and remains dynamically available through the XYSIZE statement after it is installed.

SPECIAL NOTE ON THE "X" AND "Y" SIZE SPECIFICATIONS. It should be noted that these are DEFAULT values only, and are easily changed with the XYSIZE statement. If you are continually running 15 inch by 15 feet - long plots of petroleum log data, or something similar, this option allows you the liberty of specifying that size - which may be "standard" for you - to be the default size of each plot.

ENABLE THE "OPTIMIZATION" MODE OF THE EPSON DRIVER (ISTAT)

This "feature" is for use only on EPSON printers.

There is a feature in the EPSON microware, located in the printers own controller, and hence not readily accessible by the user, which allows data to be occasionally lost if the plot data is presented to the printer at the proper (or in this case, improper) time. The PLOTWARE-z routines have an algorithm which will negate this "feature" and this routine must be used if you are using a buffered serial card in your EPSON. This card has the nasty habit of accepting data from the line and presenting this data to the printer in "spurts" which will most definitely cause this "loss of data" feature to be invoked. If your printer is on either an unbuffered serial card (of which there are several available, and which must be decided upon by the user for his particular unit) or on a parallel port of your computer, you can realize a significant increase in throughput by enabling this option. It must be 0 to be enabled. It is set to 1 (disabled) in the standard system.

(OPTIMIZING MODE ONLY) PRINTER PORT I.D. (LPORT)

This is for EPSON printers only.

This specifies the port of the computer which is used to sense the "busy" or "not busy" status of the printer device. It is delivered as 043H in the standard system (the printer status port of the major I/O processor card of the ENERCOMP TWIN-Z system).

(OPTIMIZING MODE ONLY) PRINTER STATUS MASK (LBIT)

This is for EPSON printers only.

This specifies the bit mask of the status port to use, in order to examine only the bits pertinent to the printer. The standard system is set to 1, meaning bit 0 is the status of the print device itself (not the I/O processor card itself).

(OPTIMIZING MODE ONLY) HI/LO MASK BIT. (LANDZ)

This is for EPSON printers only.

This specifies whether the LBIT mask looks for a "high" or a "low" status mask. If this is the same mask as the LBIT variable, the mask is looking for a "low" or zero result of the mask. If the value is zero (regardless of the value of LBIT), the mask is looking for a "high" or non-zero result of the mask test. It is set to 1 in the standard system (high means "busy", low means "not busy" - ready to accept data).

(OPTIMIZING MODE ONLY) STATUS STABLE TIME (ISWAIT)

This is for EPSON printers only.

This is used to calculate the "stable status" time of the EPSON printer. It is set to 100, which is adequate for a 6 MHZ or 4 MHZ processor (or the 5 MHZ 8085, often referred to as a 10 MHZ processor, by virtue of its external clock). It should be halved, if an 8080 or 2MHZ Z80 processor is used. This is a relatively non-critical value, as it can be "too large" without resulting in any significant degradation of the plotting process.

DEVICE DRIVER ADDRESSES

The remaining parts must be modified by the user, and are available for those printers or plotters which must be attached through an external routine. These are the critical entry points of the PLOTWARE-z system, and are to be used for assembler level (or the like) modifications.

EPSNMAP.COM (vector to raster scan conversion)

THE ENTRY POINT TO THE LIST DRIVER (014EH)

This is the point to which the software branches to output a line of plot data. The register convention is as follows:

HL - points to the byte array to be output.

DE - points to the number of bytes to output. THIS MAY BE ZERO - handle it properly. 16 bit integer.

BC - points to the maximum for this line - may indicate an error condition if less than the 16 bit integer pointed to by DE.

LAST BYTE OF LIST DRIVER ROUTINE (0150H)

Used to limit the user routine from expanding into other core of other routines.

REVISION 1.4.....(6-83)

These are not generally needed by the user, but if it is necessary to write assembly language routines, these give the necessary "hooks" to get to the proper places. DO NOT PLACE CODE OUTSIDE OF THESE BOUNDARIES.

MAXIMUM BIT (print wire) TO ADDRESS (014AH)

Set to 128 for the standard system (bit 7). It can be less, if fewer wires than 8 are calculated at any one pass. For example, this would be 32 (bit 5) for the early PAPER TIGER printer which uses only six wires on its graphic pass.

PEN PLOTTING AND WORD PROCESSING PLOTTING

PLOTTING PRIMITIVE DRIVER ROUTINE

This routine is used to provide the plotting for the system when using the pen plotting routines. It is a standard routine that can interpret the "pseudo code" of the LPTABL array and interface just about any plotting device to the system. This routine should serve any reasonable plotting device which can be encountered. If you choose to write your own routine, you will find the device entry point to be "ZXPLOT", if linking to the system, and the address (in stand-alone programs) will be pointed to by location 0145H. The last address of the routine will be pointed to by 0147H. No register convention is used in the call. The data passed is through the COMMON statement and appears in compiled form at the end of this section.

!!!!!! NOTE OF CAUTION !!!!!!!

There are several "things" built into the plotting software which are designed to protect the proprietary nature of the software. These are keys to several different items, including the serial number embedded in several different places in the code, the copyright notice included in the front of the software, and certain checksum factors strung throughout the code. This is to also insure the integrity of the software - in order to catch any possible disk failures as well. If you get any "undocumented" error message, this is probably because you made a coding error. Be very careful in this, as some errors, when found by the checking software, will attempt to overwrite your disk directory with garbage - in order to destroy what appears to be a pirated software product.

COMPILING A NEW PEN PLOTTER INTO THE SOFTWARE with PLOTCOMP

The program PLOTCOMP accepts the file PLOTSPEC.SRC and processes this into a new code file to drive the plotting devices. It should be noted that the plotting software requires positions 0 thru 3 to be allocated to four different types of dot matrix printer-plotters. These are usually the MX70/80 and MX100 from EPSON. There can then be an unlimited (in a logical sense) number of plotters appended to the software by building a code

file for each plotter. The first plotter is indexed as number 4 and is known by default as HIPLOT. Other plotters, including the word processing printers, high performance industrial plotters and the like are referenced by the TYPE command in the PLOTS call (TYPE 5, 6, ...) or explicitly by number in the calling programs that may be linked to the software.

The program PLOTCOMP.COM builds a set of "code" which is interpreted by a virtual routine loaded at execution time. This interpreter will accept this code and dessiminate the data to what ever device it is routed to. It expects the code to provide several routines which will be executed as they are requested from the software. These routines, referred to as "modules", are listed by index. They must be presented to the compiler in order, by this index order.

Module 0 - Initializes the device for plotting

Module 1 - De-implements the device for plotting

Module 2 - Instructs the "pen" to drop to the surface. This may be an "IDLE" or "RETURN" command only if the pen instructon is handled by the movement modules.

Module 3 - Raises the "pen".

Module 4 - Directs the plotting movement in the "north east" direction, in relation to the present pen location. The top of the plotting surface is referenced as "north".

Module 5 - 11 - Directs the movement to East, SE, SOUTH, SW, WEST, NW, and North respectively.

Module 12 - Directs the device to provide an "inter - plot" pause. This could be an operator message, or a screen copy and erase (as on the Tektronics tubes) or any other such action desirable.

Module 13 - N (up to maximum memory capacity of 512 bytes). Allows the individual control of the pens. The first module is for pen 1, and so on. If there is any necessary pen manipulation in the device at startup, this should be done in the initialize routine, as the software does not execute an implicit call to PEN # 1.

FORMAT OF THE CODE ARRAY FROM PLOTCOMP

The code array that is produced from the compiler is organized into 2 groups. The first 16 bytes specify the STEP, XFACT, YFACT, MAXXS and MAXYS to the software. It should be noted that these are dynamically specified by the PLOTCOMP routine and any "hard wired" parameters put into the machine with the modification routine will be overwritten when a call to PLOTS is made for the standard routines. The next 496 bytes are allocated to driving the plotter with the following "primitive" commands.

Each command that allocates memory requires one byte for the command and one byte for each parameter. The commands are numbered beginning with 0, extending up through whatever is the top command number - each is listed.

The parameters can be decimal numbers, or negative decimal numbers. There is a small array of memory to which the negative decimal numbers refer. These are available for use in the conversion of the internal (binary) X and Y and PEN numbers or commands into different formats. The reference is as follows:

-1 : the sign of the last converted decimal item

-2 through -6 : the decimal number last converted from X or Y.

-2 through -4 : the COMPLOT number just converted (in 5 bit code, with the (maximum) 3 characters.

-2 through -6 : the TEKTRONIX number just converted with the ENHANCED (4096 x 4096) graphics included.

-7 The last converted decimal pen number

-7 The result of the last PENDOWNUP operation executed.

The operations may also reference literal strings (enclosed in quotes) for the PUNCH, DISPLAY, and LIST commands. This allows the easy transfer of string data to these devices. This is specifically excluded from the OUTPUT command, as there will probably be a combination of INPUT and other commands (such as WAITMS) which are used to support direct port devices.

PRIMITIVE CODES for PLOTTING MODULES

MECHANICAL - Required for each plotter specification. This has three parameters - the STEPS, XFACT, and YFACT parameters. Entered in INTEGER or REAL format. Must be first item for a plotter specification.

PLOTTER - Required. Three parameters supply the MAXXS, the MAXYS, and the PLOTTER INDEX code. Must be second item for each specification.

PUNCH- Routes items to the CP/M output port commonly referred to as READER/PUNCH, and usually used to supply modem support, second printer support, etc. One parameter - may be integer, negative integer, or literal string.

DISPLAY - Functions as PUNCH, except routes to the operator console.

LIST- Functions as PUNCH, except routes to the CP/M list device.

OUTPUT - Functions as PUNCH, except routes to a specified port code (parameter 1) the code or byte referenced by a negative integer (parameter 2). No literals available.

SENSE- used to support direct port manipulation. Reads the port specified by parameter 1, ANDS it with the byte specified by parameter 2, XORS this result with parameter 3 and continues this until a zero result is obtained.

Example : PORT : 43 "busy bit" : 2 "busy is HIGH"

SENSE 67 4 4

The 67 is decimal for 43H. The 4 is the byte represented bit 2 "on". The third parameter duplicates parameter 2 because "busy is HIGH" when ready for the next byte.

Example : PORT : 50 "busy bit" : 7 "busy is LOW"

SENSE 90 128 0

WAIT- Allows the program to wait a multiple of 500 microseconds. Parameter 1 specifies how many 500 microsecond periods to wait.

CONVERTX10A- Converts absolute X position to the -2 through -6 memory locations.

CONVERTY10A - Converts the absolute Y.

CONVERTX10R - Converts the relative X.

CONVERTY10R - Converts the relative Y.

CONVERTXCPS - Converts the X to COMPLOT format. This takes the fifteen bit integer (negative is o.k.) and breaks it into 3 segments of 5 bits each. It then adds each segment to 20H and stores it

CONVERTYCPS - Convert the Y in CPS format.

CONVERTEK- Converts the X and Y to the enhanced TEKTRONIX format used in the tektronix machines. It should be noted that this will drive the machines WITHOUT enhanced graphics as well, as they are upward compatible. The COLOR TEKTRONIX 4027 and 4113, etc. can use either this format or a far simpler, direct format.

READER- Accepts a byte from the CP/M READER device and test it for PARAMETER 1. If not, equal, it repeats. If parameter 1 is 255, it accepts any character.

INPUT - Functions as READER, except reads port specified by parameter 1 and tests based on parameter 2.

KEYBOARD- Punctions as READER, except used operator console input.

PENDOWNUP- If the pen is presently specified as DOWN, places parameter 1 into the -7 location for subsuquent routing. If pen is UP, places parameter 2 into -7.

CALL- Loads HL with the address of the internal conversion area (-1 through -7) and calls the location specified by parameter 1 and parameter 2 (in inverted format).

MICRO500 - Respecifies the WAIT period for the WAIT routine. Based on cycles through the routine:

```
WAITR:  DEC   D
        JP    NZ,WAITR
```

IDLE- Effects a NOOP - can be used to provide multiple entry points into a common routine.

RETURN - Returns control to the plotting software. Must be used to end each routine (module).

SUBR- This allows the user to load a subroutine into the "transient" area. This extends from 145H to 200H. Only 127 bytes may be loaded at any given time. If more room is needed, several subroutines may be executed in turn.

CALL- This allows the user to call the (previously entered) subroutine. It must reference the location in inverted byte sequence and be entered as (two) decimal quantities.

There is no example section to this part as the entire code structure, in totality or in part, is available through the INSTALL procedure. This generates the file PLOTSPEC.SRC which contains these types of codes. It also will compile this file (either after you have generated it by INSTALL or after you have entered and/or modified it by a text editor) and generate the associated PLOTSPEC.DAT file. Be sure to always re-install your dot matrix printer options into the file after compiling it, as each PLOTCOMP run will re-instate the DEFAULT parameters.

DEFERRED DEVICE MODIFICATIONS

This section covers items unique to deferred devices such as dot matrix printers. It is intended to serve as a guide to those needing to modify the operation of PLOTWARE-z in these sections. Do not proceed into this section until you have read the previous section. Much of the code common to this section and the previous section is explained there - and not here.

The INSTALL program will usually install whatever items are needed to specify a printer. It does this by the following manner:

a. Most printers and other devices will need some sort of sequence to specify "GRAPHIVS MODE". This is a ONE-TIME initialization phase.

b. Some printers may need to have each line preceeded by a special "line-intialization" sequence.

c. Some printers may need to nave each line terminated by a special "line termination" sequence.

d. Some printers may need each byte to be inverted to allow for the different ways that a data byte is to be presented to the printer(s).

e. Some printers may need the data length of that line presented to them in some fashion. The more popular types are included in the standard set of instructions.

f. Some printers need each data byte to be "OR-ed" to assure that certain bits are on in each data item.

These items are all controlled by a set of "control bytes" placed into the PLOTSPEC.DAT file at INSTALL time. These are automatic for direct printer type selection, and may be specifically, entered for other printers. Run the INSTALL program and proceed through the necessary prompts to accomplish this.

OPERATING and INSTALLATION INFORMATION

This section is intended to cover any items not specifically covered in other sections. It is necessary to read this section only if the other sections did not cover your specific operation questions.

The PLOTWARE-z system is distributed on diskettes with many different pieces of software. The diskette has very little room for any work files and should be copied onto more usable media before doing any graphics work. The original diskette then should be filed away for back-up use in the future, and for obtaining new releases of the software.

COMPILER SUPPORT FILES

The "compiler" support software consists of a file called "PLOTLIB.REL". This file holds all of the routines for all of the attachment options in a relocatable format for MICROSOFT compilers. It is written in an 8085 code assembler format, and will operate with any 8080, 8085, or Z80 system, provided that you have the MICROSOFT support programs needed to link these to your FORTRAN, COBOL, or BASIC programs.

The routines may be included in your overall system library by use of the routine LIB on your MICROSOFT diskette. This will result in a larger library for all of your programs and will require some extra time during the link phase. If the majority of the programs you use are non-graphics oriented, the library should be left as a separate file or put onto a diskette for usage only with graphics systems.

To build an "integerated" support library with your current MICROSOFT library, use the following method:

1. RENAME your library:

```
REN FORLIBX.REL=FORLIB.REL
```

2. MAKE THE NEW LIBRARY:

```
LIB
FORLIB=EPSNPLOT,FORLIBX
/E
```

The above sequence will build a new (FORTRAN) library and preserve your old library as FORLIBX.REL.

To compile and execute your program, the following sequence would be used:

```
F80 =MYPROG
L80 MYPROG,MYPROG/N/E
MYPROG
```

This is the same sequence that would have always been used. To

use the library without integrating it into the MICROSOFT library, the following procedure would be used:

```
F80 =MYPROG
L80 MYPROG,EPSNPLOT/S,FORLIB/S,MYPROG/N/E
MYPROG
```

This would perform the same sequence.

COMMAND FILE or INTERACTIVE PROGRAMS

The standalone programs (which also support the COMMAND FILE format), are called

```
CHART.COM and
OVERLAYS.OVR and
GRAPHIT.COM and          (if you choose to use this)
OVERLAYG.OVR and         (if GRAPHIT is included)
PLOTSPEC.DAT
```

on the distribution diskette.

DOT MATRIX PRINTERS - the PRINTMAP printers.

The dot matrix printers have a built in graphics support feature known as OKIGRAF or GRAFTRAX or something similar which allows the placement of about 10,000 to 30,000 dots to the inch on dot matrix printers. This gives an extremely dense and very good approximation to the lines drawn at any angle that the program desires. As a comparison, this resolution - with the APPLE "high-resolution" graphics as a comparison device - would give the same "density" in a square about 1.5 inches by 1.5 inches at 10,000 dots per inch. A better way to mentally compare this would be to imagine the APPLE sceen reduced to this size : it becomes obvious how the "blocky" appearance of the typical consumer computer will become very insignificant on this scale.

This feature creates some problems with the computer, however, because the obvious solution to the vector to raster conversion - the computation of a very large matrix - either in core or on disk - becomes very unwieldy long before any decent graphics appear. The data base mapping techniques take out this problem however, and pick up another problem as this one is solved. If you generate an 8 inch by 10 inch picture on the dot matrix printer with the "dense" option, the software needs to transfer bits to generate a matrix containing 1,300,000 elements (1.3 million - this is not a typo. error). Hence many times more bytes need to be transferred in the GRAPHICS mode than in the normal printing mode. It is suggested that the user obtain the fastest method available to his system to run the printer. The fastest is usually called a "parallel" or "Centronics" interface. If a serial interface is used, some slow operation may be experienced if the baud rate is very slow. For instance, on the C.ITOH PROWRITER 2, at 160 dots per inch, about 2500 bytes need to be

passed to the printer per "scan line" pass in the densest mode. This amount of data is quite a bit more than is passed to print just 132 columns! If at all possible, use a parallel interface to maintain the speed of the system.

There are several interfaces which cannot reliably operate at the higher speeds necessary to promote maximum speed printer speeds. The "high speed serial buffer" from OKIDAT and the "SERIAL BUFFER ADAPTER" from EPSON are most notable in this manner. If you experience erratic operation on these devices, try a slower baud rate or contact your printer distributor for a "microcode fix".

SUMMARY OF DISK FILES

GRAPHIT.COM (menu driven graphics)

CRT's, Plotters:

a. GRAPHIT.COM (on default disk)
b. OVERLAYG.OVR (may be placed on any disk - prefer default)
c. CHART.COM (on default disk)
d. OVERLAYS.OVR (may be placed on any disk - prefer default)
e. PLOTSPEC.DAT (on default disk)

DOT MATRIX PRINTERS:

a. GRAPHIT.COM (on default disk)
b. OVERLAYG.OVR (may be placed on any disk - prefer default)
c. CHART.COM (on default disk)
d. OVERLAYS.OVR (may be placed on any disk - prefer default)
e. PRINTMAP.COM (on default disk)
f. PLOTSPEC.DAT (on default disk)

CHART.COM (for COMMAND FILE plotting)

CRT"s, Plotters:

a. CHART.COM (on default disk)
b. OVERLAYS.OVR (may be placed on any disk - prefer default)
c. PLOTSPEC.DAT (on default disk)

DOT MATRIX PRINTERS:
a. CHART.COM (on default disk)
b. OVERLAYS.OVR (may be placed on any disk - prefer default)
c. PRINTMAP.COM (on default disk)
d. PLOTSPEC.DAT (on default disk)

PRINTMAP.COM (used automatically for others - for dot matrix)

Reference files:

a. PLOTWARE.DAT (holds last "picture" from any program) (required)
b. PLOTWARE.OVR (holds return overlay information for auto-restart of calling (chaining) program.

SYNTAX DESCRIPTION
of
PLOTWARE-z*

This section gives a description of the syntax of each PLOTWARE-z* function. Most modules are accessible in one of the two syntax forms listed below.

1. Your module may be accessible through the syntax used for standard FORTRAN calls, or other MICROSOFT compiler object programs, or others with compatible linkers.

2. Your module may be accessible through the syntax used for COMMAND FILE calls. These may also be entered one line at a time from the console in an "INTERACTIVE mode", allowing the system to process the commands one at a time.

Those modules which are not accessible in these two forms are listed with the differences in form. The structure of each call follows the same pattern. When a particular case requires a variation, this variation is noted at he time of requirement. The variation call will be in the form listed below.

VERB [parameter, parameter, ... , parameter]

Parameters may be optional, as indicated for each verb. All FORTRAN programs will require the FORTRAN CALL verb to precede the PLOTWARE-z* verb. The parameters are enclosed in parentheses, as per standard compiler syntax. As an example of the basic difference between these two forms, the following program calls indicate the command, in both forms, as required to establish a plot on a dot matrix of "TYPE 1".

EXAMPLE:

FORTRAN call:

```
CALL PLOTS (x,y,1)
```

(In this case "x" and "y" are specified parameters.)

COMMAND FILE or INTERACTIVE call:

```
PLOTS TYPE 1
```

You will notice a difference in the basic appearance of these two calls. The FORTRAN call simply reflects the traditional calling structure of FORTRAN, while the COMMAND FILE and INTERACTIVE calls reflect their free-form, BASIC-like structure.

This part of the manual describes the "syntax" of the commands of the system. The "syntax" can be said to be similar to the "grammar" of a speaking language. Hence there are some conventions of syntax which are followed here.

A. Each command begins with a VERB or COMMAND. This command is used to invoke a computing module of the system. Hence it is referred to as a "module".

B. Each module may be followed by modifiers which are required

to enable the computing module to perform the commanded task. These may be referred to as "parameters".

C. The parameters are grouped into three categories. They may be **VARIABLES**, signified by upper case letters which are explained in the text following the module declaration. These may take on numeric values. In the compiler forms, of course, these may remain variables, but in the COMMAND FILE form, they must be expressed as their actual value.

D. Another form of the parameter is the "literal". This is a string of characters which are alphanumeric in nature and may be used for titles, headings, and similar things. They are always represented as strings of characters enclosed in quotes or apostrophes (sometimes referred to by programmers as a "single quote"). This allows the inclusion of either of these special characters within the literal. The only rule to follow when forming literals is to always terminate the literal in the same way that you began it. For example, if you started with quotes, finish with quotes. Here are two properly formed literals:

"Anybody's guess"

'AND WITH APOSTROPHES'

E. The last form of parameter is the **DISK FILE NAME**. This is usually enclosed in quotes or apostrophes as well. The name is formed in the standard way in which your operating system requires disk files to be formed. This is explained in your computer system documents and will not be approached here.

F. Some parameters may be enclosed in parentheses in the **COMMAND FILE** form. This is usually the case when optional forms of the modules are available to extended functions.

G. As is true of verbs, some modules may have more than one "conjugation". For example, the **AXIS** module has six forms.

H. While entering the actual items for parameters and/or literals, use these rules:

1. All literals will appear as entered.

2. Parameters may have upper or lower case letters interchanged.

PLOTWARE-z® contains many facets of operation, in it's complex structure. These facets are loosely grouped into "modules". Each "module" has been directed to perform one specific project on a given graphics device.

These modules are described below, in alphabetical order.

1. **ARC:**
This module serves the purpose of generating arcs

(portions of a circle) of specified dimensions.

2. **AXIS:**
   This module serves the purpose of generating axis groups for graphs. It includes all variation calls: AXISA, AXISB, AXISC, AXISD, and AXISLG. It also includes information on the special entry point GRID.
3. **CHART:**
   This module will generate the PIE and BAR charts. It includes verbs which provide for special shading and coloring accents.

4. **CIRCLE:**
   This module generates cirlces of specified dimensions.

5. **DATA:**
   This module is used in the COMMAND FILE and INTERACTIVE mode in order to enter data into the plot.

6. **PRINTMAP:**
   This module will automatically draw dot matrix printer graphs from the data base generated for dot matrix (raster scan) printers. However, when stated as a stand alone verb at the operating system level it's function will be to generate multiple copies of a graph on the printer.

7. **EXIT:**
   This module allows for the termination of the interactive or disk attachment program.

8. **FACTOR:**
   This module allows for the reduction or enlargement of all subsequent graphic commands.

9. **FONT:**
   This module changes character sets. It may be implicitly called from SYMBOL

10. **FONTFQ:**
    This module allows the selective modification of the ornate characters appearance. Width, height, slant, and skew can be added to any of the characters with this feature.

11. **GRID:**
    This module generates linear, logarithmic and linear-log grids with solid borders. The user has a choice of solid, dashed, or mixed grid lines. The GRID command should be specified immediately before the AXIS command to insure a GRID graph.

12. **LINE:**
    This module draws line points or lines which convert the data values to the proper perspective of the axis used in inches. LINELG and LINEX variations are included in this module.

13. **NEWPEN:**

This module specifies pen changes for plotters and ribbon changes for word processing printers. This module specifies action for the ribbon carrier, if so equipped, on word processing printers. It will also specify special action by dot matrix printers to emmulate color.

14. **NUMBER:**
This module allows the drawing of numbers in any size and at any angle. It includes these variations of the term **NUMBER:**

NMBERL = justification of number used to the left.
NMBERR = justification of number used to the right.
NMBERC = justification of number used to the center.

15. **PLOT:**
This module draws "point data" in "pen" movements. **PLOT** is the basic routine of all of the other modules and is frequently called upon by the **PLOTWARE-z®** program. It's accessibility to the user gives him the ability to manipulate the x - y data and order the rudimentary "pen up" and "pen down" movements. Extensions here include enlarging or shrinking the plot and rotating it about any point inside or outside of your object. **PLOT** is useful for collecting a series of plots, specified by various programs, into one composite plot. It is also useful for logos and other frequently plotted items.

16: **PLOT3D:**
This module is for compiler attachment and is analagous to the PLOT verb except that X,Y,Z data is used to create a 3D drawing. See also verbs ZPERSX, ZPERSY, ZSWAPX, and ZSWAPY for complete 3D information.

17. **PLOTS:**
This module primarily serves to initiate a graph. It specifies a "convenient" page size for graphic displays when using dot matrix or word processing printers. It also describes the **PLOTSX** variation.

18: **REDRAW:**
This module allows the user the ability to create a packed binary disk picture of his picture (through the PICTURE BUFFER as installed at your specifications) and then redisplay this picture at some other time. It can be enlarged, reduced, rotated, projected into three dimensions, or any combination of these actions.

19. **SCALE:**
This module has two primary functions. First, it examines raw data and calculates scaling factors. Second, it adjusts these factors in order to allow for a pleasing, even-numbered annotation value to be used in the x - y line graph. The following three extensions are included within this module.

SCALEX = extended scaling
SCALEL = logarithmic scaling
SCALEM = manual scaling

20. **SYMBOL:**
This module draws printed symbols, or special "point" data to accentuate "points" of the graph. Symbols may be produced to any size specifications and any angle of inclination. SYMBOL allows the text strings to be justified to the left, to the center or to the right.

21. **WHERE:**
This module allows the user to return the pen to the last exact drawing location. (compiler usage only)

22. **XYSIZE:**
This module allows the user to change the size of the drawing page. XYSIZE is only applicable to dot matrix printers.

23. **ZPERSX:**
This module allows the user to establish a perpective in three dimensions. It provides for a true three dimensional viewing perspective allowing for the selective viewing of the object from any viewpoint, at any rotation. It provides a mechanism to allow magnification and projection, simulating a microscope with magnification far greater than 10,000X to a wide angle view greater than any current "fish-eye" lens currently available. The ablity to produce animation and the "sense of movement" (in flight, etc.) is available through this module.

24. **ZPERSY:**
This module cancels the effects of ZPERSX.

25. **ZSPEED:**
This module specifies the speed of the CPU in systems requiring wait cycles. This feature is found only in printers which operate under the "older" GRAFTRAX (EPSON trademark) eproms.

26. **ZSWAPX:**
This module allows the "swapping" if the X specification with the Z specification in the three dimensional features. This is a short cut to producing a 90 degree rotation of the selected plane.

27. **ZSWAPY:**
This module allows the "swapping" if the Y specification with the Z specification in the three dimensional features. This is a short cut to producing a 90 degree rotation of the selected plane.

This concludes the introduction to the **PLOTWARE-z®** modules. It is now time to look in depth at the working purpose of each module.

ARC:
The ARC module allows the user to draw an arc on the plotting

surface.

SYNTAX:

(COMMAND FILE FORM)

ARC XPAGE,YPAGE,DIAMETER,ANGLESTART,ANGLESTOP

where:

XPAGE, YPAGE describe the center of the arc.

DIAMETER is the diameter of the arc in inches.

ANGLESTART and **ANGLESTOP** define the beginning and ending locations of the arc, with respect to the center. These are in degrees.

EXTENDED SYNTAX:

(COMMAND FILE FORM)

ARC XPAGE,YPAGE,STEPSIZE,DIAMETER,ANGLESTART,ANGLESTOP

Where:

STEPSIZE is a NEGATIVE number whose absolute value describes the distance between points used to draw the arc. It is usually .01 in the standard form, but can be modified in the extended form in order to more efficiently take advantage of the media on which the arc is drawn. On CRT screens which have a 500 or 600 by 200 or 300 pixel matrix, the value should be around .06. On Plotters which have very fine resolution and require a perfect arc to be drawn (as on the ZETA 8 plotter) the number may be reduced to .003 or so in order to obtain a perfect arc on the paper.

AXIS:
The purpose of **AXIS** is to generate an x - y axis, which can be used with X -Y graphs and bar graphs. Grids may be generated when **AXIS** is used, first, for the vertical lines, then for the horizontal lines, in conjuction with the verb **GRID**. The grids may be solid lines, dashed lines or a combination thereof. **AXIS** will also serve to generate any special purpose display, which requires the drawing of a ruled line, or "tic mark" indexed segments with possible annotations on the segments, or the axis itself.

SYNTAX:

standard FORTRAN:

basic x - y axis:

```
CALL AXIS (XPAGE,YPAGE,IBCD,  NCHAR,AXLEN,ANGLE,
FIRSTV,DELTAV)
```

This form generates an axis, with annotations expressed in the specified values, with two decimal points. The tic marks used in the annotation are 1 inch apart. This is the same syntax construct used in the "large mainframe" software.

EXTENSIONS:

The following extensions allow the user to vary the x -y graph.

INTEGER ANNOTATION: (TYPE "A")

```
CALL AXISA (XPAGE,YPAGE,IBCD,NCHAR,AXLEN,ANGLE,
FIRSTV,DELTAV,TICFAC,NUMDEC)
```

This form gives the same axis type as above, but the distance between tic marks and the number of decimals to be used on the tic mark annotations are specified.

EQUI-LENGTH ALPHANUMERIC NOTATION: (TYPE "B")

This allows for an x - y axis with alphanumeric annotation on the tic marks. All of these annotations are a specified number of characters in length.

```
CALL AXISB (XPAGE,YPAGE,IBCD,NCHAR,AXLEN,ANGLE,
FIRSTV,DELTAV,TICFAC,TICARA)
```

COMMAND FILE and INTERACTIVE:

```
AXISB XPAGE YPAGE IBCD NCHAR AXLEN ANGLE
FIRSTV DELTAV TICFAC annotation-length "annotation"
```

In this case, FIRSTV and DELTAV are included for adjustment and use with the SCALE and LINE routines. The actual annotations used at the tic marks are not plotted to reflect these values. They are computed with these values, but are taken from the literal array TICARA. This allows the annotations to express months, coded limits and items which are not usually expressed in a numeric form.

ANY-LENGTH ALPHANUMERIC NOTATION, PUBLICATION FORM: (TYPE "C")

This provides an x - y axis with extended alphanumeric labels which are not of the same length and are too long to fit on the annotation tic marks, without running together.

```
CALL AXISC (XPAGE,YPAGE,IBCD,NCHAR,AXLEN,ANGLE,
FIRSTV,DELTAV,TICFAC,TICARA)
```

COMMAND FILE and INTERACTIVE:

```
AXISC XPAGE,YPAGE,IBCD,NCHAR,ASLEN,ANGLE,
FIRSTV,DELTAV,TICFAC," ... annotation ...."
```

This places the annotation perpendicular to the axis. It adjusts the label of the axis to leave sufficient room for the longest annotation to be placed on the tic mark. This is the normally requested format of abstract and formal presentation papers. This program will give camera ready copies without the usual extra drafting required for professional presentations.

PUBLICATION FORM, FOR X AXIS USE: (TYPE "D")

This provides an x - y axis with the annotations of AXISC. (AXISC will not draw on a 45 degree angle.)

CALL AXISD (... (SAME PARAMETERS) ...)

The annotations are placed at a 45 degree angle to the axis. This leaves room for long annotations, yet maintains the readability of the chart.

LOG AXIS ANNOTATION: (TYPE "LG")

If this form is used to express the graph in terms of the magnitude (LOG 10) of the data, rather than in the actual data itself, a special module for the drawing of axis is encountered, since the tic marks, annotations, and lines are all expressed in terms of the magnitude of the data.

CALL AXISLG (XPAGE,YPAGE,IBCD,NCHAR,AXLEN,ANGLE,FACTL1, FACTL2)

FACTL1 and **FACTL2** are calculated with **SCALE** using the log version. Always call **SCALE** (log version) before calling **AXISLG**. Then use the calculated values for **FACTL1** and **FACTL2**. These correspond to **FIRSTV** and **DELTAV** of the linear axes.

(more on **FACTL1** and **FACTL2** in "MANUAL SCALING")

INTERACTIVE or COMMAND FILE SYNTAX:
AXIS = same parameters as above, without parentheses
AXISx = same parameters as above, without parentheses.

(x is A,B,C,D, or LG)

XPAGE,YPAGE:
These are coordinates of the starting point of the axis, in inches, with respect to the present "origin". These must be zero when using **LINE** and **SCALE** routines to adjust data to the axis. Use **PLOT** to change the "logical" origin if necessary. Listed here are two instances when this change would be necessary.

1. It would be necessary when placing several plots on one page.

2. It would be necessary when the origin is the lower left-most position on the plotter and the ticmarks, annotation and label would be positioned outside of the plotting "window". (This is always true on dot matrix printers.)

IBCD:
This is the annotation data to be used as the title of the axis. This can be a literal or an array of any type desired, provided the data characters are contiguous.

NCHAR:
This indicates the number of characters in the title of the axis. If it is negative, the title is placed "below" the axis. (clockwise). This is the usual case with the "x" axis. If the number is positive, the title is above the axis. In the **INTERACTIVE / COMMAND FILE** form, this may be 99 or -99: The actual length of the title, as determined by counting the characters in the literal, will be used for this parameter. The labourious counting of characters for the title can be eliminated by this form of call.

AXLEN:
This is the length of the axis, in inches.

ANGLE:
This is the angle at which the axis is to be drawn. This is usually expressed as 0.0 for the X axis and 90 for the Y axis.

FIRSTV:
This is the beginning value, to use to annotate the tic marks, on the axis. It is ignored, when annotating in the extended versions which use alphanumeric annotations. In the **INTERACTIVE / COMMAND FILE** form, this may be 999 and the previously calculated scaling factors from **SCALE** will be used for **FIRSTV** and **DELTAV** or **FACTL1** and **FACTL2**.

DELTAV:
This is the increment value to apply to the tic marks, in their annotation (from the **FIRSTV** value). It is also to be ignored if one is using alphanumerical annotations.

TICFAC:
This is the distance, in inches, between the tic marks of the axis.

NUMDEC:
This specifies the number of decimals to be used as annotations past the decimal point. If the given value is zero, no decimal point will be drawn.

TICARA:
This is a logical array with the tic mark annotation information. When all of the elements of the annotations are of the same length, element one is the length of the annotation of each tic mark. Elements 2 through N are the tic mark annotations themselves. This must be a single byte/character array. If this is entered in the **INTERACTIVE / COMMAND FILE** form, this entry is made with two parameters. The first indicates the length of each parameter. The second is the annotation itself.

EXAMPLE:
The following would produce an axis labeled with abbreviations

for the first six months of the calendar year.

3 "janfebmaraprmayjun"

This same example, stored in the array, for Fortran use, and used as a literal in the calling parameters would appear as follows.

"3janfebmaraprmayjun"

This is a logical array for variable length and longer annotations, since the annotations are presented in groups. Each group will consist of a logical value, which gives the length of the annotation, followed by the annotation itself. These groups are preceeded by a single value, which is the maximum annotation length.

Example:

If your annotations are "ABC", "DEFGH", and "XY"
then TICARA would be:

"53ABC5DEFGH2XY" (in FORTRAN)

INTERACTIVE / COMMAND FILE format calls for the specification to be made by using a series of parameters. The first is the total number of literal strings used as the annotation. In the above case, this would be:

... 3 "ABC" "DEFGH" "XY"

When building an array, in a compiler program, which will link the routines as inline subroutines, the length parameters can be expressed as either the actual values of the lengths or as the literals representing these lengths (as seen above). When annotations are greater than nine characters in length, the use of literals will assist to express this length. Use the ASCII chart to find literals for greater value than nine in order to express this length.

Example: Use a ":" for a length of 10 and a ";" for a length of 11.

MANUAL SCALING:
Manual scaling may be used if you wish to specify your own parameters.

FOR LINEAR AXES:
The **FIRSTV** references the first number to occur at the origin of the graph. It is the beginning value to be used on the graph. **DELTAV** is a number, which will be added to **FIRSTV**, to obtain the printed value at the next mark.

FOR LOGARITHMIC AXES:

FACTL1 and **FACTL2** correspond to **FIRSTV** and **DELTAV**, however their meanings are related to BASE 10 functions.

FACTL1:
This is the lowest "POWER OF 10 VALUE" at the axis origin. It can not be any number other than a POWER OF 10. This assures an accurate representation by the decile placement of the axis.

FACTL2:
This is the number of log-cycles which will be represented in a one-inch portion of the graph. A simple mathematical manipulation also shows this to be the inverse of the length of one log-cycle.

NOTE:
Scaling is a complex art and extensive understanding of this art should be acquired before a serious attempt is made to manually scale the data. Do not, for instance, set **DELTAX** to 11 and **FIRSTX** to 7. This would result in a very unusual graph. Always try to use numbers with multiples of 2,5 or 10. You might use 4 or 8 if necessary. Make sure that the entire scale is specified and will fit exactly into the axis you have requested. This involves the length of the axis and its "TIC FACTOR", which is the length between each tic mark.

CHART

CHART is a verb wnich invokes the "business chart" and functions with bar and circle charts.

SYNTAX: (COMMAND FILE or INTERACTIVE only)

CHART [kind] [modifiers]

where [kind] is:

PIE or CIRCLE

followed by [modifiers] :

```
XPAGE,YPAGE,DIAMETER,
DATA (color) "segment label" AMOUNT,
DATA...(repeated as needed for each segment),
END
```

or where [kind] is:

BAR

followed by [modifiers]:

```
XPAGE,YPAGE,XLENGTH,YLENGTH,VERTICAL or HORIZONTAL code,
BARWIDTH,BARSPACE,"X label","Y label"
DATA (color) "bar label",AMOUNT1,AMOUNT2,SHADE,DENSITY,
DATA...(repeated as needed for each amount or bar) END
```

Here is an explanation of the "modifiers".

XPAGE,YPAGE:
This is the "origin" of the chart.

In the creation of a CIRCLE or PIE chart, XPAGE YPAGE is the center of the circle. In the creation of the BAR chart XPAGE YPAGE is the lower left corner of the axis group.

DIAMETER:
This is the diameter of the circle for the CIRCLE or PIE chart. It is expressed in inches.

XLENGTH, YLENGTH:
This specifies the length of each axes used to build the BAR chart. XLENGTH is for the horizontal axis and YLENGTH is for the vertical axis. These are expressed in inches.

VERTICAL OR HORIZONTAL:
This is a required entry in the BAR charts. It specifies that the bars drawn will be drawn either vertically or horizontally. One and only one of these options must be in each CHART specification. Use the number 1 to indicate preference of a HORIZONTAL graph and 0 to indicate preference of a VERTICAL graph.

BARWIDTH:
This specification can be optional, but some value must be present. If the width is set at 0, the software will calculate and present the most appropriate width and distance between the bars. If the user opts to order a different width setting, the opportunity is provided to enter data into BARWIDTH and BARSPACE.

BARSPACE:
This specification is required. It signifies the distance between the bars of the chart. A carefully calculated division of the bars will insure an attractive graph.

"X label":
This is the label to be used on the horizontal axis.

"Y label":
This is the label to be used on the vertical axis.

DATA:
This item begins an entry which will specify the quantity of a portion of a chart. The verb DATA must be entered before each quantity of each portion of a chart is assigned. Anotner DATA statement or an END statement will conclude each quantity entry. **DO NOT CONFUSE THIS STATEMENT WITH THE X - Y DATA STATEMENT.** see page k-14.

(color):
This is an integer item which may specifiy a color to a specific segment or bar of the chart. COLOR is optional and may be hardware dependent.

"segment or bar label":
This is the actual label appended to the segment or bar of the

chart. The label will be included inside the segment of a circle chart. If the label requires more room than is allowed inside the segment, it will be produced outside of, but pointing directly to the segment. If a BAR chart is drawn, the label will be used as the individual annotation on the axis which serves as the base axis for the bars. Caution should be used when specifying these, for lengthy segments will result in a clumsy appearance of the chart. They will be drawn by the software, but the large space allocated to a long label might mar the professional appearance of the chart.

AMOUNT:
This is the quantity to be assigned with each segment or bar of the chart. When producing a CIRCLE or PIE chart, the software will check all amounts encountered until the END statement, for "reasonable values". This means that the amounts encountered must be equal to or less than 100%. If the amount exceeds 100%, an error message will result. If the amounts entered are equal to less than 100% this will facilitate the automatic inclusion of another segment labeled "OTHER", as needed to fill out the pie to 100%. Its color will be in the default color (usually black or "pen number 1") of the hardware device drawing the chart. Its amount will be the difference between the total of the specified amounts and 100%.

AMOUNT1:
This specifies the amount of the "major" bar of the drawn bar.

AMOUNT2:
This specifies the amount of the "sub bar" of the drawn bar.

AMOUNT1 and **AMOUNT2** are used to allow the inclusion of two-piece bars, on the bar chart, as shown in the examples given for the BAR CHART functions.

SHADE:
The numbers 1 or 2 or 3 will specify the amount of shading to be given to a bar. The number 1 orders that no shading is to be performed. The number 2 provides for single, angled lines. The number 3 provides a cross hatched pattern, for the densest shading available. The angle at which the lines are drawn is calculated based on the angle between opposite corners. This gives a consistency to the appearance of each bar on the chart.

DENSITY:
This specifies the distance between shading lines. **DENSITY** is determined by specifying how many lines are to be drawn per inch. This is a required entry. Even if tne **SHADE** option is 0, this entry is required for proper syntax.

END:
This item must be the last item of any chart specification. The chart process begins the drawing of the chart, following the occurrence of the **END** verb. The user may then use other CHART options, or any of the other available options such as SYMBOL, to label the overall page on which the chart is located. If no

other items are required, the normal item to be entered following the END would be an EXIT statement.

OPERATIONAL NOTE on CHART:
As the CHART program runs, it examines the amount of memory available. It then calculates the amount of "line" storage available. This calculation is displayed, along with the copyright notice, before any plotting action begins.

CIRCLE:
The CIRCLE module allows the user to draw a circle on the plotting surface.

SYNTAX:

(COMMAND FILE FORM)

CIRCLE XPAGE,YPAGE,DIAMETER

where:

XPAGE, YPAGE describe the center of the arc.

DIAMETER is the diameter of the circle in inches.

EXTENDED SYNTAX:

(COMMAND FILE FORM)

CIRCLE XPAGE,YPAGE,STEPSIZE,DIAMETER

Where:

STEPSIZE is a NEGATIVE number whose absolute value describes the distance between points used to draw the circle. It is usually .01 in the standard form, but can be modified in the extended form in order to more efficiently take advantage of the media on which the circle is drawn. On CRT screens which have a 500 or 600 by 200 or 300 pixel matrix, the value should be around .06. On Plotters which have very fine resolution and require a perfect circle to be drawn (as on the ZETA 8 plotter), the number may be reduced to .003 or so in order to obtain a perfect circle on the paper.

DATA:
The purpose of **DATA** is to specify a data set to be used in X - Y line graphs. Raw data (not converted to inch equivalents) must be used. If converted data is used, it must be accompanied by the appropriate scaling factors included as the last two data points of the data set. (see **MANUAL SCALING** in the **AXIS** section) Each data set must have the data ordered into X - Y pairs. These X -Y pairs must be in a free form ASCII format, with integer or real (decimal) numbers. Any number of **DATA** verbs may be used in the generation of an X - Y graph, but the total number of data point pairs must not exceed the maximum number allocated from the memory space. This

number is always displayed when the CHART program is initiated.

SYNTAX - COMMAND FILE:

DATA n "datafile"

where:

"datafile" is in the standard disk file naming convention: NNNNNNNN.EXT

Following the occurrence of "data file", a search will begin for X - Y pairs and will continue until an end of file entry is encountered. The structure of the file is free-form. All X-Y pairs are represented as integers, reals or a combination of the two. Any number of X - Y pairs may be in a single record. There is no realistic limit to the number of X -Y values which can be contained in one logical record. The only formatting requirements are that an X - Y pair cannot span a logical record, and a single X or Y value may not span a record. "n" is a number which orders that number of records to be read from the file prior to processing the data. It can be omitted if no records are to be skipped. When several data sets are placed on a single graph, each data set must be specified by a separate DATA verb statement. The first statement will be numbered as 1. As the SCALE and LINE statements are used to enter insuing data sets, an implicit numbering system will yield the necessary line number, required for all later statements. Therefore, if the software has encountered 6 DATA statements, it will number them as LINE 1, LINE 2, LINE 3, LINE 4, LINE 5 and LINE 6. When, at any point after entry is completed, the user communicates with the software, about a data statement, LINE and the appropriate number must be referenced.

When buiding an X - Y graph, the following order must be observed.

DATA statement
.
.
.
DATA statement (the last one)
SCALE statement (for the X's, if needed)
SCALE statement (for the Y's, if needed)
optional **GRID** statement
AXIS statement
optional **GRID** statement
AXIS statement
LINE statements (one for each **DATA** statement)

PRINTMAP:
The purpose of **PRINTMAP** is to draw the graphic display on dot matrix printers. It is the essential process of vector to raster conversion that is required for the dot matrix printers to generate the entire, complete picture in a single pass. There is no practical limit to the number of pictures which can be stored in a single conversion

file in a data base format, because each picture becomes one single unit of the complete plot. Most "usual" graphs and charts of reasonable complexity require between 25 and 40 sectors (of 128 bytes each) to store the data base segment for that graph or chart. PRINTMAP may also be used as a seperate progam to REDRAW the last picture (s).

SYNTAX:

PRINTMAP

This call is at the operating system level. The picture name is always PLOTWARE.DAT.

EXIT:
The purpose of **EXIT** is to provide a way to close the data base in the **INTERACTIVE / COMMAND FILE** form of tne **PLOTWARE-z®** system and to begin the processing of the data. It is not required in self terminating plotter and word-processing printer systems. This option begins an automatic call to **PRINTMAP**. The graphic data generated will then be drawn on the printer.

SYNTAX: (for FORTRAN)

CALL PLOT (0.,0.,999)

This form was chosen in order to retain consistency with the syntax of large mainframe computers.

DO NOT USE THE FORM: CALL EXIT

SYNTAX: (Command File)

EXIT

FACTOR
The purpose of this routine is to provide the ability to modify the actual size of the displayed picture. If the **FACTOR** is set to .5, for instance, all subsequent display commands will produce a half-size product. This can be useful in the debugging stages of a graph or chart. It snortens the plotting time and gives a "rough" miniature copy of the final product.

SYNTAX :

standard FORTRAN:

CALL FACTOR (FACT)

FACT is the desired factor.

INTERACTIVE / COMMAND FILE call:
FACTOR factor

FONT:
The **FONT** module directs the system to use a new set of characters in the generation of any further alphanumeric representations on the graphs. This is to allow the user access to the **"HERSHEY FONTS"** (named after the man responsible for digitizing them) in pictures. This is an optional feature and must have the proper options installed in PLOTWARE-z® in order to make the characters available.

SYNTAX (compiler form)

CALL NEWFNT(literal)

SYNTAX (COMMAND FILE form)

FONT literal

Where:

The literal describes the name of the font to be used. The tables of the fonts are in the example section. This literal must match the name at the heading of the table of characters which you desire to access. The name is limited to six characters, so you need enter only the first six characters of any name.

GRID:
The purpose of **GRID** is to allow a grid to be generated on an **AXIS** set. It is often useful to apply an overlay, of some sort of grid, particulary when using log plots, to more accurately display the line data. This option allows a "solid line" grid, composed of evenly spaced lines, log spaced lines, or a solid line at regular intervals with dashed lines between the solid lines. **GRID** is actually a special entry point into the **AXIS** command and is invoked by specifying the **GRID** verb immediately before specifying the **AXIS** verb. It is not necessary to avoid intervening commands, but the **GRID** verb effectively "turns on" the gridding software, and the **AXIS** verb, as it finishes, "turns it off".

SYNTAX (COMMAND FILE or INTERACTIVE)

GRID GRIDHEIGHT DASHLENGTH DASHINTERVAL

where:

GRIDHEIGHT:
This is the height of the lines to be drawn perpendicular to the axis. The **GRIDHEIGHT** of the X axis would normally be the length of the Y axis.

DASHLENGTH:
This is the length of the dashes, when dashed lines are used. Solid lines will always occur at the ends of the axis, but may be specified at regular intervals if the **DASHINTERVAL** option is given a quantity , in inches, other than zero.
DASHINTERVAL:
This is the interval at which solid lines are to be encountered.

If DASHINTERVAL is 5, every fifth line will be solid. If DASHINTERVAL is zero, all lines will be solid.

LINE:
The purpose of line is to place a data set, as specified by X - Y data points, onto a line or X - Y graph.

SYNTAX

Fortran ("standard call")

```
CALL LINE(XARRAY,YARRAY,NPTS,INC,LINTYP,INTEQ)
```

Fortran ("extended call")

```
CALL LINEX(XARRAY,YARRAY,NPTS,INC,LINTYP,INTEQ,TICFAC,
           TICFACY)
```

COMMAND FILE or INTERACTIVE :

```
LINE  linenumber,NPTS,INC,LINTYP,INTEQ
LINEX linenumber,NPTS,INC,LINTYP,INTEQ,TICFACX,TICFACY
```

LOG type (only)

```
CALL LINELG(XARRAY,YARRAY,NPTS,LINTYP,INTEQ,LOGTYP,TICFACL)

LINELG linenumber,NPTS,LINTYP,INTEQ,LOGTYP,TICFACL
```

where:

XARRAY, YARRAY:
These are the REAL arrays used in the FORTRAN program to store the X and Y data.

NPTS:
This is the number of points in the arrays of X and Y data. Also see **AXIS** for an explanation of the manual scaling process. If **NPTS** is zero, then (in **COMMAND FILE** or **INTERACTIVE** mode only) the actual number of data points, will be used in this value. This saves laborious counting of the data points.

INC:
This is an optional increment for selecting data points. It is usually set at 1. If it is set at 2, then every other data point will be selected. If it is set at 3, then every third point will be selected, etc.

LINTYP:
This is a parameter, which specifies an optional marker, to be drawn on each specified point. This marker can be one of fourteen different centering markers. These are shown in the chart of symbols in the example section, page K-50 If LINTYP is zero, the line drawn will have no markers drawn at any point. If LINTYP is greater than zero, the data is represented by a line,

with markers drawn at specified points on that line. For instance, if **LINTYP** is valued at 3, a marker will be presented at every third point. If **LINTYP** is less than zero, no line appears. Only marker symbols will be drawn. These will be at intervals specified by the absolute value of **LINTYP**. If **LINTYP** is -3, there will be no line. Only markers will be drawn at every third point, etc. It is easily seen that **LINTYP** is usually -1, 0 or 1.

INTEQ:
This specifies which marker symbol to use. See the chart of symbols page to verify the the appearance of the markers that **INTEQ** will yield. These choices are seen in the first fourteen entries shown in the first column.

LOGTYP:
0 means "LOG-LOG"
1 means "LOG-linear in X"
-1 means "LOG-linear in Y"

TICFACX,TICFACY are the "ticfactors" for the plot.
See **AXIS** and **SCALE** for details.
TICFACL is the tie factor for the linear axis, if one is used.

NEWPEN:
The purpose of **NEWPEN** is to provide the user with the opportunity to change to a different colored pen, if the hardware of the user will allow for this option. This ability is usually found only on specially equipped, multipen plotters, dual color ribbons on word processing printers, or on color CRT'S.

SYNTAX

CALL NEWPEN(colornumber)

NEWPEN colornumber (COMMAND FILE or INTERACTIVE)

where:

colornumber is a number in range of the hardware device. For example:1 - 4 on the CPS plotter. This device is for a special use on dot matrix printers.

EXAMPLES:

CALL NEWPEN (3) (Fortran)

NEWPEN 2 (COMMAND FILE / INTERACTIVE)

FOR DOT MATRIX PRINTERS:
It is possible to generate very impressive pictures on dot matrix printers, by using a "three color process", which is offered by 3-M and other manufacturers. This requires three sheets for the picture (split into the three primary colors).

The process is as follows:
RUN YOUR "JOB" three times.
The first time, immediately after the PLOTS statement use NEWPEN -1 (or call NEWPEN (-1) for Fortran.
The second time use -2 in the NEWPEN statement.
The third time use -3 in the NEWPEN statement.

This gives the three sheets needed for the color process. The actual process will not be explained here. See your 3-M dealer for additional information.

TABLE:
"Color Mixing" for single color dot matrix printer:
Use NEWPEN for colors 1-7 to get the folowing combinations.

NEWPEN	COLOR MIX (mixes these colors)
1	1 and 2 and 3
2	1
3	2
4	3
5	1 and 2
6	1 and 3
7	2 and 3

Obviously "color 1" will usually give a "black" color. The white, or "light" color possible with this 3 color process is used as the background or "filler" color. This would correspond to "color 8".

NOTE:
There are no examples of the above commands as with the other verbs. This is obviously due to this documentation being available in only one color of ink. It is left to the reader to visualize the effects of multi-color plots.

NUMBER
The purpose of this routine is to "draw" real (floating point) numbers on the plotting device, with any necessary scientific notation as needed to represent a floating point number. The software is capable of drawing this in any size, rotating the numbers at any angle, and centering the product about a given point, as well as right or left justifying the number at any point specified. These options are identical to the options available with the SYMBOL routine.

SYNTAX

Fortran: (standard, "compatible" call)

```
CALL NUMBER(XPAGE,YPAGE,HEIGHT,FPN,ANGLE,NDEC)
extended calls (left justify, center, right justify)
CALL NUMBRL(XPAGE,YPAGE,HEIGHT,FPN,ANGLE,NDEC)
```

CALL NUMBRC(XPAGE,YPAGE,HEIGHT,FPN,ANGLE,NDEC)

CALL NUMBRR(XPAGE,YPAGE,HEIGHT,FPN,ANGLE,NDEC)

COMMAND FILE or INTERACTIVE : same, without parentheses

where:

XPAGE,YPAGE:
These specify a point where the number is to be referenced. This is the lower left of the leftmost character position, if the number is to be left justified, the lower right of the rightmost character, if it is to be right justified, or the midpoint between these two positions, if the number is to be centered. These measurements are in inches, relative to the (current) origin of the plot.

HEIGHT:
This is the height, in inches, of each character used to make up the number as it is drawn.

FPN:
This is the floating point number that is to be drawn on the plotting area.

ANGLE:
This is the angle at which the number is to be rotated, as referenced to the plotting surface. The rotation occurs about the point referenced by XPAGE, YPAGE.

NDEC:
This determines the number of "decimal precision" digits which are to be displayed, provided that NDEC is greater than or equal to zero. As the number is drawn, the whole number part of the number is drawn, then the decimal, then NDEC digits are drawn to the right of the decimal.

If NDEC is -1, only an integer (no decimal point) is drawn.

For compatibility reasons, the following two features are included:

If NDEC is less than -1, the ABSOLUTE VALUE of NDEC -1 determines the number of digits truncated from the number while plotting. For instance, if 1234.56 were plotted with NDEC = -2, this would result:

123

The "continuation feature" on XPAGE and/or YPAGE:
When X is equal to 999.0, then X is continued from the x position where the last SYMBOL call left off. If Y is equal to 999.0 then Y is continued from the Y position where the last SYMBOL call left off. If both are equal to 999.0 the drawing of text continues immediately from the final position of the previous call to SYMBOL.

PLOT:
The purpose of **PLOT** is to allow X - Y point data to be entered onto the plotting surface. This is the mechanism used by all of the other routines to "draw" on the plotting surface. The procedure consists of simulating the human hand as it draws. The pen is lifted up, placed at a convenient "beginning point" and left down as it draws straight lines from that point to the next point of the line or curve to be drawn. Logically speaking, all lines are a collection of straight lines connecting many points, hence curved lines or complex figures are a collection of many, probably very short, lines. The PLOT routine expects the data to be presented as X - Y point data. The third specification of each point is that of the position of the pen, as that point is approached. This is the universal approach to all plotting of **PLOTWARE-z®**, regardless of the type of hardware which is used on the computer.

SYNTAX

Fortran:

CALL PLOT (XPAGE,YPAGE,IPEN)

COMMAND FILE or INTERACTIVE:

PLOT data

or (extended form 1)

PLOT (PFACTOR XP1 YP1 ANGLE) data

or (extended form 2)

PLOT (PFACTOR XP1 YP1 ANGLE XP2 YP2) data

Extended form #1 is used to place the plot data, specified in the "data" portion of the **PLOT** verb, with respect to **XP1** and **YP1**. This is the specification of a new "temporary origin" which is in effect only during the period of entering the "data" onto the plotting surface. In addition, the "data", whether it is actual data coded into the **COMMAND FILE** or **INTERACTIVE** statement, can be rotated about the origin of the data itself. It should be noted that the data may be in a file, or may have been entered by digitizing some figure from a graphic digitizer. This file will have an "origin" of its own. This would be the original "origin" which was referenced during the digitizing process. It is this original "origin" about which the data is rotated.

Extended form #2 is identical to extended form #1 except that the user can re-specify the "original origin" of the "data", brought in by the **PLOT** statement, before the rotation takes place. For instance, assume that a LOGO has been digitized, and the logo is 5 inches by 5 inches. The "original origin" which was referenced by the digitizer, was the lower left corner of the logo. The user now wishes to draw a 1 inch square logo on the front of a part which he has designed and drafted with **PLOTWARE-z®**. He also needs to turn the

logo 90 degrees in order to give it the proper placement on the part. It would be most convenient to specify that the CENTER of the logo was the position about which to rotate the logo. This would avoid some clumsy shuffling with a calculator or pencil. Therefore, he would let XP1 and YP1 refer to the very center of the logo position on the part he had designed. PFACTOR would be 0.2. ANGLE would be 90.0, and XP2 and XP2 would now be 2.5 each, in value. This would effectively shift the origin of the original data, as held in a separate "LOGO" file, so that the origin would appear to be at the center of the logo. The logo would be rotated about this center, which would be the exact point on which it was to appear on the part.

PARAMETER EXPLANATION:
XPAGE,YPAGE is the point referenced.

IPEN must be 2 to specify "pen down".
IPEN must be 3 to specify "pen up".
IPEN can be -2 or -3, and the effect will be to move to the point specified, then make that point the NEW origin of the plot. (It will then appear to be 0.0,0.0). For "pen up" use -3. For "pen down" use -2.

IPEN may be 0. If this occurs, this indicates that this is the last action to be taken on this particular plot segment. It can be used to make indefinitely long plots with dot matrix printers, by specifying the plot as several segments, or it can be used to allow the user to service a pen plotter or word processing printer before drawing another plot. This may be useful in producing a smooth flow with an otherwise complex drawing problem.

PFACTOR:
This is a factor which can be used to reduce or enlarge the size of the data in the plot statement. It is referencing that data only, and no subsequent verbs will be affected by it.

XP1 and YP1:
This is to reference a "temporary reference origin" (or "where exactly do I put this data on the plot?") for the placement of the "data".

ANGLE:
This is the angle of rotation (about the "original" origin of the "data" referenced by the verb PLOT) which is to be performed on all of the "data". The "original" origin is usually the same as the "plot origin". Examples would include logos, electronic symbols, Egyptian hieroglyphics, or mathematical symbols.

XP2 and YP2:
These refer to an adjustment to be performed to the "original origin" prior to rotation of the "data".

"data":
These may be one or more X - Y pairs with the IPEN specification attached and followed by the word "END". It may be a file name

specified as a literal which will take the X - Y pairs, with an IPEN accompanying each, from that file name. This data is in free-format, and may be integer or real, except that the X and the Y and its accompanying IPEN value must all be encountered in the same logical record.

OTE: When using the "embedded" form of plot data, there MUST be an END statement following the last X - Y and IPEN value. The software also expects to find an END statement at the end of all "external" plot data files.

Example:

PLOT 1 1 3 2 2 2 1 2 3 2 1 2 END
will draw an "X" of dimension 1 inch by 1 inch.

If the picture can be drawn in segments or "frames", each data base segment may be terminated, with the special inter-plot call to **PLOT**, which uses a zero value for **IPEN**. If the dot-matrix option is specified, the word **STAT** will be displayed, followed by the listing of the statistics of that segment of the data base. There will be 21 items displayed. The first item is the actual number of 128 byte sectors required by that data base segment. The following 20 numbers are the NODE usage indices (the number of "tree branches" encountered in the data base structure itself). Each NODE usage index is a critical number, for it must be equal to or less than the maximum NODE capacity of the vector to raster scan program (PRINTMAP) or that program will terminate. This is not an actual NODE count, but rather a percentage based on the capacity of the ENERCOMP TWIN-Z computer which has about 80 K individual partition memory, with 63 K available to the user per partition. The capacity of this machine is taken as 100. This is enough data base capacity to generate a "very black" picture, which is actually covered with lines about 2.5 times. In a typical 48 K system (the smallest size recommended) this number hovers around 65. This is still a very large capacity. The usual problem encountered, when this limit is exceeded, will be some loop in the program which tends to draw the same line or figure thousands of times.

PLOTS:
The purpose of **PLOTS** is to begin the plotting process. It MUST be used once, and once only, to initialize the process. It defines the type of hardware available to the software. If any hardware parameters are modified, they must be modified prior to the call to PLOTS. It effectively specifies that an origin exists at the lower left corner of the drawing area, which is set to 8 by 10 inches, for dot matrix printers, type 0 and 1, or to 15 by 10 inches, for the dot matrix printers, type 2 and 3. This origin is of no real consequence on pen plotters or similar devices, which do not require the extra vector to raster scan conversion step after the plotting is finished. These devices usually allow the user the pleasure of manually positioning the "pen" at some "nice" place before he begins the plotting procedure.
The practice of specifying the origin at the lower left-most position is both easily understood and most logical, until one realizes

that the line graphing routines will try to label the axis BELOW and to the LEFT of this point. Obviously this will result in lost parts of the picture, since the dot matrix printer is unable to "move" to that area. The extended form of the **PLOTS** verb is **PLOTSX**, which addresses this problem.

PLOTSX:
This is used as a convenient way to specify that a one inch "offset" is used on the "windowed" plotting devices, such as a dot matrix printer. This allows the extra room needed below and to the left of the origin to draw the tic marks, annotation, and titles for line graphs. This inch is usually a "sufficient" offset amount, unless unusually long annotations are used. The equivalent structure would be found in the use of the PLOT verb by saying:

```
CALL PLOT (1.0,1.0,-3) (FORTRAN)
or:
PLOT 1 1 -3  END  (COMMAND FILE / INTERACTIVE)
```

SYNTAX

```
PLOTS hardware statement  (ZSPEED and XYSIZE must be
                           here, if used)
PLOTSX hardware statement    (ZSPEED, XYSIZE, ...)
```

where:

The hardware statement is one or more of the following options.

HARDWARE	FORTRAN (third parameter)	COMMAND FILE
eight-inch wide dot matrix printer "dense mode"	0	TYPE 0
eight-inch wide dot matrix printer "fast mode"	1	TYPE 1
13.6 inches-wide dot matrix printer "dense mode"	2	TYPE 2
13.6 inches-wide dot matrix printer "fast mode	3	TYPE 3
Your Pen Plotter	4	TYPE 4
Your Graphics CRT	5	TYPE 5
Your Word Processing Printer/Plotter	6	TYPE 6

INSTALL may be used to place any item you define as types 4, 5 or 6. (See Install Section.)

REVISION 1.4.....(6-83)

REDRAW:

The purpose of **REDRAW** is to produce a new picture from a picture previously saved by the PLOTWARE-z system. The picture must have previously been created by a run through a PLOTWARE-z program with the plotter deviced specified as the PICTURE BUFFER. The output of this was a file called PLOTWARE.BUF. It can be renamed to other standard names though any external means. The referencing of that file can then be made to draw another picture. This is a convenient way to compose complex presentations, ship pictures to other locations without shipping your software, and transmit final pictures from one location to another by electronic means. Since colors and large sizes are preserved, it gives a graphics user with an inexpensive praphics device (such as a dot matrix printer) the ability to refine his output to the final stage, then transmit the finished picture (in the full size, with colors and other enhancements) to some central site for final display.

SYNTAX

COMMAND FILE or INTERACTIVE:

(form 1):

REDRAW "filename"

(form 2):

REDRAW (PFACTOR XP1 YP1 ANGLE) "filename"

(form 3):

REDRAW (PFACTOR XP1 YP1 ANGLE XP2 YP2) "filename"

It can be seen that this is an identical form to the PLOT statement for displaying data in disk files. Refer to that module for further explanation.

SCALE:
The purpose of **SCALE** is to produce a convenient transformation from the "raw data" that is to be plotted, with its natural tendency to be just about any range of numbers imaginable, to the necessary inch measurements of an X - Y graph. It will also produce reasonable tic mark annotations for the size graph requested, in such a manner as to present an "even" appearance.

SCALE does not recompute any numbers it examines. Hence it does not require duplicate storage or extra overhead requirements from the computer. It does produce a "starting value" and an "increment value", referred to as FIRSTV and DELTAV respectively. These will later serve as factors to be used when plotting the graph. These factors are also used by the axis labeling routine and are always stored in two extra locations above the X - Y data. If the user wishes to compute his own FIRSTV and DELTAV, they should be stored there, if the subroutine LINE is used. Otherwise they may be passed

directly as parameters to the AXIS routines. See AXIS for more details on **FIRSTV** and **DELTAV**.

SYNTAX

Fortran "compatible" form:

```
CALL SCALE(ARRAY,AXLEN,NPTS,INC)
```

FORTRAN "extended" form:

```
CALL SCALEX(ARRAY,AXLEN,NPTS,INC,TICFACTOR)
```

Fortran LOGARITHMIC form:

```
CALL SCALEL(ARRAY,AXLEN,NPTS)
```

COMMAND FILE or INTERACTIVE:

```
SCALE X or Y,AXLEN,NPTS,INC

SCALEX X or Y,AXLEN,NPTS,INC,TICFACTOR

SCALEL X or Y,AXLEN,NPTS

SCALEM X or Y,FIRSTV,DELTAV (for manual scaling)
```

where:

ARRAY:
This is the dimensioned (in Fortran) array which houses the data to be scaled. It will usually be either an array of X values or an array of Y values.

AXLEN:
This is the length, in inches, of the AXIS which is to serve as the reference for that array. If the data is the X data, then this length will be the length of the X axis.

NPTS:
This is the number of points in the array. In the **COMMAND FILE** and **INTERACTIVE** modes, this may be zero, and the data will be counted by the software.

INC:
This can be used for two purposes. It will specify if the axis is to reflect data in a "descending" fashion (annotation wise) or in an "ascending" fashion. It will also give the option of selecting only a portion of the data in the ARRAY, at regular intervals, for the scaling calculation.

If INC is greater than zero, the data will be plotted with "ascending" notation. The value calculated for **FIRSTV** will be the minimum value which the software will expect to find in the data for that axis.

If INC is less than zero, the data will be plotted in a "descending" manner. The annotation of the tic marks will "descend" as the axis progresses. The FIRSTV calculated will be the maximum value that the software will expect to find in the data, for that axis.

The ABSOLUTE VALUE of INC can specify that data is to be sampled at regular intervals, other than 1. For instance, if INC is +3 or -3 then every third point will be sampled to compute FIRSTV and DELTAV.

STORAGE of FIRSTV and DELTAV:
These values are stored in the next two locations above the data, in memory, when INC is +1 or -1. If INC is not +1 or -1, then FIRSTV will be stored at the next specified interval from the last point selected for sampling, with DELTAV occupying the next specified interval after that used to store FIRSTV.

TICFACTOR:
This is the distance between tic marks, as expected when the linear axis is drawn. It is expressed in inches.

SCALEM:
This allows "manual scaling". (See AXIS)

SYNTAX

SCALEM n FIRSTV,DELTAV

where : n is X or Y

This allows the user to insert manual parameters into the system.

SYMBOL:
The purpose of SYMBOL is to produce "drawn" alphanumeric lettering in the drafting style for annotating the graphs and for marking the data points on the drawn lines of an X-Y line graph. It is installed in the software, as delivered, with a "default character set" which is described in the examples page K-50 The first fourteen characters are symbols which are centered on data points of X-Y line graphs for point accentuation. The remaining characters are available as needed to build text or numbers for the plot. This character set may be changed by the user with optional software, allowing the inclusion of mathematical symbols, script lettering, foreign text characters, and the like.(See FONT) SYMBOL allows the text generated to be any reasonable size, with centering, or right or left justification about a point. The text may also be rotated any number of degrees about that point. If the optional "ORNATE FONTS" are installed, this character information is gathered from the last call to FONT.

SYNTAX

Fortran: (standard, "compatible" call)

```
CALL SYMBOL(XPAGE,YPAGE,HEIGHT,"text",ANGLE,NCHAR)
```

extended calls (left justify, center, right justify)

```
CALL SMBOLL(XPAGE,YPAGE,HEIGHT,"text",ANGLE,NCHAR)

CALL SMBOLC(XPAGE,YPAGE,HEIGHT,"text",ANGLE,NCHAR)

CALL SMBOLR(XPAGE,YPAGE,HEIGHT,"text",ANGLE,NCHAR)
```

COMMAND FILE or INTERACTIVE : same, without parentheses.

where:

XPAGE,YPAGE:
This specifies a point where the characters are to be referenced. This is the lower left of the left-most character position if the text is to be left justified, the lower right of the right-most character, if it is to be right justified, or the mid-point between these two points, if the text is centered.

HEIGHT:
This is the height, in inches, of each character used to make up the text.

"text":
This is the literal string or logical array, which is to be drawn on the plotting area.

If "text" is displayed as ASCII characters, the first character of this text is always zero or greater than zero. A special case exists when "text" is taken as a single variable (logical or integer), whose value is between -1 and -14, inclusive. This allows the access of the "centered symbols" of the first fourteen positions of the chart, for accentuating points. When this occurs for "text",only one symbol is drawn.

ANGLE:
This is the angle at which the text is to be rotated, as referenced to the plotting surface. The rotation occurs about the point referenced by XPAGE, YPAGE.

NCHAR:
This determines the number of characters of text which are to be displayed, provided that NCHAR is greater than or equal to zero. If NCHAR is less than zero, a centered symbol, selected from the first fourteen symbols of the chart, will be drawn at the point referenced. If NCHAR is -1, the pen is "up" while moving to that point. If NCHAR is -2, the pen is "down" while moving to that point. The symbol generated is determined by the absolute value of the logical variable used in the place of "text".

For compatibility reasons, the following feature is included:

The "continuation feature" on XPAGE and/or YPAGE:
When X is equal to 999.0, then X is continued from the x position where the last SYMBOL call left off. If Y is equal to 999.0 then Y is continued from the Y position where the last SYMBOL call left off. If both are equal to 999.0 the drawing of text continues immediately from the final position of the previous call to SYMBOL.

WHERE:
The purpose of **WHERE** is to provide a "feedback" mechanism in the FORTRAN versions to indicate the current position of the "PEN". This is not usually needed, as the location of the pen is usually apparent to the programmer. However, in the case of using **PLOTWARE-z*** calls, which perform their own pen commands, such as **SYMBOL** or **NUMBER**, it is sometimes helpful to know just where the "PEN" finished drawing. This call will return the information of that location to the programmer, as well as returning the current "factor" which is in effect.

SYNTAX:

CALL WHERE(RXPAGE,RYPAGE,RFACT)

XYSIZE:
The purpose of **XYSIZE** is to provide a different sized plot from the normal 8 x 10 or 10 x 13.6 provided by dot matrix printers.

This routine is a subroutine under FORTRAN:

CALL XYSIZE (XLENG,YHT1,YHT2)

where:
XLENG is the new standard page length for the printer.

YHT1 is the new height (carriage width) for the narrow printer.

YHT2 is the new height for the wide printer.

In the **COMMAND FILE** format, this MUST be on the **PLOTS** command line. It cannot be elsewhere.

Examples:

FORTRAN:

CALL XYSIZE (100.,8.,13.6) (creates LONGGGGG plots)

COMMAND FILE :

PLOTS ... XYSIZE 5 8 13.6 ... (creates short plot)

ZPERSX:

The purpose of ZPERSX is to define the perspective of the user for three dimensional projections. The concept of three dimensions is a complex one, and the full explanation of that concept is not the purpose of this manual. A short summary of this follows, however.

Two approaches to three dimensions are usually taken. The first, or traditional method, introduces the concept of some imaginary "vanishing point" where all parallel lines of an object seem to converge. Drafting technicians and engineers often call the usage of this concept "orthogonal projection". A more realistic view is found when two vanishing points are used, letting the object be rotated about 45 degreed and letting the observor view the object "head-on". If the object consists of mostly conventional shapes (rectangles, smoothly curved surfaces, etc.) the appearance is somewhat realistic. The most realistic of the "traditonal approach" methods is the usage of three "vanishing points", where the viewer usually "looks down" or "looks up" at the object.

Two severe drawbacks are found in this approach. The first is that rotation of the object is severly hampered, because the realistic appearance of the view is highly dependent on the angle to the observor. The second drawback is this: the model is not based on sound mathematics, but on some "seat of the pants" guess or some "pleasing to the eye" decision. With some calculations, one may produce "vanishing points" that are close to the mathematically correct points, but for the purpose of this program, this approach is discarded.

The second approach (as used in PLOTWARE-z) is to define an object in matrix form (in this case 3 x 3) and mathematically derice the true projection into a plane that is perpendicular to the viewer. This plane is obviously analgous to a CRT screen, the finished drawing from your dot matrix printer, plotter, etc. The viewer can then get as close to or remove himself as far away form the origin of the object (expressed in X, Y, Z form) as he desires. He can, of course, even place himself WITHIN the object prior to viewing, but the finished product may appear somewhat confusing. The viewer may also position the screen at any place. This allows for a very wide range of magnification and reduction. In addition to this, he may RESPECIFY the X, Y, Z origin at any time or directly rotate the X or Y plane prior to it projection.

The derivation of this is found in many books. The parametric summary is as follows:

a. There exists an object to be viewed, expressed as a set of X, Y, and Z location points.

b. There is an observor whose "single point of observation" is at some point. This point is at an angle with respect to the Z axis of the object. This angle is referred to as THETA.

c. The observor is a distance away from the X, Y, Z origin of the viewed object. This distance to the "single point of

observation" (close one eye please!) is RHO.

d. The observor, in addition, is some angle away from the X axis of the object. This angle is PHI.

e. Finally, the observor is "looking at" (with his one eye, of course - keep the other closed during this explanation) the projection of the object onto a screen. The observor is some distance away from the screen. This distance is D.

Here are some practical exercises to use to confirm these concepts mentally.

Place a sheet of paper on a table in front of you. This is an X-Y plane.

a. Stand directly over it and look down directly on it. At this point the Z axis is directly through your "line of site". At this point, PHI is 0.

b. Rotate the paper COUNTER-CLOCKWISE. You have just move the paper in the THETA direction, positively. YOU, however, as the observor, have experienced a NEGATIVE shift in THETA, relative to the paper. You could, of course, have directly simulated this by walking to the LEFT.

c. Draw back away from the table. You have just changed your your PHI positively. You no longer look directly down at the paper ("looking directly down means zero degrees PHI).

d. Imagine the "cone of vision" formed by the left-most, the right-most, the bottom-most, and the top-most parts of the observed item (the piece of paper). This cone of vision (in this case, more of a pyramid) holds the key to the concept of the "screen of projection". At any place in this cone of vision, you may place the screen, as long as it is perpendicular to your line-of-sight. Now you can see that the screen becomes your final resting place for your image. It can be from the plotter, the dot matrix printer, or your CRT graphics screen. The screen is not limited to position between your viewpoint and the object. It may be BEHIND the object. In this way, it MAGNIFIES the object. Use this, of course for display of small objects. When between the observor and the object, it REDUCES the object. Use this for large objects. The distance to the observor is D.

e. One last experiment will cement the concept of Z into the picture. The paper rests on the table. it references the X-Y plane on its surface. The actual X-Y plane is the table top, for this purpose. Now carefully raise the paper and assume that it is exactly parallel to the table top, but one inch above it. You have just changed the Z of the object from 0 to 1.

f. In reference to the table top, you may change the origin at any convenient time. Use N3DORG X Y Z in the COMMAND FILE

mode or PLOT3D(X,Y,Z,-3) in the compiler mode to move this reference point.

THIS IS THE ONLY EXPLANATION OF N3DORG.

The SYNTAX is now presented.

SYNTAX:

COMMAND FILE mode:

ZPERSX RHO,THETA,PHI,D,Z

compiler mode:

CALL ZPERSX(RHO,THETA,PHI,D,Z)

Try this to get started (in CHART):

```
PLOTS TYPE n
PLOT 1 1 -3 END
PLOT 0 0 3  0 5 2  5 5 2  5 0 2  0 0 2  END
EXIT
```

This draws a 5 x 5 box.

Now:

```
PLOTS TYPE n
PLOT 1 1 -3 END
ZPERSX 5 -90 30 5 0
PLOT 0 0 3  0 5 2  5 5 2  5 0 2  0 0 2  END
EXIT
```

This draws the same box tilted AWAY (from the -90) from you and at a 30 degree angle (PHI).

ZPERSY:

The purpose of this module is to cancel the effect of ZPERSX and return to two dimensional representation.

SYNTAX: (COMMAND FILE)

ZPERSY

ZSPEED:
The ZSPEED command is used for certain EPSON printers which cannot

provide the proper data handling without loosing data at high speed transfers. This is particularly directed toward the buffered interface.

The standard factor is 16 (4 times a 4 MHZ processor speed).

The following factors are used for other common processors:

13 - The INTERTEC QD and COMPUSTAR (uses significant amount of CPU for screen handling - results in slower execution)

8 - For 2 MHZ processors (8080, Z80, etc.)

9 - For 8085 with 5 MHZ crystal

15 - For 8085 with 10 MHZ crystal (using typical wait states)

24 - For the ENERCOMP TWIN - Z processor (or other Z-80 processors with 6 MHZ CPU)

Adjustment may be made downward for wait states.

Examples

FORTRAN :

```
CALL ZSPEED (ISPEED)
```

COMMAND FILE :

```
PLOTS    ...   ZSPEED  24   ....
```

ZSWAPX:

This rotates the plane on the X axis 90 degrees into the Z axis. It effectively "swaps the X and the Z".

SYNTAX: (COMMAND FILE)

ZSWAPX

ZSWAPY:

This rotates the plane on the Y axis 90 degrees into the Z axis. It effectively "swaps the Y and the Z".

SYNTAX: (COMMAND FILE)

ZSWAPY

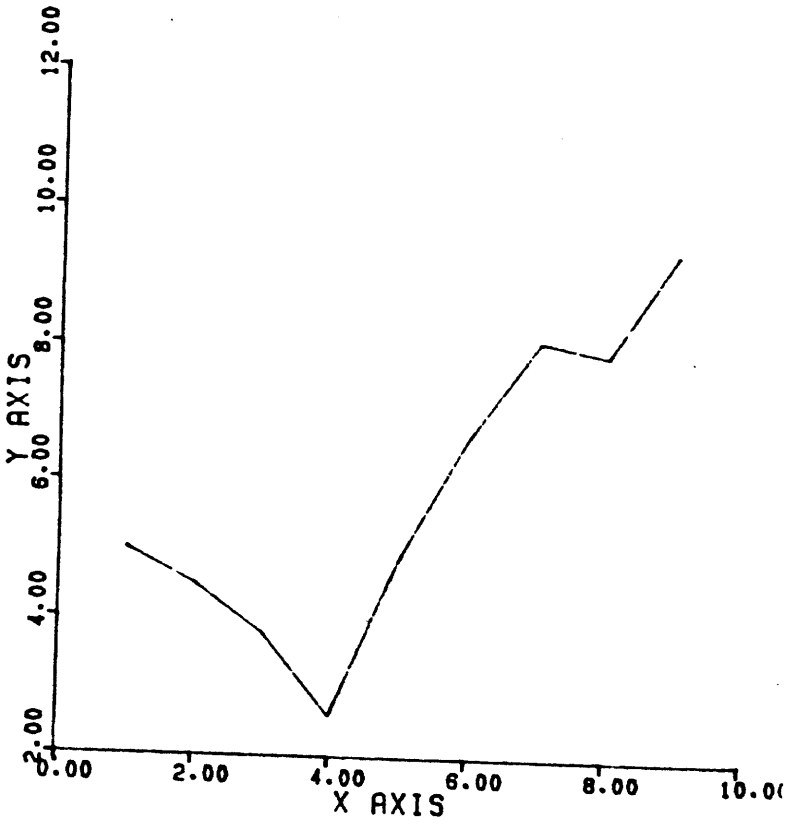
```
****** THE TEST FILE FOR AXIS GENERATION "TESTAXIS.CMD" ******

 .OTS TYPE 4
rICT -.5 -.5 -3 ENI
DATA "TESTDATA.LOG"
SCALEX X 8 0 1 .4
SCALEL Y 6 0
AXISE 0 0 "ALPHA X ANNOTATION" -99 8 0 99 0 .4 3
"JANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDECJANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDE(
AXISIG 0 0 "LOG Y AXIS" 99 6 90 99 0
LINELG 1 0 0 0 1 .4
SMBOLC 4 6.5 .12 "TEST OF EXTENDED AND LOG SUBROUTINES:" 0 99
SMBOLC 4 6.3 .12 "LINE, SCALE, AND AXIS" 0 99
PICT 0 0 0 ENI
DATA "TESTDATA.LOG"
SCALEX Y 6 0 1 .6
SCALEL X 8 0
AXISE 0 0 "ALPHA Y ANNOTATION" 99 6 90 99 0 .6 3
"JANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDECJANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDE(
AYISIG 0 0 "LOG X AXIS"-99 8 0 99 0
LINEIG 1 0 0 0 -1 .6
SMBOLC 4 6.5 .12 "TEST OF EXTENDED AND LOG SUBROUTINES:" 0 99
SMBOIC 4 6.3 .12 "LINE, SCALE, AND AXIS" 0 99
PLOT 0 0 0 ENI
DATA "TESTDATA.LOG"
SCALEL Y 6 0
SCALEL X 8 0
AXISLG 0 0 "LOG Y AXIS" 99 6 90 99 0
AYISIG 0 0 "LOG X AXIS"-99 8 0 99 0
LINEIG 1 0 0 0 0
SMBOLC 4 6.5 .12 "TEST OF LOG-LOG SUBROUTINES:" 0 99
SMBOIC 4 6.3 .12 "LINE, SCALE, AND AXIS" 0 99
PLOT 0 0 0 ENI
DATA "TESTDATA.LOG"
SCALEL Y 6 0
SCALEL X 8 0
GRID 8 .1 0
AXISIG 0 0 "LOG Y AXIS" 99 6 90 99 0
GRID 6 .05 0
AXISIG 0 0 "LOG X AXIS"-99 8 0 99 0
LINELG 1 0 0 0 0
SMBOIC 4 6.5 .10 "TEST OF LOG-LOG SUBROUTINES WITH GRID:" 0 99
SMBOLC 4 6.3 .08 "LINE, SCALE, AND AXIS" 0 99
PIOT 0 0 0 ENI
DATA "TESTDATA.LOG"
SCALEL Y 6 0
SCALEL X 8 0
GRID 8 0 0
AXISLG 0 0 "LOG Y AXIS" 99 6 90 99 0
GRID 6 0 0
AXISIG 0 0 "LOG X AXIS"-99 8 0 99 0
DATA "TESTDATA.001"
SCALE X 5 0 1
SCALE Y 5 0 1
AXIS 0 0 "X AXIS"-99 5 0 99 0
```

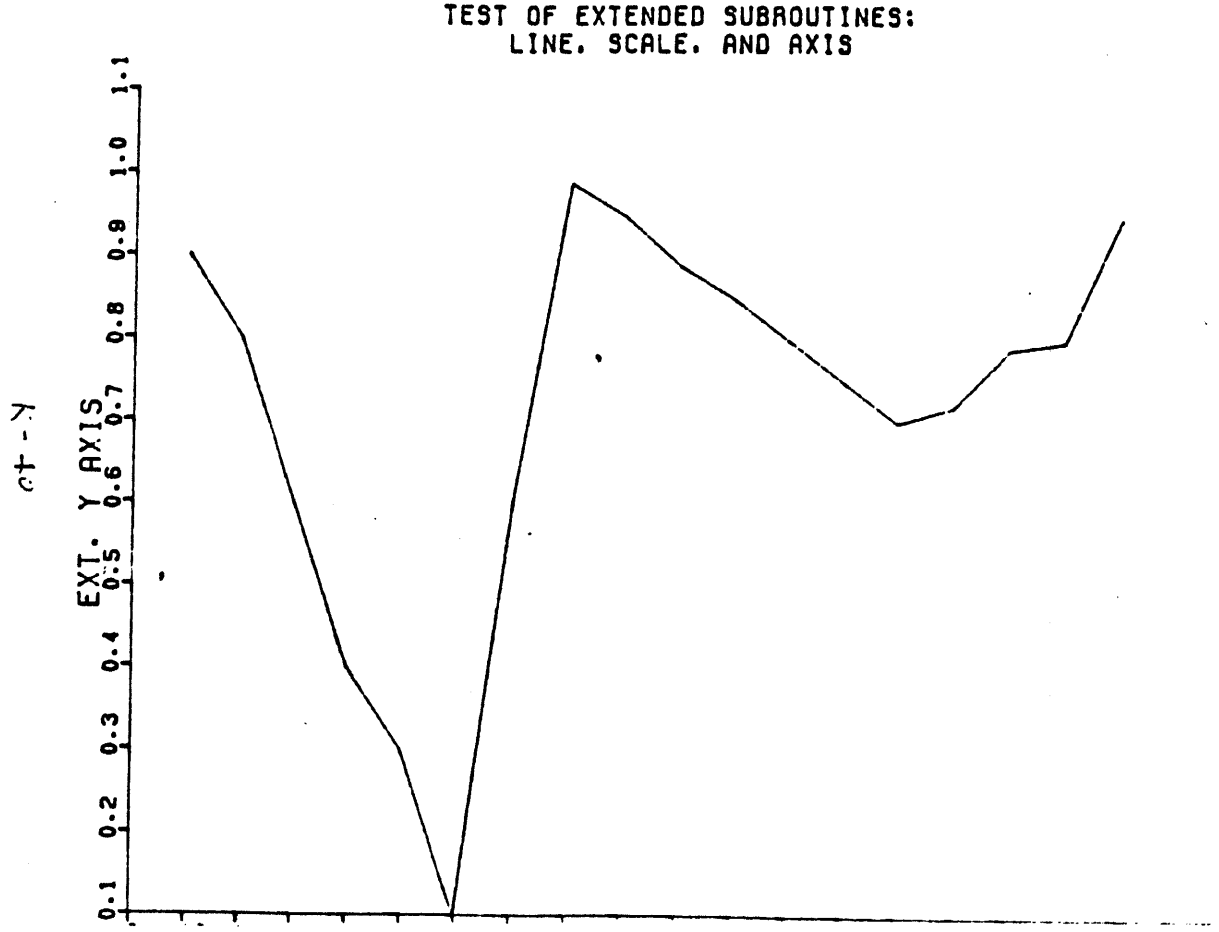
```
AXIS 0 0 "Y AXIS" 99 5 90 99 0
LINE 1 0 1 0 0
SMBOLC 2.5 6 .12 "TEST OF STANDARD SUBROUTINES:" 0 99
SMBOLC 2.5 5.8 .1 "LINE, SCALE, AND AXIS" 0 99
PICT 0 0 0 END
DATA "TESTDATA.002"
SCALEX X 8 0 1 .4
SCALEX Y 6 0 1 .6
AXISA 0 0 "EXT. X AXIS" -99 8 0 99 0 .4 0
AXISA 0 0 "EXT. Y AXIS" 99 6 90 99 0 .6 1
LINEX 1 0 1 0 0 .4 .6
SMBOLC 4 6.5 .12 "TEST OF EXTENDED SUBROUTINES:" 0 99
SMBOLC 4 6.3 .12 "LINE, SCALE, AND AXIS" 0 99
PICT 0 0 0 END
DATA "TESTDATA.002"
SCALEX X 8 0 1 .4
SCALEY Y 6 0 1 .6
AXISB 0 0 "ALPHA X ANNOTATION" -99 8 0 99 0 .4 3
"JANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDECJANFEBMARAPRMAYJUNJULAUGSEPOCTNOV
AXISA 0 0 "EXT. Y AXIS" 99 6 90 99 0 .6 1
LINEX 1 0 1 0 0 .4 .6
SMBOLC 4 6.5 .12 "TEST OF EXTENDED SUBROUTINES:" 0 99
SMBOLC 4 6.3 .12 "LINE, SCALE, AND AXIS" 0 99
PLOT 0 0 0 END
PICT .5 .5 -3 END
DATA "TESTDATA.002"
SCALEX X 8 0 1 .4
SCALEX Y 6 0 1 .6
AXISD 0 0 "45 DEGREE ALPHA X ANNOTATION" -99 8 0 99 0 .4 23
"GROUP A" "GROUP B" "SEG 1" "SEG 2" "SEG 3" "T-1" "T-2" "T-3"
"T-4" "T-5" "T-6" "A" "B" "C" "I" "E" "TEST 10" "TEST XYZ" "TEST CHEC
"X-1" "X-2" "X-3" "X-4"
AXISC 0 0 "90 DEGREE ALPHA Y NOTATION" 99 6 90 99 0 .6 11
"N-1" "N-2" "N-3" "N-4" "FIFTH" "SIXTH" "SEVENTH" "EIGHT" "9TH" "10" "
LINEX 1 0 1 0 0 .4 .6
SMBOLC 4 6.0 .12 "TEST OF EXTENDED SUBROUTINES:" 0 99
SMBOLC 4 5.8 .12 "LINE, SCALE, AND AXIS" 0 99
EXIT


****** TEST FOR BARCHART COMMANDS "BARCHART.CMD" ******

PLOTS TYPE 4
SMBOLC 5 5 .14 "APPENDIX B - ANNUAL REPORT"0 99
CHART_BAR 1.2 .6 3 3 1 0 .325 "EXPENSES, 1000'S" "SALES PERSON"
DATA "TAYLOR" 65 62 3 20
DATA "S.SMITH" 75 70 3 20
DATA "B.SMITH" 62 60 3 20
DATA "BROWN" 40 30 3 20
END
SMBOLC 2.5 4.5 .12"EXPENSE REPORT BY INDIVIDUAL" 0 99
SMBOLC 2.5 4.3 .09"SHADED PORTION IS TRAVEL-RELATED EXPENSE" 0 99
SMBOLC 2.5 4.15 .085 "FOR THE PERIOD 1-80 THRU 12-80" 0 99
CHART_BAR 6.2 1.5 3 2 0 .25 .5 "WAREHOUSES, BY LOCATION" "CAPACITY (M
DATA "PITTSBURG" 92 85 3 20
```

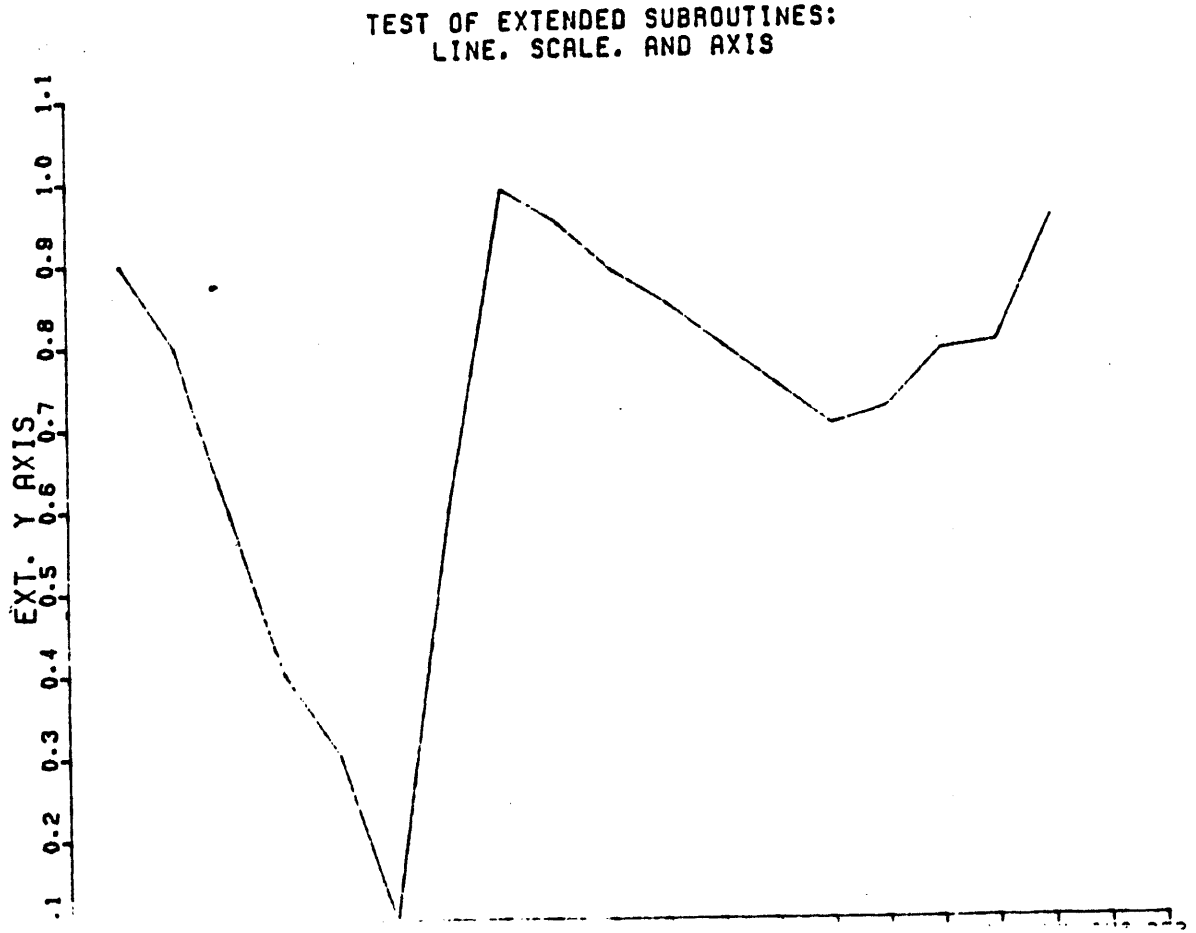
```
DATA "TRENTON" 60 23 2 20
DATA "MIAMI" 54 51 2 20
DATA "ATLANTA" 85 85 3 20
DATA "BATON ROUGE" 72 42 2 20
DATA "HOUSTON" 98 95 3 20
DATA "SAN DIEGO" 23 21 2 20
DATA "SAN FRANCISCO"87 80 2 20
DATA "PORTLAND" 95 45 3 20
END
SMBOLC 7.5 4.5 .12"WAREHOUSE FACILITY REPORT" 0 90
SMBOLC 7.5 4.3 .09"SHADED PORTION INDICATES FILLED CAPACITY" 0 90
SMBOLC 7.5 4.15 .09 "DENSE PORTION SHOWS SECURITY-SENSITIVE PRODUCT" 0 9
SMBOLC 7.5 3.9 .085 "REFLECTS CAPACITY NOTED IN JANUARY, 1981 REPORT" 0
EXIT

****** TEST FOR PIE CHART  "AJAXPIE.CMD"  ******

PLOTS TYPE 4
PICT 0 12 -3 END
SMBOLX 5 7.5 .25 "COST BREAKDOWN - AJAX WIDGET CO." 0 90 1
SMBOLX 5 7.2 .12 'DEMONSTRATION OF "GRAPH" ROUTINE' 0 90 1
CHART PIE 5 3.75 6
DATA "OVERHEAD" 25.6
DATA "LABOR" 16.2
DATA -1 "MATERIAL" 11.5
DATA "MACHINES" 10.2
DATA "GAS" 1.0
DATA "TOOLS" 8.4
DATA "POWER" 5.3
DATA "LEGAL" 5.5
DATA "UNION" .9
DATA "INSUR." 9.1
DATA "OTHER" 4.3
END
EXIT

****** TEST FOR NUMBERS AND SYMBOLS "TESTNMSY.CMD" ******

PLOTS TYPE 4
FACTOR .9
PICT 0 0 3 0 4 2 3 4 2 0 4 2 0 6 2 4 0 2 4 0 2 0 0 2 END
SMBOLC 3.9 4 .25 'TEST OF SYMBOL & NUMBER' 90 90
SYMBOL .2 0 .12 'STANDARD SYMBOL CALL' 90 20
SMBOLL .4 0 .12 'SMBOLL CALL (LEFT JUSTIFY)' 90 26
SMBOLC .6 4 .12 'SMBOLC CALL (CENTER)' 90 20
SMBOLR .8 8 .12 'SMBOLR CALL (RIGHT JUSTIFY)' 90 27
NUMBER 1.5 .1 .12 12.345 90 3
NUMBER 1.5 3  .12 12.345 90 2
NUMBER 1.5 6  .12 12.345 90 1
NUMBER 1.7 .1 .12 12.345 90 0
NUMBER 1.7 3  .12 12.345 90 -1
NUMBER 1.7 6  .12 12.345 90 -2
NUMBRL 2.2 0 .12  12.345 90 3
```

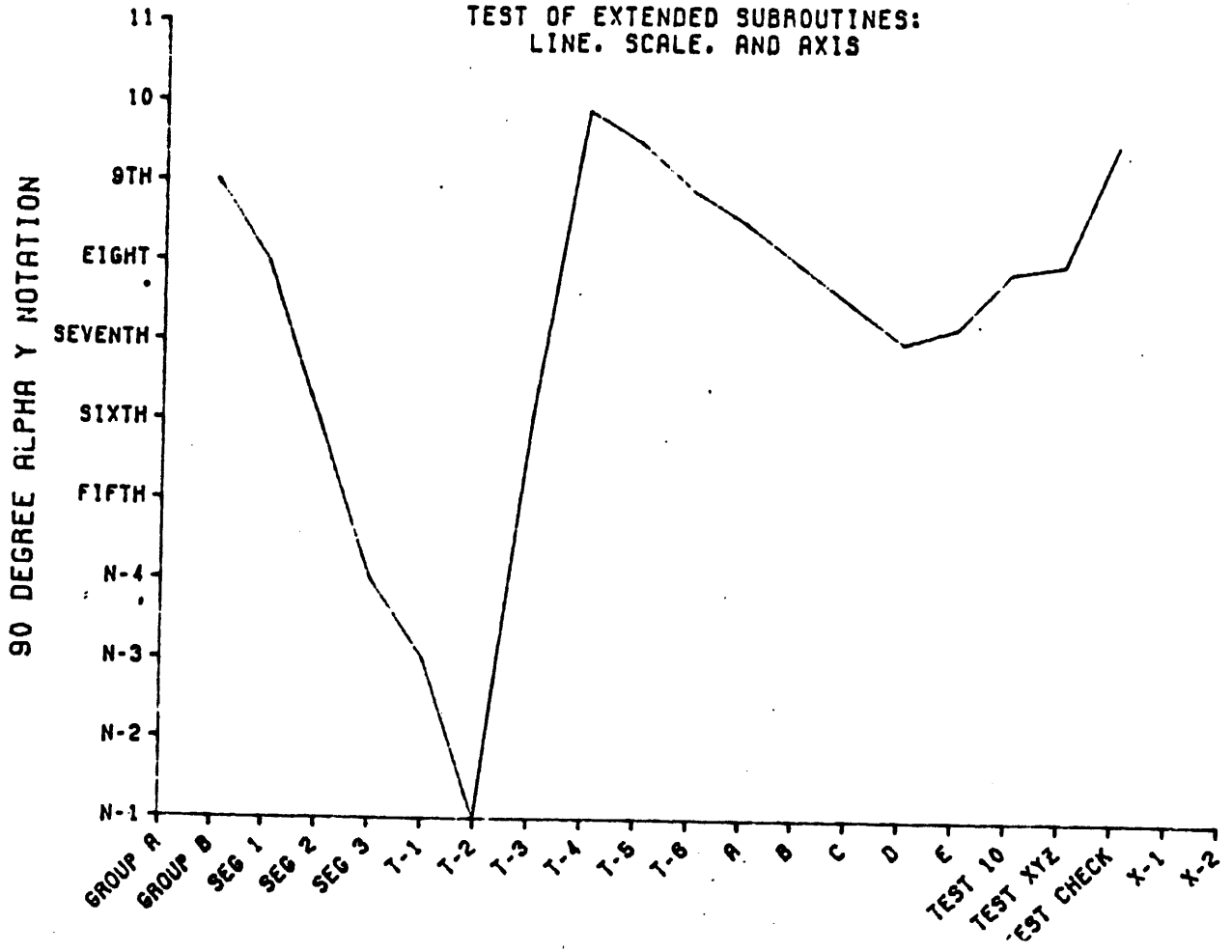
```
NUMBRC 2.2 4 .12  12.345 90 3
NUMBRB 2.2 8 .12  12.345 90 3
NUMBER 2.6 0 .12  55.555 90 2
SMBOLC 8 4 .25 'TEST OF ROTATION' 90 99
SYMBCL 5.5 2.75 .14 '.ROTATE.' 0 99
SYMBOL 999. 999. .14 '.ROTATE.' 45 99
SYMBOL 999. 999. .14 '.ROTATE.' 90 99
SYMBCL 999. 999. .14 '.ROTATE.'135 99
SYMBOL 999. 999. .14 '.ROTATE.'180 99
SYMBOL 999. 999. .14 '.ROTATE.'225 99
SYMBCL 999. 999. .14 '.ROTATE.'270 99
SYMBOL 999. 999. .14 '.ROTATE.'315 99
SMBOLC 8.2 4 .15 'CREATED BY "TESTNMSY.CMD"' 90 99
EXIT
```

TEST OF STANDARD SUBROUTINES:
LINE, SCALE, AND AXIS

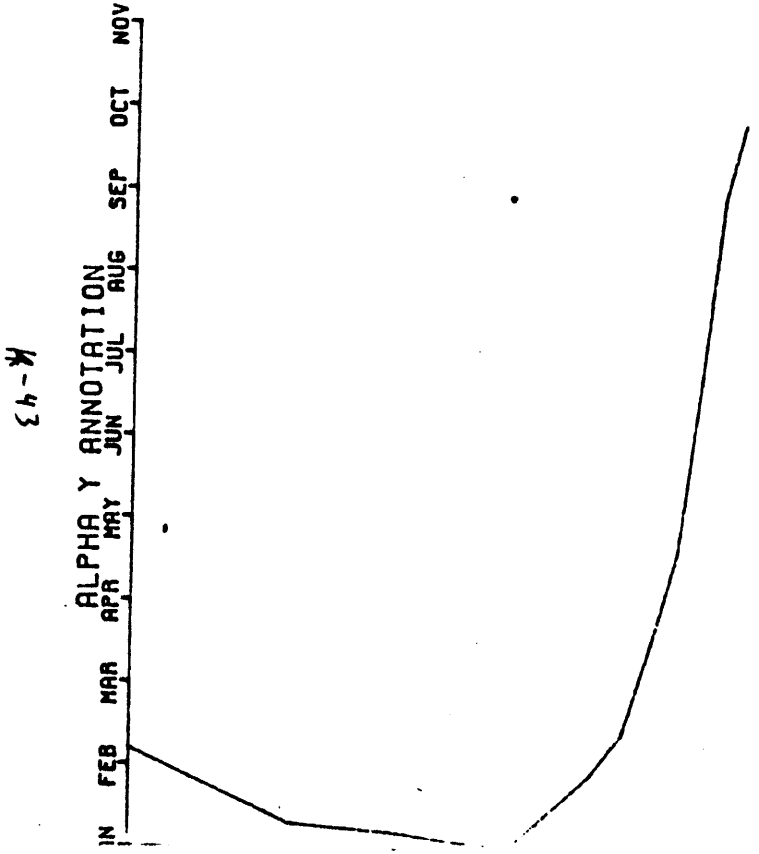



TEST OF EXTENDED SUBROUTINES:
LINE. SCALE. AND AXIS
EXT. Y AXIS
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
1.1

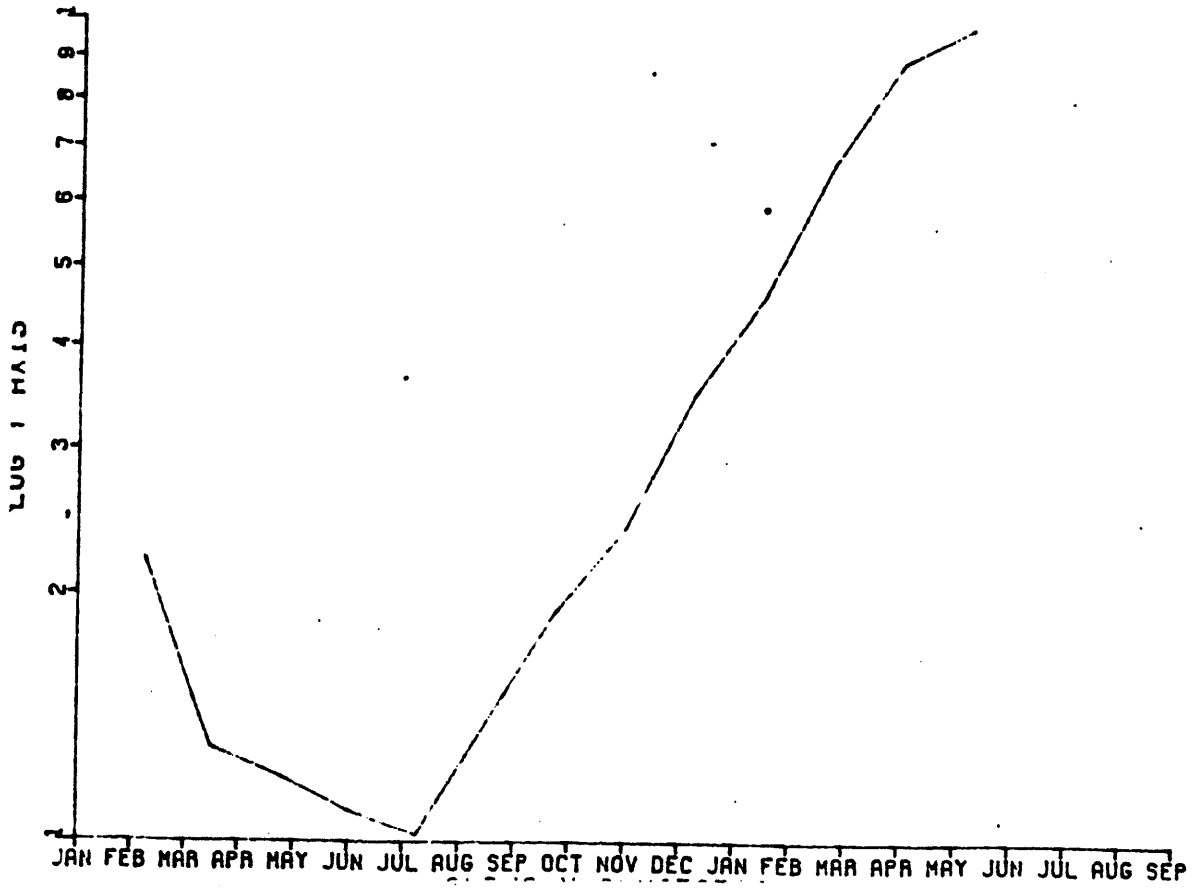

TEST OF EXTENDED SUBROUTINES:
LINE. SCALE. AND AXIS
EXT. Y AXIS
.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.8
1.0
1.1

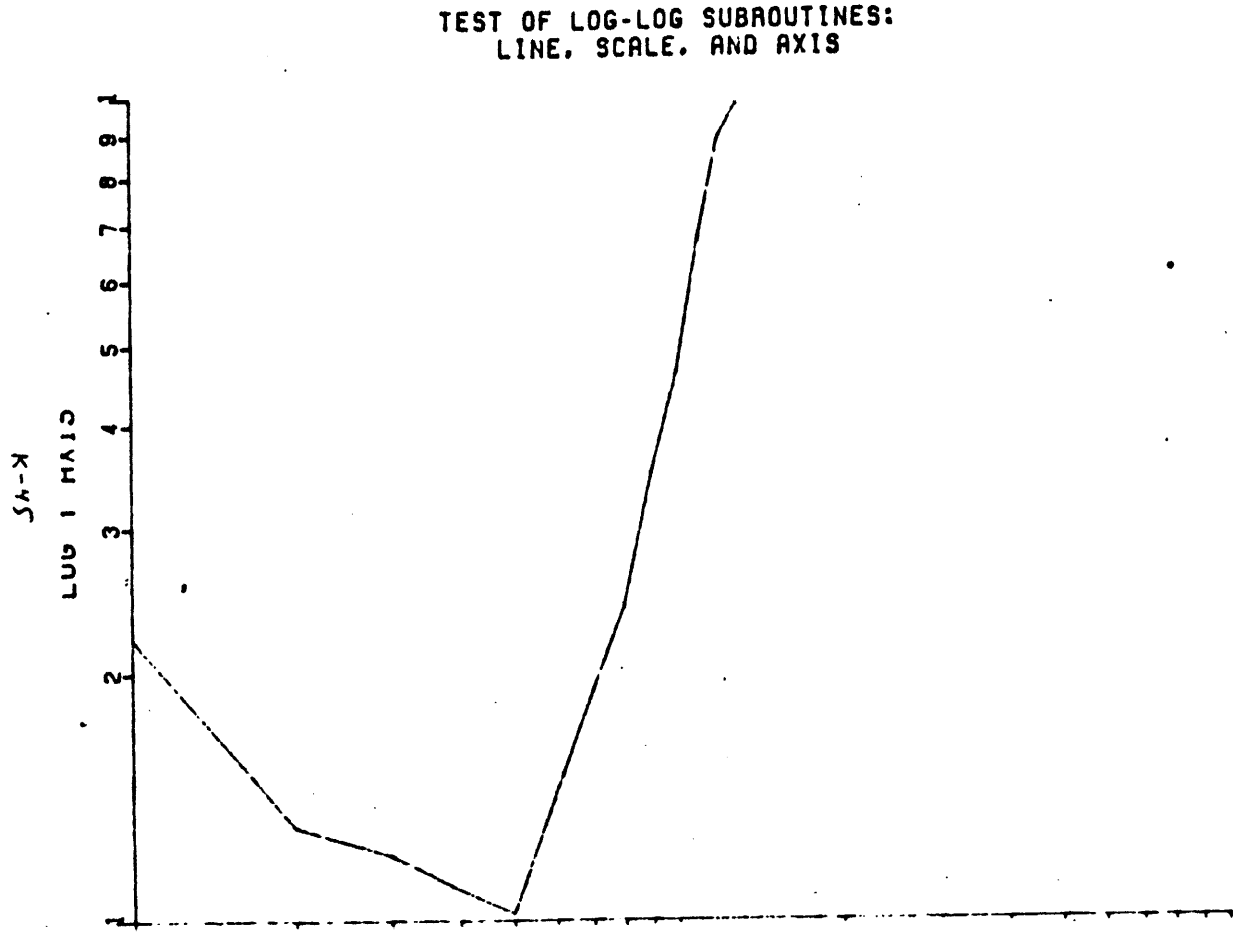

TEST OF EXTENDED SUBROUTINES:
LINE. SCALE. AND AXIS
90 DEGREE ALPHA Y NOTATION
11
10
9TH
EIGHT
SEVENTH
SIXTH
FIFTH
N-4
N-3
N-2
N-1
GROUP A
GROUP B
SEG 1
SEG 2
SEG 3
T-1
T-2
T-3
T-4
T-5
T-6
A
B
C
D
E
TEST 10
TEST XYZ
EST CHECK
X-1
X-2

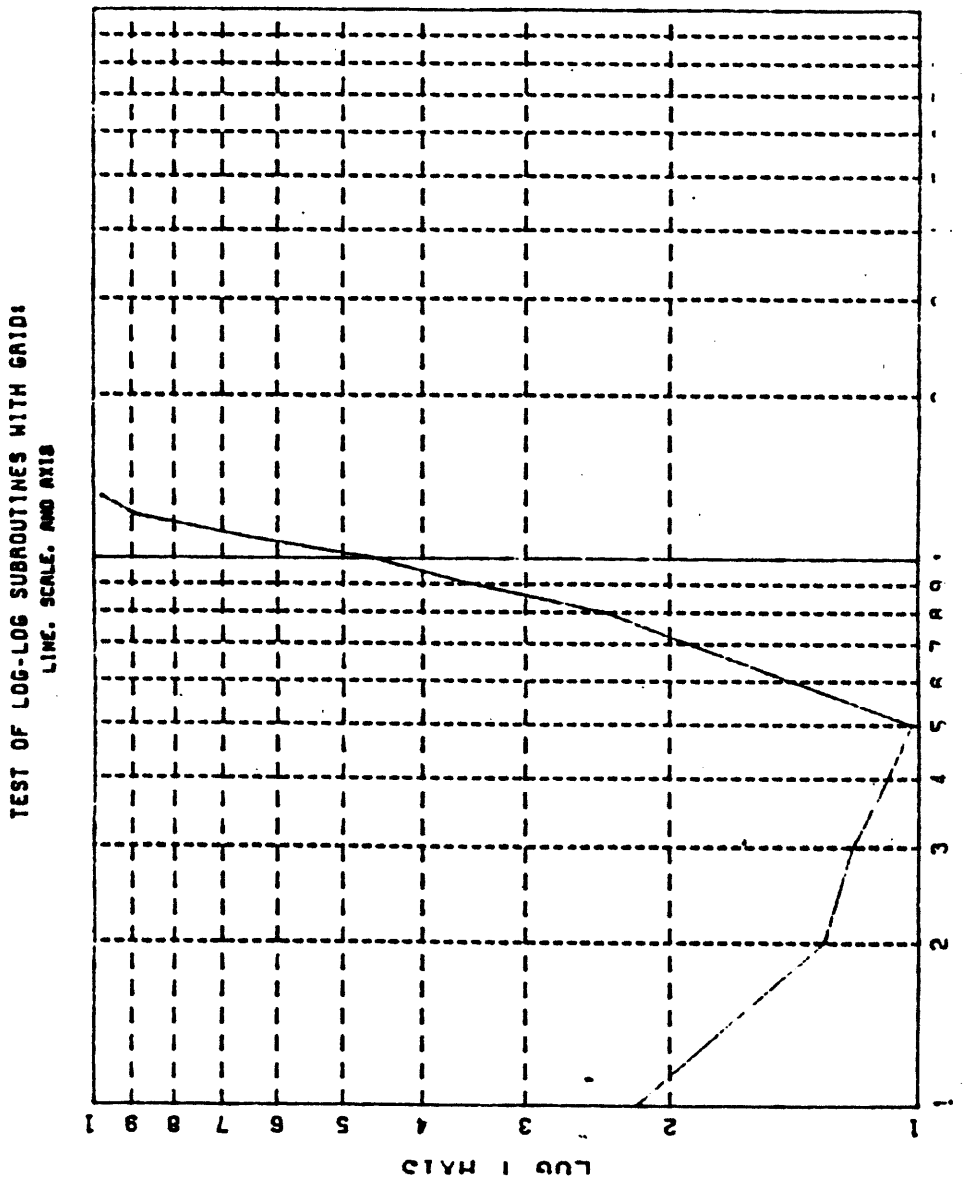
TEST OF EXTENDED AND LOG SUBROUTINES:
LINE. SCALE. AND AXIS



TEST OF EXTENDED AND LOG SUBROUTINES:
LINE. SCALE. AND AXIS

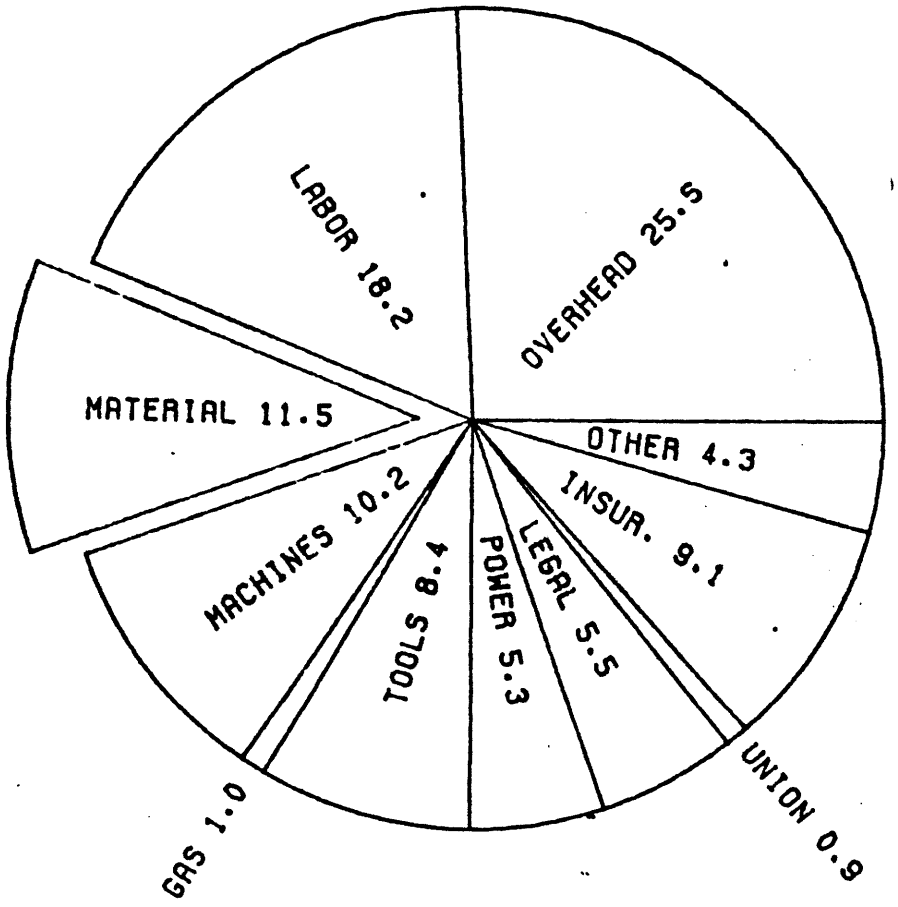



TEST OF LOG-LOG SUBROUTINES:
LINE, SCALE, AND AXIS
LOG (X-45)
1
2
3
4
5
6
7
8
9
1


TEST OF LOG-LOG SUBROUTINES WITH GRID
LINE. SCALE. AND AXIS
LOG 1 RATIO
1 2 3 4 5 6 7 8 9 1
1 2 3 4 5 6 7 8 9 1

COST BREAKDOWN - AJAX WIDGET (

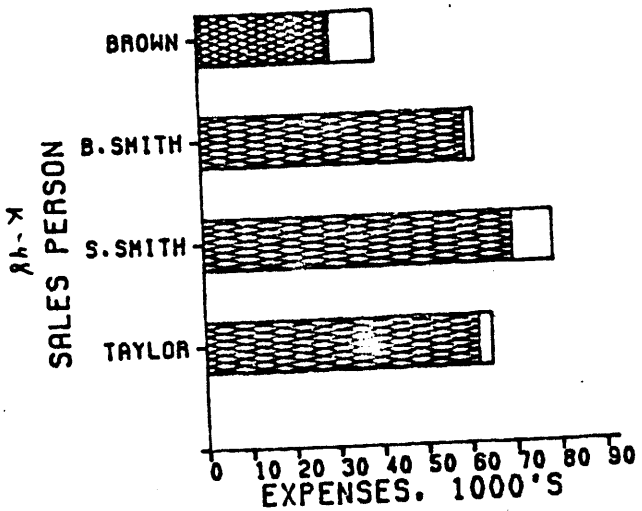
DEMONSTRATION OF "GRAPH" ROUTINE



APPENDIX B - ANNUAL REPORT

EXPENSE REPORT BY INDIVIDUAL

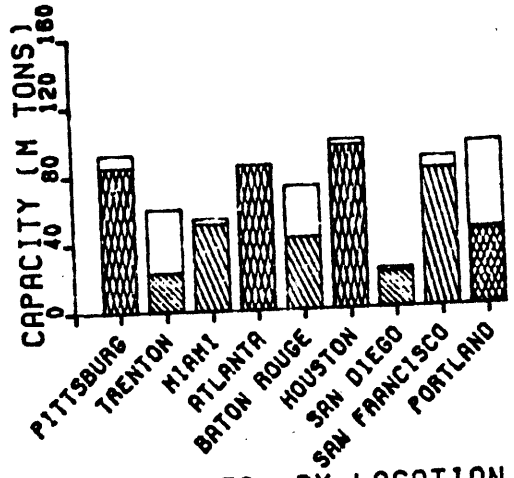
SHADED PORTION IS TRAVEL-RELATED EXPENSE
FOR THE PERIOD 1-80 THRU 12-80



WAREHOUSE FACILITY REPORT

SHADED PORTION INDICATES FILLED CAPACITY
DENSE PORTION SHOWS SECURITY-SENSITIVE PRODUCT

REFLECTS CAPACITY NOTED IN JANUARY, 1981 REPORT



STANDARD SYMBOL CALL
SMBOLL CALL (LEFT JUSTIFY)

12.345
12.

12.345

55.56

SMBOLC CALL (CENTER)

12.35
12

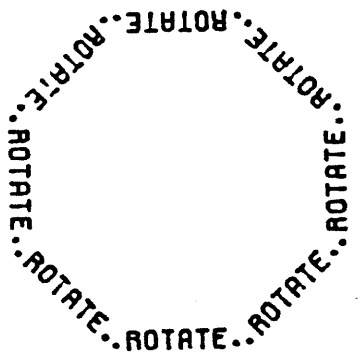
12.345

SMBOLR CALL (RIGHT JUSTIFY

12.3
1

12.34

TEST OF SYMBOL & NUMBER



TEST OF ROTATION
CREATED BY "TESTNMSY.CMD"

STANDARD CHARACTERS
LISTED WITH HEX AND DECIMAL EQUIVALENTS

00 0	30 32	40 0 48	50 @ 64	60 P 80
01  1	31 ! 33	41 1 49	51 A 65	61 Q 81
02 2	32 " 34	42 2 50	52 B 66	62 R 82
03 3	33 # 35	43 3 51	53 C 67	63 S 83
04  4	34 $ 36	44 4 52	54 D 68	64 T 84
05  5	35 % 37	45 5 53	55 E 69	65 U 85
06  6	36 & 38	46 6 54	56 F 70	66 V 86
07  7	37 ' 39	47 7 55	57 G 71	67 W 87
08  8	38 (40	48 8 56	58 H 72	68 X 88
09  9	39) 41	49 9 57	59 I 73	69 Y 89
0A  10	3A * 42	4A : 58	5A J 74	6A Z 90
0B  11	3B + 43	4B ; 59	5B K 75	6B [91
0C  12	3C , 44	4C < 60	5C L 76	6C \ 92
0D 13	3D - 45	4D = 61	5D M 77	6D] 93
	3E . 46	4E > 62	5E N 78	6E ^ 94
	3F / 47	4F ? 63	5F O 79	6F _ 95