CONF-781159--2

A GENERALIZED PLOTTING FACILITY*

R. D. Burris and W. H. Gray
Oak Ridge National Laboratory
Oak Ridge, Tennessee

ABSTRACT

A command which causes the translation of any supported graphics file format to a format acceptable to any supported device has been implemented on two linked DECsystem-10s. The processing of the command is divided into parsing and translating phases. In the parsing phase, information is extracted from the command and augmented by default data. The results of this phase are saved on disk and the appropriate translating routine is invoked. Twenty-eight translating programs have been implemented in this system. They support four different graphics file formats, including the DISSPLA and Calcomp formats, and seven different types of plotters, including Tektronix, Calcomp, and Versatec devices. Some of the plotters are devices linked to the DECsystem-10s and some are driven by IBM System/360 computers linked via a communications network to the DECsystem-10s. The user of this facility can use any of the supported packages to create a file of graphics data, preview the file on an online scope, and, when satisfied, cause the same data to be plotted on a hard copy device. All of the actions utilize a single simple command format.

INTRODUCTION

1.1 General

In the last few years the advances in computer hardware and software have been impressive. For the most part such advances have been beneficial, but the problem of "future shock" exists in computer science as well as in the rest of our culture. The problem has been perhaps more severe in the specific area of graphics, where graphics support packages and plotting devices are legion.

1.2 Environment

The system we describe was developed for DECsystem-10s, of which our installation has two. They are linked to one another and to several IBM System/360s by communications lines and software. We are concerned in this paper with timesharing users of the DECsystem-10s.

Several types of plotters are available: Tektronix graphics terminals, Versatec printer/plotters, Calcomp pen-and-ink flatbed plotters, and a FR80 COM device with graphics capability. Over the last 15 years many graphics support packages for various plotters have been written or acquired by our installation, so that now there are about 50 graphics support packages available.

1.3 Problem Description

Consider the following set of basic decisions which must be made before a graphics application is designed and developed.

1. The designer must decide upon the plotter to be used in production.
2. The implementer must decide upon the plotter to be used in debugging.
3. The implementer must choose a graphics software package which
   (a) is appropriate to the solution of the problem,
   (b) supports the production plotter, and
   (c) supports the debugging plotter.

Some problems arise in making these decisions.

1. The implementer must understand and support the interface between the graphics package and the plotter.
2. The implementer should understand all packages and interfaces so that accurate decisions are possible.
3. Everyone must consider the interface between the user and the product — if it is difficult to use, no one will use it.

In an environment where so many options are available, these problems are immense.

1.4 Problem Solution

If a system were available which would translate a given file or graphics data to a format consistent with any available plotter, and if such support were available for all supported graphic files, all the problems would be solved.

1. The designer and implementer no longer need to choose the plotter, since all applications are device-independent.
2. The implementer need not consider the chosen plotter while deciding upon the support package — the package most appropriate to the problem may be used.
3. The interface between the support package and the plotter is already done.
4. There is only one user interface.

*Research sponsored by the Office of Fusion Energy (ETM), U.S. Department of Energy, under contract W-7405-eng-26 with the Union Carbide Corporation.

Note that in such a system, the interfaces must be well done since users of all levels of expertise, from rankest greenhorn to seasoned pro, will be using them.

1.5 Purpose of This Paper

The system described in this paper is an implementation of the concept set forth in Section 1.4. At this time four graphics file formats and seven types of plotters are supported, with all 28 possible combinations of those elements implemented. A simple, flexible, powerful, and extremely forgiving interface has been developed. We discuss the design parameters for this system and the structure that evolved for their implementation, then describe the system components, and finally present scenarios of the use of the system by novice and experienced users.

2. SYSTEM DESCRIPTION

2.1 General

We have been describing some of the problems facing users to emphasize the primary design goal of this facility, the provision of a device-independent, user-oriented plotting facility. We wanted to provide a user interface which would not require much learning, would be essentially static over long periods of time, would require a minimum of user input, and would permit corrections without repeating the entire request.

2.2 Design Parameters

The generalized plotting system was designed in accordance with these specifications:

1. The system must be capable of handling several graphics file formats and several plotters.
2. The system must be easily and consistently expandable to new file formats and devices.
3. The user's view of the system should be static except for additions supporting new facilities.
4. A single command should be able to direct the conversion of any supported graphics file to a format compatible with any supported device.
5. The command should have a syntax familiar to the user.
6. The command must be easy to use.
7. The command processor must be exceptionally forgiving of user errors.
8. As much existing software as possible should be used, to minimize development time.
9. The system must be easy to maintain.

2.3 System Structure

The processing of a user command was seen to consist of two phases: the inspection of the command for validity and the translation of the specified data to the required plotter format. The system was therefore divided into two parts: the command scanner (the user interface) and the translators. Since many translation programs already existed, they were to be used to the greatest extent possible. In fact, the only modification to the translators presently implemented has been to cause the translator to look in a disk file for the name of the graphics file.

In general, the command scanner processing consists of the following steps (see Figure 1):

1. Acquire the command to be inspected.
2. Parse the command into its constituent fields.
3. Verify the existence of the required translator.
4. Acquire system default information.
5. Acquire default information for the required translator.
6. Acquire and apply user defaults (SWITCH.INI file).
7. Prompt for any omitted mandatory information.
8. Prompt for any needed corrections.
9. Build a parameter file for the translator.
10. Invoke the translator.

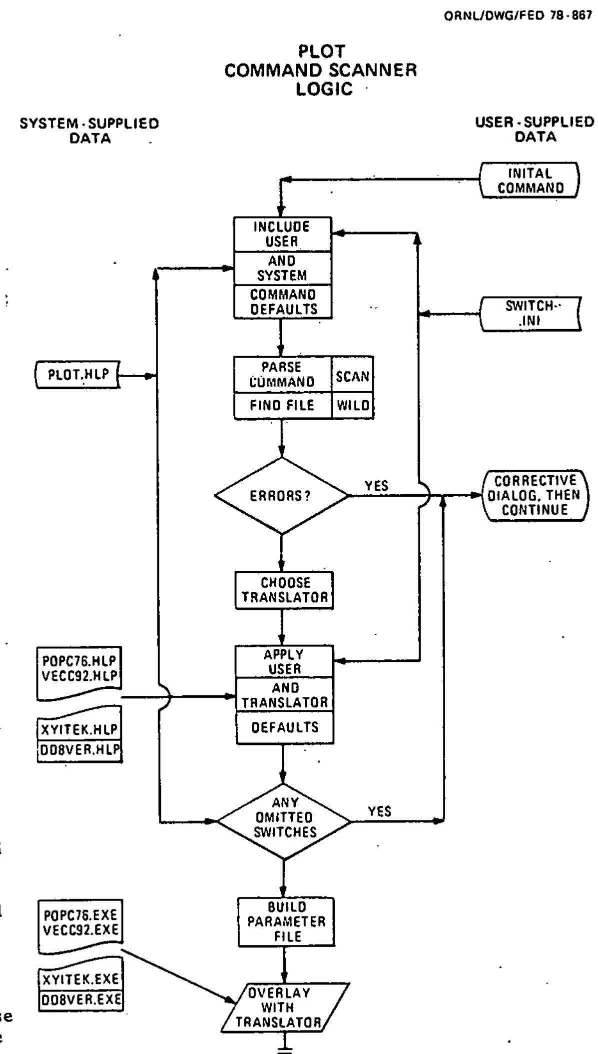

Fig. 1

3. USER INTERFACE

When a system provides great power and flexibility, so many options may be available that the system could be impossible to use. Consequently, a great deal of attention was placed upon the structuring of the system for ease of use. Among the steps taken were the specification of a simple, familiar command syntax; the definition of many defaults; and the corrections of many errors.

3.1 Command Syntax

Since the facility was intended for installation on a DECsystem-10, a syntax appropriate to that computer was desired. The Peripheral Interchange Program (PIP) syntax was chosen as being the most familiar to the user. Specifically, the syntax was

```
.PLOT PLTDEV:=DEV:FILE.EXT/SWITCH,DEV:FILE.EXT/SW
/SW,...
```

where
PLTDEV is the destination plotter,
DEV is the device upon which the source file is found,
FILE is the name of the graphics file,
EXT is the type of data (i.e., Calcomp, Tektronix, vectors, etc.), and
/SWITCH is additional information about the file or the translation to be done.

To maintain maximum compatibility with DEC software, the SCAN and WILD programs were used to process the command string.

3.2 Defaulting

The user of a facility on a computer system rarely knows all of the available opticns. In a facility as extensive as this one, the number of options is very large, so the user could find effective functioning very difficult unless the system is carefully designed. One way of aiding the user is an extensive set of default values.

Defaults to two types of fields are provided. The first type is called the command field and includes device, file name, and extension. The second type of field is the switch.

Defaults are also provided in a variety of ways. There are defaults provided by the system, by the user in a SWITCH.INI file, and by the user while invoking the command.

3.2.1 Command Defaults — There are two sources of default information for the command field defaults — the system and the user. If the user types

```
.PLOT
```

the system will interpret the command to mean, for instance,

```
.PLOT VER:=SEGMNT.VEC
```

These defaults are a function of the computer upon which the generalized plot facility is installed and are contained in a file known to the command scanner.

It often happens, however, that devices or file formats are used in bursts. For instance, users will perform debugging using Tektronix terminals to take advantage of quick turnaround and would thus like a different default plot device to obviate typing

```
.PLOT TEK:=
```

every time. Accordingly, the switches DEVICE, FILE, and EXTENSION have been implemented for inclusion in the SWITCH.INI file. If the user desiring extensive use of the TEK device includes the line

```
PLOT/DEVICE:TEK
```

in the SWITCH.INI file, the results of

```
.PLOT
```

are now

```
.PLOT TEK:=SEGMNT.VEC
```

Of course, if the user explicitly specifies some field, that specification overrides any default, just as the SWITCH.INI specifications override the system defaults.

3.2.2 Switch Defaults — As in the case of command defaults, there are several sources of default information for switches. Again, there are system-supplied defaults, but now the defaults are specific to each translator, rather than to the system as a whole. These defaults are kept in files known to the command scanner and are applied once the translator has been chosen. In addition to the specification of values for switches, these files contain information about the switches required for the translator and switches meaningful to it.

The user may also specify default values by including information in the file SWITCH.INI. Any switch may be included in the SWITCH.INI file. If the same switch has a system default, the SWITCH.INI value takes precedence.

A third means of specifying defaults is by sticky defaults. That is, in a command which will plot several files, switches common to all files may be specified before the first file name and will then be applied to all subsequent files until overridden explicitly. For instance,

```
.PLOT =/COPIES:3 A.VEC,B.VEC
```

will plot three copies each of A.VEC and B.VEC. The command

```
.PLOT =/COP:3 A.VEC,B.VEC,C.VEC/COPIES:4
```

will plot three copies of A.VEC and B.VEC, but four copies of C.VEC.

Explicit specifications of a value by a user take precedence over all forms of defaults. Sticky defaults take precedence over SWITCH.INI defaults, which in turn override system defaults.

3.2.3 Unspecified Values — As mentioned above, there is information available to the command scanner concerning switches which must be specified for some translators. If there is no source of information for such a switch, including any form of default, the user will be prompted for that value.

3.3 Overall Logic

Figure 2 presents the overall logic of the generalized plot system, including the flow from the command scanner through the translators to the graphics device. Note that in several places in the diagram there are entries of "NETWORK QUEUE". These represent the transmission of the file to some other computer in the extensive network of which our DECsystem-10s are a part. For instance, some portion of the work for all the Calcomp plotters is done on IBM System/360s.

3.4 Error Correction

Few events are more frustrating to a user of a timesharing system than typing a long command string and having the operating system throw it all away because the user misspelled a switch. This problem is highlighted in this facility since there are so many switches which can be used. To alleviate this problem, the user is told of the mistake made and prompted for correct data. Among the errors which are correctable are:

1. null device,
2. wildcard device,
3. null switch,
4. unknown switch (name misspelled),
5. ambiguous switch (name misspelled or too abbreviated),
6. omitted switch value,
7. ambiguous switch value (keyword misspelled),
8. ambiguous switch value (keyword misspelled or too abbreviated),
9. zero or excessive length project or programmer number, and
10. no file satisfying a wildcard specification.

4. TRANSLATORS

4.1 General

The discussion so far has considered the user-software interface. That interface makes known to the plotting system a set of information describing the work to be done and the program which is to do that work. The remaining functions

ORNL/DWG/FED 78-866

PLOT SYSTEM LOGIC

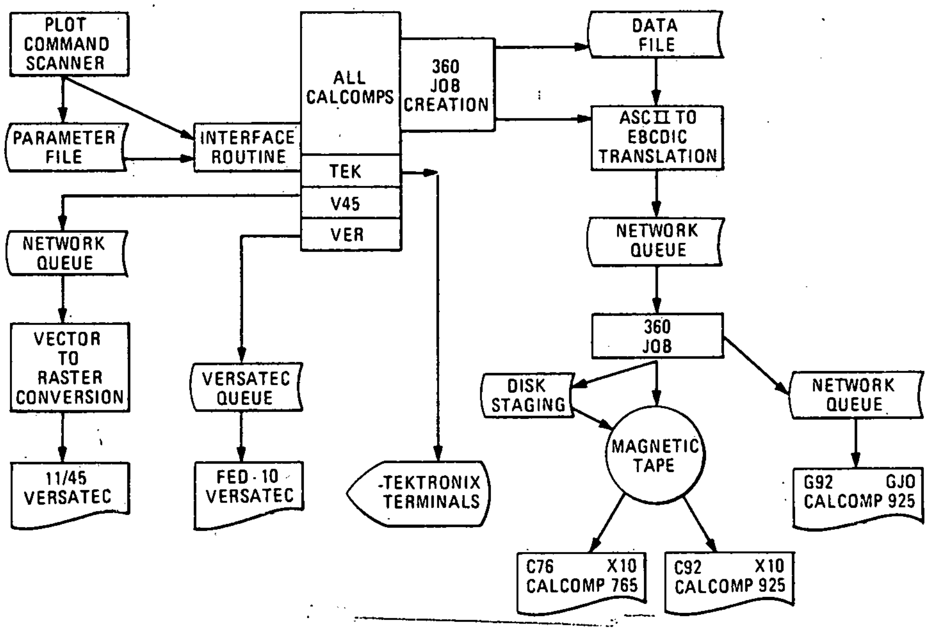

Fig. 2

to be performed are to save the information in a standard manner and to invoke the translation program. Each translator must be able to access the saved information, convert the identified graphics data to the required plotter format, and generate the plotter device access protocol.

The following steps are a simplified list of the flow of a general translator program:

1. Invoke and initialize the translator.
2. Access the information about the intermediate graphics file and specified switches.
3. Open a graphics data file.
4. Input a graphics data file buffer.
5. Decode the packed information into graphics file primitive information.
6. Translate graphics file instruction into device instructions.
7. Encode the device driving instructions into the device primitive format.
8. Output to device.
9. Repeat steps 4-8 until graphics file is exhausted.
10. Close graphics file and device.
11. Check for completion of job step and terminate or return to step 2.

4.2 Scanner-Translator Interface

There are three elements to the scanner-translator interface. First, the information identifying the graphics data and the required conversion must be saved in a format accessible to the translator. To do this a disk file is created, the name of which contains the number of the timesharing job assigned to the user doing the work, thus providing uniqueness to the disk file name. The specifiers are saved in the following order:

1. switches applicable to all graphics files in the request (global switches),
2. file specifications, including each of the remaining elements,
3. file structure,
4. file name,
5. extension,
6. PPN, and
7. switches relevant only to this request (file switches).

The entries are saved in a format consistent with their nature. File structures, names, and extensions are saved in SIXBIT, the PPNs are binary, and the format of the switch values depends upon their nature. The global switches are all ASCII, while most of the remaining switches take binary values. The global switches have positional values — the first six words of the parameter file contain the value of the NAME switch, the next ten contain the ADDRESS values, etc. The file switches are saved in the format:

1. SIXBIT switch name (truncated to six characters),
2. word length of the switch value (in binary), and
3. switch value.

Next, the proper translator must be invoked. The name of the translator is created by concatenating the extension of the file name and the plot device. For example, if the file SEGMNT.VEC is to be plotted on the Versatec, the name of the translator is VECVER, for VECtor file to VERsatec format. Control is passed to that translator by means of the RUN UUO, so that the entire memory space of the command scanner is overlaid by the VECVER translator.

Finally, the translator must be able to access the parameter file. A subroutine capable of this has been written and must be called by each translator. The calling sequence for the parameter access routine is

CALL PLPOST(I,J)

where

PLPOST stands for PLot POSTprocessor,
I is an integer code for the datum desired, and
J is the area in which the value for the datum is to be stored.

The calling program should be written to expect the datum in the format in which it is being passed, which is generally not the same as the format expected by the original translator.

4.3 Tektronix Processing

4.3.1 Translations — The graphics file to Tektronix translators are the least complicated of the programs discussed in this report. A specific decoder is used for each supported graphics file type which unpacks the graphics instructions. These instructions are then translated into synonymous instructions for the Tektronix storage tube. Vendor software, available from Tektronix (1), is used to encode the translated instructions into device primitives which actually control the position of the beam on the face of the Tektronix storage tube.

4.3.2 Plotter interface — The Tektronix storage tubes are treated as teletype ports into the PDP-10. As such, graphic postprocessing done at a terminal of this type is interactive. No spooling of job requests is required.

4.4 Versatec Printer/Plotter Processing

4.4.1 Translations — The graphics file to Versatec plotter translators are the most complicated of the programs discussed in this report. They are typically 7 to 10 times larger in executable module size than corresponding Tektronix translators, due specifically to the vector-to-raster conversion which must be performed. The same set of graphics file decoders is used to unpack individual file types. These instructions are translated into line segments. The first stage of the vector-to-raster conversion process maps all line segments into the raster buffer (which is currently an entire Versatec page 1024 by 800 dots). The second stage, which begins as soon as a new page is requested, compresses the raster buffer in preparation for transmitting this information to the Versatec plotter spooler. The compressed raster data are placed in another file and when all input has been exhausted, program control passes to the I/O interface which queues the compressed raster data to the Versatec plotter spooler.

4.4.2 Plotter Interface — At one of our installations a Versatec printer/plotter is used as the only hard copy output device, serving as both line printer and hard copy plotter. Two queues have been defined to serve that device, each queue having its own spooler. Given the queue for plot work and the spooler to handle that work, all that the translator needs to do is to enter its output in that queue. The rest of the processing is handled by the system.

At the same installation a second computer (a DEC PDP-11/45) is available with its own Versatec printer/plotter. If that Versatec is specified, the untranslated file is transmitted to the PDP-11/45 and converted to raster image there, thus providing much faster operation and reduced load on the DECsystem-10.

4.5 Calcomp Processing

4.5.1 Translations — The graphics file to Calcomp plotter translators are based upon a slightly different principle. Actually, these translators create a DISSPLA (2) compressed graphics data set from the input graphics data file. This scheme was chosen since a program which would transmit a DISSPLA compressed graphics file to the IBM System/360 and a program which would plot the same file on the remote Calcomps already existed (although not in a convenient form). Therefore, in keeping with the modest software development of the generalized plotting facility, the same set of graphics file decoders was used to unpack the data files. These graphics instructions are translated into line segments and "drawn" into the DISSPLA compressed graphics data set.

4.5.2 Plotter Interface — The Calcomp Plotters at our installation are both stand-alone plotters. That is, they are not physically connected to any computer. To do plotting the user puts the data on a magnetic tape which then is transported to the plotter. Because of various considerations, including the availability of operators, tapes, and software, all tapes for the Calcomp plotters are created by IBM System/360 computers. Since much of the plot development work is done or saved on the DECsystem-10, the generalized plotting system must be capable of generating a job to run on the IBM System/360s which will create a tape for the Calcomp plotters. Since the IBM System/360s are linked by communications lines and software to the DECsystem-10, the plot system can then enter the generated job into a queue for the IBM System/360s and let the system do the rest of the work.

The primary tenets of the generalized plot system apply to the job generating code as well as to the rest of the system. The user should be insulated from unnecessary detail and should be provided with as many defaults as possible. To that end, only a few mandatory data are defined and SWITCH.INI support for those data is provided (so that the user may define those data once and then forget about them). The required data are the user identifier (a 3-character field assigned administratively to provide job name uniqueness on IBM System/360 work), the charge number (which is used in IBM System/360 accounting and is a standard field), and the address of the user (which is used in routing output).

Given the above information, two general methods of providing a tape for the Calcomp plotters are available. For jobs doing relatively little plotting, a high turnaround method is available.in which the graphics file is created with a standard name upon IBM System/360 direct access storage on a disk pack of a given name. At regular intervals the operators of the IBM System/360s invoke a program which searches that disk pack for all files of standard name construction and places their contents on a single tape for transport to the Calcomps.

Jobs creating large graphics files must create their own tapes in the interests of saving on-line direct access space. The generalized plot system, then, needs to build a different job, one which calls for the mounting of a tape and which causes the creation of a plot control card to accompany the tape to the plotter.

The creation of each type of job under the generalized plotting system obeys the same rules as the rest of the system — the user does not have to know what is going on. In fact, the user will never see the job control language until the output of the job is delivered.

It is important to note that the plotting system exists on two DECsystem-10s about six miles apart, which are linked by communications lines and spoolers so that either one may be used with the plot system with no modification of the procedure. If the user is connected to the remote DECsystem-10, an extra routine step is interposed (without the user's knowledge) which sends the generated job to the DECsystem-10 in the room next to the IBM System/360s, from which the job is entered into the IBM System/360 queue. The user cannot tell from the job output which computer was used to initiate the job.

5. SCENARIOS

5.1 General

Let us now consider the appearance of the generalized plot system to the user. We consider two users — the novice and the old pro. The novice is defined as a person who has no previous experience with the plot system, but who does know how to create a graphics file and has had enough experience with DECsystem-10 to know that HELP files exist for many facilities and that often a /H switch will provide assistance. The old pro has been using the plot system for a long time and knows about all the defaults and short cuts possible.

5.2 Novice

We begin with the novice user who has already created a graphics file, which we call SEGMNT.VEC. The user first types

.HELP PLOT (where PLOT is the command invoking the system)

and receives some general information about the system as a whole. Included in this file are some basic commands chosen to provide the user with some inkling of the range and depth of the command structure. One of the commands given as an example is

.PLOT TEK:=SEGMNT.VEC

which will cause the user's file to be plotted on the user's Tektronix scope. Cheerfully, the user enters that command and finds that the resultant plot is not as intended.

Having found errors in the program and used the Tektronix plotting capability of the system to verify that the file is correct, the user now desires a hard copy of the graph. Hoping that a command like

.PLOT/HELP

will provide helpful information, as it does for many facilities on the DECsystem-10, the user tries it. The response is a list of categories of information available from which the user is asked to choose. Interested in finding out what plotters are available, the user responds appropriately and finds that a VECC92 processor is available (vector file to Calcomp 925/1036 plotter). (The HELP facility then prompts for the next category of information desired.) Continuing the dialog, the user finds that several switches are mandatory or recognized but not being in the mood to figure them all out simply types the command:

.PLOT C92:=SEGMNT.VEC

which the user hopes will take care of all problems and plot the file on the Calcomp 925/1036 plotter.

Unfortunately, further information is required before the plot can be made, since an IBM System/360 job must be created to do some of the translation. The user is then prompted for a user identifier, a charge number, and an address. The user is then pleased to find that the request is complete and the work is scheduled.

Flushed with success, the user decides to plot another file, one named FILE.ABC. Entering the command

.PLOT C92:=FILE.ABC

the user is disappointed to note that no translator ABCC92 exists and that prompts for correct device and file extension have been issued. Knowing that the information was correctly entered and knowing that the file is really in the DISSPLA POP format the user renames the file to FILE.POP and types

.PLOT C92:=FILL.PPP

misspelling grievously. The plot system announces that no such file exists and prompts for correction. When the user responds correctly, the system again prompts for user identifier, charge, and address before scheduling the work.

5.3 Old Pro — Our old pro knows all about the generalized plot system. Long ago this user tired of responding to prompts for switches and built a SWITCH.INI file containing the line

PLOT/DEVICE:TEK/UID:RDB/ADDRESS:"9201-2 Y-12"/CHARGE:12345

so that the IBM System/360 jobs required for various applications could be easily built. Furthermore, this user knows about the default file names and devices for each installation, so to plot a SEGMNT.VEC file on a Tektronix scope this user types only

.PLOT

The device field and mandatory switch values will be provided by the SWITCH.INI field while the file name and extension fields will be provided by the system defaults.

When all the bugs are out of the application and the old pro wants a Calcomp plot, the SWITCH.INI file is modified to contain /DEV:C92 so that the command

.PLOT

takes care of it with no prompting. The old pro also knows about the copies and forms capabilities of the system and about the multiple pens available on the Calcomp 925, so when desiring to take advantage of these features the pro types

.PLOT =A.VEC/COPIES:3/FORMS:602/INK:(GREEN/L,BLACK,RED/L)

which causes the file A.VEC to be plotted three times on 602-type forms with the three pens set to liquid green ink, black ballpoint pen, and liquid red ink, respectively.

6. SYSTEMS MAINTENANCE

In a system as complex as this one, considerable effort should be directed to making the system easy to maintain and to expand. Part of the means of doing so was to use existing software wherever possible and to make only minimal changes to interface to the new regime. But various new facilities were added, including the automatic application of various translator-specific defaults and the definition of mandatory switches for each translator.

The new facilities are largely supported by means of tables which are kept on disk. One file, the PLOT.HLP file, contains the list of supported translators, mandatory and legal switches for each one, and special information about each translator, as well as the system default specifications for plot device, graphics file name and extension, and the plot command (used when the user types only "PLOT"). In that file the translator names are SIXBIT, the switch fields are in the form of bit maps, and the default command information is ASCII. A program named HELPFI was written to create and maintain this file.

The translator-specific defaults are kept in separate .HLP files with the same names as the translators themselves. That is, the defaults file for the VECVER translator is named VECVER.HLP. This file is created by the same program which maintains the PLOT.HLP file, but it is ASCII. It contains the list of default values in the form of switch strings recognizable by the SCAN program, lists of mandatory and legal switches and of special information, and an ASCII text string providing additional information. Since this file is in ASCII no update provisions are made for it — the system maintainer needs only to use some editor, such as TECO, to modify it.

In accordance with the design specifications for this system, the HELPFI program was designed for ease of use. The maintainer is prompted for each entry and each is validated. The maintainer may create, delete, or modify translator entries.

7. CONCLUSIONS

The generalized plot system described in this paper provides to the user of a DECsystem-10 the capability to use virtually any graphics support package provided with any plotter available. The user is not required to know very much about the system and the most probable user errors are correctable via prompts. The system has been designed for ease of maintenance and expansion, so that minimal support effort is required.

REFERENCES

1. Tektronix Plot 10 Terminal Control User Manual, Tektronix, Inc., Beaverton, Oregon (1976).

2. DISSPLA (Display Integrated Software System and Plotting Language) is a proprietary software library of graphics routines copyrighted by Integrated Software System Corp., San Diego, California.

ACKNOWLEDGEMENTS

Reid Gryder, Kathe Fischer, Betsy Clark, and Charles Thompson all made valuable suggestions concerning the design of this system. Ms. Rose Ann Pemberton prepared the manuscript. Our thanks to them all.