USER-WRITTEN DEVICE DRIVERS

Howard Houston, SAS Institute Inc.

INTRODUCTION

User-written device drivers have the advantage of allowing users to tailor device support to their specific needs. This can take the form of an environment that does not allow the standard Institute support, or a graphics device which is not supported by the Institute. Presently, the only way the users can provide their own support is with the linkable device driver. This serves many users needs quite well, but it is not device intelligent and is designed more for use with a plotter than with a CRT.

In the next release of the SAS/GRAPH® product, there will be a new method for user support of a graphics device. This method will allow users to develop device-intelligent drivers. These device drivers are not linked into the SAS® System. This removes the need to link your device driver with each new system release, and allows this type of device driver to be used with systems that do not allow the linking of external routines with the SAS® System, as is the case with minicomputers.

OVERVIEW OF OPERATION

The basic structure of this new facility is a device driver in the SAS/GRAPH® product which produces a metafile. This is a sequential file containing a device independent description of a graph. The user writes an external driver that translates this metalanguage to commands specific to the graphics device. The internal driver device is made device intelligent by an attribute file provided by the developer of the external driver. This file describes the graphics device and the external driver. The internal driver uses the information in this file to configure itself to the characteristics of the graphics device being used.

DEVICE COMMUNCIATION

The user has the option of communicating directly with the graphics device or allowing the internal driver to handle communications with the device. If the internal driver is used, then the external driver writes the data to be sent to the graphics device in a sequential file. The internal driver will transmit this data to the graphics device when the external driver terminates.

LEVELS OF INTERACTION

The degree of interaction between the internal and the external drivers is controlled by the developer of the external driver. There are three levels of interaction possible between the internal driver and the external driver. At all levels of interaction the external driver can handle the communications with the graphics device, but at certain levels the internal driver can be used to communicate with the graphics device. This frees the developer of the external driver from supporting communications with the graphics device.

The lowest level of interaction is when the external driver is a post-processor. The internal driver only produces the metafile. After the SAS/GRAPH® program is finished, the user must execute the external driver to produce the graphic output. The external driver must support all communications with the graphics device.

The next level of interaction allows use of the internal driver to handle communications with the graphics device. The internal driver produces the metafile for a procedure, and then invokes the external driver at the end of each graphics procedure. The external driver processes the metafile to produce device-dependent data to be transmitted to the graphics device. If the internal driver is used to communicate with the graphics device, this data is written to a sequential file. When the external driver terminates, the internal driver will transmit the data in this file to the graphics device to generate all the graphs produced.

The highest level of interaction invokes the external driver at the end of each individual graph, and allows the internal and external drivers working in tandem to do interactive operations, such as digitization. At this level of interaction, the internal driver may invoke the external driver several times during the same procedure. The transfer of information between the internal and external driver is done in the same manner as the previous level of interaction.

If the internal driver is used to communicate with the graphics device, the internal driver will invoke the external driver twice for each digitizing operation: the first time to produce the graphic input command, and the second time to request the external driver to decode the data received from the graphics device. If the external driver supports direct communication with the graphic device, the internal driver invokes the external driver once to receive

decoded graphics input. In this case, the external driver output file must always be empty whenever digitizing is not in progress.

HEADER AND TRAILER SUPPORT

When there is a need for header and trailer information, as in the case of spooling graphs to a plotter. The user may supply header and trailer files that the internal driver will copy to the metafile at the beginning and end of each graphics procedure. If updated information, such as the time of day, is needed for the header and trailer, the user may supply programs that will write the header and trailer files. The internal driver will invoke these programs to produce the header and trailer files that will then be copied to the metafile. To conserve resources, the same file can be used for the header, the trailer, and the external driver output. In this case, the programs that produce the header and trailer information are required.

ATTRIBUTE FILE

The attribute file is used by the internal driver to configure its operation to the requirements of the graphics device and the external driver. The name of this file is the same as the device name. It contains keyword parameters defining the characteristics of the graphics device, the names of the various files to be used, the host commands with which to execute any external program to be used, and information about communicating with the graphics device.

ATTRIBUTE FILE KEYWORD PARAMETERS

HEADER	Host command to execute the program that writes header file
HEADERFILE	Name of file containing header information
TRAILER	Host command to execute the program that writes the trailer file
TRAILERFILE	Name of file containing trailer information
PROCESS	Host command to execute the external driver
PROCESSINPUT	Name of the input file for the external driver
PROCESSOUTPUT	Name of the output file for the external driver
INTERACTIVE	Level of interaction between the internal and external drivers
NAK	Negative handshake response string
HANDSHAKE	Type of handshake to perform
FORMAT	Data format of the input and output files
ID	Character string to identify this attribute file to the external driver
FOREGROUND	Default foreground color names
BACKGROUND	Name of the default background color
OPTIONS	Defines which options are available in the hardware
SYMBOLS	Defines which symbols are available in the hardware
DASHLINE	Defines which line styles are available in the hardware
PENMOUNTS	Number of pen mounts
RECTANGLEFILL	Defines which fill patterns are available in the hardware
MAXPOLY	Maximum number of vertices per polygon
FILLINC	Increment for drawing a solid fill pattern
VTEXT	Method of drawing vertical text
CHARACTERTYPE	Defines the size of hardware fonts
ROWS	Default number of rows
COLUMNS	Default number of columns
HEIGHT	Height in pixels of the default hardware characters
WIDTH	Width in pixels of the default hardware characters
XSCALE	Horizonal pixels per inch
YSCALE	Vertical pixels per inch
BLKSIZE	Maximum buffer size of the graphics device

EXTERNAL DRIVER FILES AND DATA FORMATS

Binary and character data formats are supported for the files used by the internal and external drivers to communicate. The character format allows the data to be printable characters. This allows easier debugging of the external driver, and may avoid problems in some environments. However, it is not a very efficient means of communicating information between the internal and external drivers. The binary format is a more efficient means of coding the information, but can be more difficult to support. There is a maximum of 80 bytes of data per record with both types of data format. Text strings are always coded as ASCII decimal equivalents.

The external driver input file contains one or more operations per record and the parameters of an operation may span several records. All the operations have a fixed number of parameters which makes parsing of the operations quite easy. The character data format has 16 fields of 5 bytes each. Each field contains integer data in character form. The binary format has 20 fields of 4 bytes each containing integer data in binary.

In the external output file, the first two bytes of each record contain the length of data on the record followed by an operation code for the internal driver. The remaining data on the record are either for transmission to the graphics device or for the internal driver, depending on what operation is to be done. There is one and only one operation per record. The character data format has 25 fields of 3 bytes each containing ASCII decimal equivalents in character form. The binary data format has 76 fields of one byte each containing the ASCII data to be sent to the graphics device.

HEADER AND TRAILER FILES

These files are optional. The first two bytes are the length of data on the record. The remaining bytes are what will be placed into the external driver input file. The the information on these records is not inspected by the internal driver; therefore, the user must insure that bytes 3 to 80 form a valid input record to the external driver.

CONTENTS OF THE METAFILE

This file is the input to the external driver. The contents of this file are a series of operations to be performed by the external driver. The internal driver uses the following operations to describe the actions necessary to produce a graph:

Header	Header information
Trailer	Trailer infomation
Initialize	Initialize graphics device
Terminate	Terminate external driver
Auto	Set autocopy/autofeed option
Shutdown	Shut-down graphics device
Erase	Start a new graph
Prompt	Prompt message
Mount	Pen colors requested
Id	Attribute file identifier
Digitize	Perform graphics input
Decode	Decode graphic input
Move	Set graphic position
Line	Draw line from present position to this location
Arc	Draw an arc
Pie	Draw a solid pie slice
Rectangle	Draw a rectangle filled with the present pattern
Polygon	Draw a solid polygon
Text	Draw a text string at this position
Symbol	Draw a centered symbol at this position
Dot	Draw a dot at this position
Foreground	Set foreground color
Line style	Set line style
Font	Select hardware character set
Orientation	Set text direction and orientation
Color	Define color
Text size	Set hardware character size

Pattern	Select hardware fill pattern
Background	Set background color
Rotate	Rotate graph by 90 degrees

The attribute file controls which operations will be requested by the internal driver. If the attribute file defines the graphics device to not have the ability to perform an operation, such as a polygon fill, the internal driver will not request that operation. The operations not specified by the attribute file to be hardware functions will be done by the SAS/GRAPH® software.

EXTERNAL DRIVER OUTPUT

When the external driver uses the internal driver to communicate with the graphics device, it must produce a sequential file to be processed by the internal driver. Each record in this file is a request for the internal driver to perform one of the following operations:

- Transmit the data on this record to the graphics device
- Wait for a response from the graphics device
- Transmit the data on this record to the graphics device and then receive graphic input
- Receive decoded graphic input

If a software handshake is not required, the external driver puts a series of transmit records in the output file. The internal driver will pack the data on these records into a buffer to be sent to the graphics device. When the buffer contains the number of bytes specified by the user, the internal driver will send this buffer and start building the next output buffer. This buffer has a maximum size of 264 bytes. The user should always leave at least 4 bytes unused, as these bytes may be needed by the system.
When a software handshake is required, the external driver must keep track of the number of bytes that the internal driver will have placed into the output buffer. When the external driver has sent all the data for a buffer, the external driver then outputs a wait-for-response record. The internal driver will then transmit the complete buffer to the graphics device and wait for a response from the graphics device. If the response matches the NAK string from the attribute file, the buffer will be transmitted again.

A problem that will occur on some operating systems is that between each buffer sent to the graphics device, the operating system will send a series of control characters. These characters will cause many graphics devices to behave in a very undesirable fashion such as drawing spurious lines on the graph. The best method to protect against this problem is to include at the beginning of each buffer a command to put the device into graphics mode, and at the end of each buffer a command to take the device out of graphics mode.

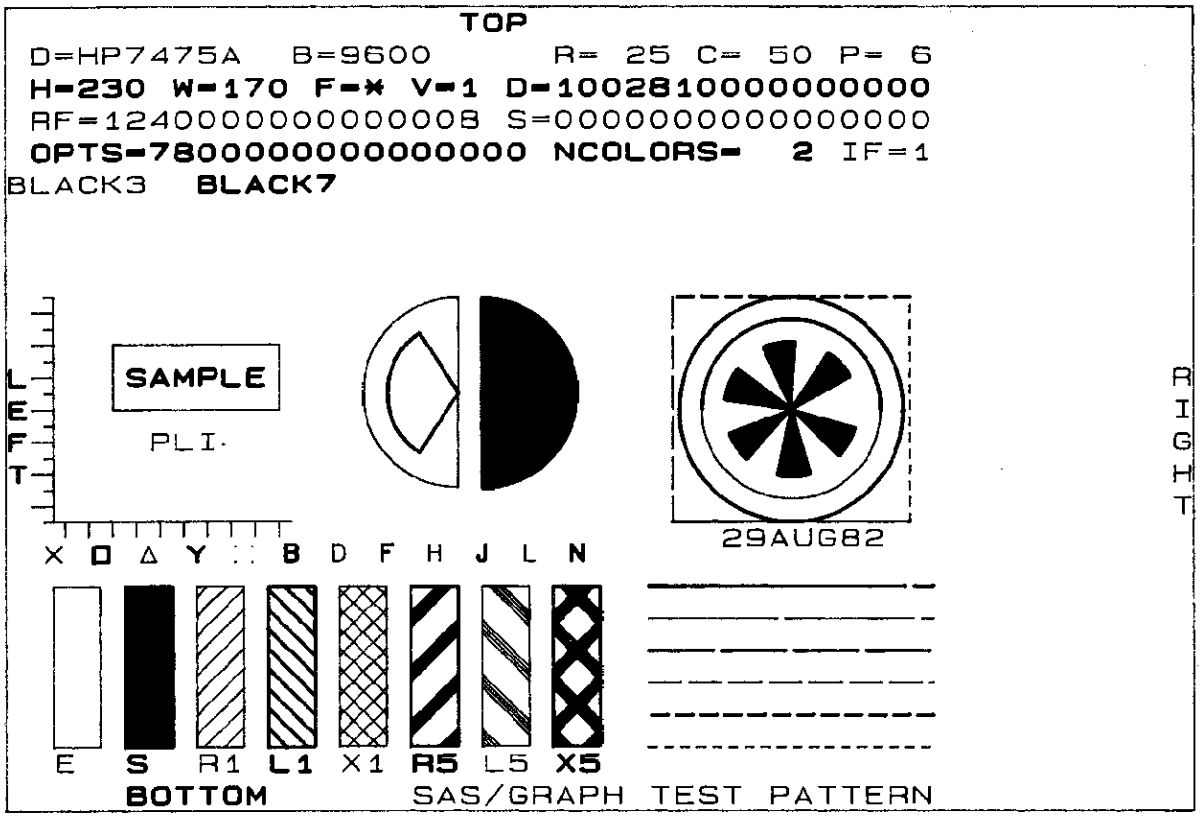
TESTING A DEVICE DRIVER

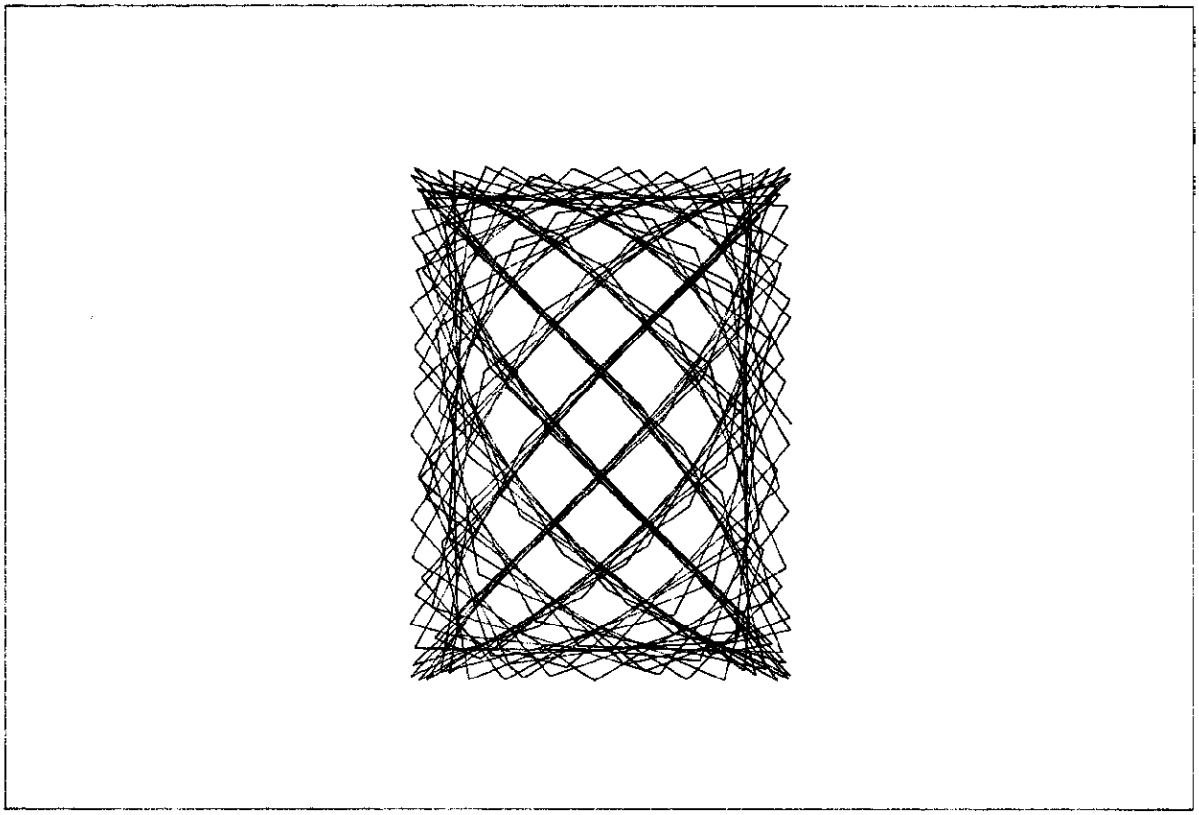
The most common way to test a new graphics device driver is with the GTESTIT procedure. This procedure draws two graphs. The first graph is a test pattern which will test most of the features of a device driver. The second graph is a test of the ability to do a continuous draw of a line through 200 vertices.

The GTESTIT procedure reports the following information:

D=	Device name
B=	Baud rate
R=	Number of rows
C=	Number of columns
P=	Number of pen mounts
H=	Character height in pixels
W=	Character width in pixels
F=	Fill increment
V=	Method of drawing vertical text 1 = Move 2 = Line-feed backspace 3 = Hardware
D=	Line styles
RF=	Rectangle fill patterns
S=	Centered symbols
OPTS=	Device options
NCOLORS=	Number of colors
IF=	Pie fill 1 = Hardware 2 = Software
GNRATIO=	Aspect ratio

The number of vertical pixels is H * R and the number of hortizonal pixels is W * C. The default aspect ratio is calculated from the values


TOP
D=HP7475A B=9600 R= 25 C= 50 P= 6
H=230 W=170 F=* V=1 D=1002810000000000
RF=1240000000000008 S=0000000000000000
OPTS=7800000000000000 NCOLORS= 2 IF=1
BLACK3 BLACK7
LEFT
SAMPLE
PLI.
29AUG82
RIGHT
X O Δ Y :: B D F H J L N
E S R1 L1 X1 R5 L5 X5
BOTTOM
SAS/GRAPH TEST PATTERN



of the attribute file parameters HEIGHT, WIDTH, XSCALE, and YSCALE as (HEIGHT * XSCALE) / (WIDTH * YSCALE).

The test pattern is 40 columns by 24 rows with a border that shows the full size of the graphic surface. If the border is the wrong size, the attribute file has incorrect values for the character size or the number of rows and columns. The color names will indicate if the colors are being selected correctly. If pies and arcs are not round, then the attribute file values for XSCALE and YSCALE are incorrect. The word SAMPLE should be centered in its box. If it is not, then there is an error with the positioning of text or the specification of the character size. The color names should start in the first character position. The four lines above the color names start in the second character position and are 38 characters long. You should especially check the position of the IF= at the end of the fourth line. If it does not line up with the right margin of the previous three lines, the character width in pixels is wrong. The words TOP, RIGHT, LEFT, and BOTTOM should be in the corresponding places on the test pattern. They indicate that the driver is correct as to direction and position. With the exception of the border, text written over any other part of the test pattern is an indication of incorrect character size or positioning.

The GTESTIT test pattern should be checked with various options of the GOPTIONS statement. In particular, set HPOS=40 and VPOS=24 to test the automatic switching to software characters on devices with fixed character sizes or the scaling of hardware characters on devices that have scalable character sets. If HSIZE= and VSIZE= do not produce graphs of the proper size, then the XSCALE and YSCALE parameters in the attribute file must be corrected. Use GOPTIONS NOFILL NOPIEFILL NOPOLYGONFILL NODASH NOCHARACTERS NOSYMBOL NOCIRCLERARC to test the software drawing of these hardware functions. The hardware and software versions should look as similar as possible. If the graphics device has both polygon fill and pie fill, the polygon fill can be tested by specifying GOPTIONS POLYGONFILL NOPIEFILL. The pie slices will then be filled using the hardware polygon fill. You should check the software polygon fill with GOPTIONS NOPIEFILL NOPOLYGON, because some fonts will use this type of fill. If the solid fills show stripes, the FILLINC parameter in the attribute file must be reduced.

In addition to using the GTESTIT procedure, the developer of the external driver should use other graphics procedures in testing the driver. A good test of the hardware polygon fill is to draw a map of the U.S. with a solid fill pattern for each state. Some devices will not be able to fill Maryland properly. If this occurs, the OPTIONS parameter in the attribute file can be set to no polygon fill which causes the polygon fills to be done in the software. The procedure GCHART with several PATTERN statements should be used to test if all rectangle fills are correct. The line styles can be tested with the GPLOT procedure and several SYMBOL statements.

CONCLUSION

This type of device driver enhances the support to users with special requirements in graphics device drivers and extends the support of user-written device drivers to minicomputers. Also, this new means of device driver support will greatly shorten the time between the appearance of a new graphics device and the ability of to use it with SAS/GRAPH® software. These device drivers will be able to use the intelligence that is in the new graphics devices, as well as give the user access to future enhancements to the SAS/GRAPH® product.