WRITING CUSTOM GRAPHIC DEVICE DRIVERS

Woody Middleton
SAS Institute, Inc.

ABSTRACT

The purpose of this tutorial is to present a method for writing a custom device driver for use with the Version 5 SAS/GRAPH software under Version 5.

SAS Institute provides a number of device drivers as well as a linkable device driver. The standard drivers provide support for a large number of graphics devices. Since SAS Institute cannot support all graphics devices, the Metagraphics Driver Facility exists to allow the user to write a custom device driver. This facility allows the user to have support for a new graphics device with full SAS/GRAPH capabilities soon after the release of the device.

FEATURES

The Metagraphics driver facility allows the user to write a device- intelligent device driver. Users can incorporate the available hardware features of the graphics device that they want to support. Some examples of these features are hardware text fonts, polygon fills, circle/arc generators, line types, rectanglefills, and so on. Another important feature of the Metagraphics facility is that it is system independant(portable). If the SAS/GRAPH software is supported on a system, then so is the Metagraphics driver facility. The Metagraphics driver facility also allows GOPTIONS support and control, and can handle all of the input/output for the graphics driver and device.

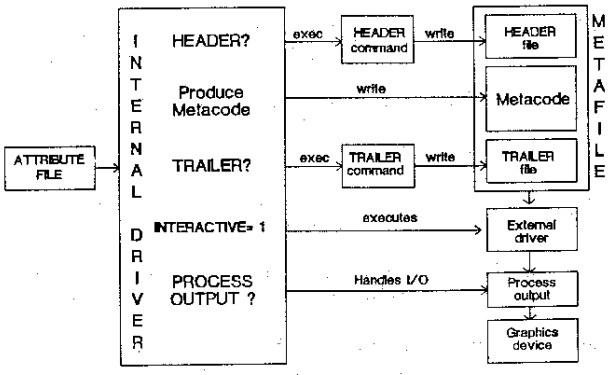
REQUIRED COMPONENTS

The Metagraphics driver facility has both required and optional components. The required components are the internal metafile driver, the external device driver, the driver attribute file and the external driver input file (metafile). The driver attribute file contains the device specific information that is loaded into the internal metafile driver. The internal metafile driver takes this information and writes a generic metafile containing the commands that are used to generate the graph on the device described in the attribute file. The external driver then takes the metafile and translates the commands to the device specific form necessary to execute the graph.

OPTIONAL COMPONENTS

In addition to the required elements, there are a number of optional components to allow the user more control over the input and output of the graphics code. Sometime the user may want to generate an External Driver output file. Another reason for this feature is to allow the user to optimize the driver code for spooling considerations. Another optional feature is the Header and Trailer writing facility. If users want a header or trailer file written to the Metafile, they can invoke the header or trailer writing code.

OVERVIEW



Since the main purpose of this tutorial is to discuss user written graphics device drivers, let this brief overview of the Metagraphics driver facility suffice. The overview above looks quite complex. The purpose is to show that the process can be much simpler. In fact the simplest way to use the Metagraphics driver facility is to write the attribute file and let the internal driver write the metafile code, which will be used to generate the graph at a later time. The only user-written element that is always required is the attribute file.

ATTRIBUTE FILE

The attribute file is the description of the graphics device characteristics, the method of file processing and the format of the output file. Following is a brief description of the elements of the attribute file.

HEADER Program that writes the header file

HEADERFILE	Name of file containing header information
TRAILER	Program that writes the trailer file
TRAILERFILE	Name of file containing trailer information
PROCESS	External driver name
PROCESSINPUT	Name of the input file for the external driver (metafile)
PROCESSOUTPUT	Name of the output file from the external driver
INTERACTIVE	Level of interaction between the internal and external drivers
NAK	Negative handshake response string
HANDSHAKE	Handshake or not
FORMAT	Data format of the input and output files
ID	Character string to identify this attribute to the external driver
FOREGROUND	Default foreground color names
BACKGROUND	Default background color
OPTIONS	Defines which options are available in the device
SYMBOLS	Defines which symbols are available in the device
DASHEDLINE	Defines which line styles are available in the device
PENMOUNTS	Number of pen mounts
RECTANGLEFILL	Defines which fill patterns are available in the hardware
MAXPOLY	Maximum number of vertices per polygon
FILLLINC	Increment for drawing a solid fill pattern
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	Horizontal pixels per inch
YSCALE	Vertical pixels per inch
ASPECT	Device aspect ratio
COLORTYPE	Type of color table
FONTMAX	Maximum font number used
ROTATION	Increment by which hardware text can be rotated
PATH	Increment by which the path of hardware text may be rotated
BLKSIZE	Maximum buffer size of the graphics device

The attribute file should be the same name as the device for which it is written.

INTERNAL DRIVER

The internal driver, supplied by SAS Institute, has the following functions:

1. The internal driver first will process the attribute file provided by the user.
2. The internal driver will process the GOPTIONS statements specified by the
3. The internal driver processes the header and trailer files, if needed.
4. The internal driver outputs the Metafile command file, either in character or binary format.

   Character format: the file is 80 bytes long and has 16 fields of 5 bytes each.

   Binary format: The file is 80 bytes long and has 20 fields of 4 bytes each.

An example of this type of command is the following: The command code is '41', which is followed by two coordinates, the x and y coordinate respectively; for example, 41 1049 1267.

EXTERNAL DRIVER

Once the metafile is generated, the INTERACTIVE= option in the attribute file determines what will happen next. If INTERACTIVE=0, then the internal driver stops processing. If INTERACTIVE=1 then internal driver calls the user written external driver to execute the metafile commands.

The external driver, then must process the metafile commands. In order to do that the external driver must be able to input the metafile

commands in the format, character or binary, they are written. This usually necessitates an input subroutine, which has the functions of inputting the command and formatting it to a form the external driver can use. If the input format is character, then the input routine must convert this to numeric if a draw or move coordinate is needed. On the other hand, the character format is fine for text to be printed.

The external driver also must be able to output the graphics commands so that the device can execute the commands. If the internal driver is to handle the output to the device then the commands must be in a set format: 80 byte records, each containing one command. The format for the character format is:

```
bytes 1-2  record length
bytes 3-4, operation code
bytes 5-79, 25 ASCII decimal
 equivalent fields 3 bytes long.
```

If the format is binary, records are 80 bytes, each containing one command The format is:

```
bytes 1-2 record length,
bytes 3-4  command code
bytes 5-79 75 bytes of ASCII data.
```

If the external driver is writing to a 'process output' file, the external driver determines the format so that the commands are best executed on the graphics device(s) used. The records must be no longer than 264 bytes. The user should leave 4 bytes unused, because these bytes may be needed by the system.

Once the input/output formats are resolved, the external driver must process the commands as they are sent from the metafile.

METAFILE COMMANDS

The Metafile contains a series of commands used to generate the commands necessary to execute a graph.

Header	Header infomation
Trailer	Trailer information
Initialization	8 bytes from GOPTION PROMPTCHARS
autocopy/autofeed	GOPTION AUTOCOPY or AUTOFEED
Terminate driver	
Shutdown device	
Erase graph	
Prompt	prompt message
Load Color table	load type of color table & list
ID	identification code
Digitize	
Decode	
Move	set position
Draw	draw present line type
Draw arc	
Fill pie slice	draw solid pie slice
Rectangle fill	draw rectangle with present fill pattern
Polygon fill	draw solid polygon
Text	hardware text string at present position
Symbol	draw hardware symbol at present position
DOT	draw dot
Machine	machine name
SAS/GRAPH version	version number
Interaction level	level of interaction
Select color	
Select line style	
Character type	select font
Set text direction	
Set character size	set cell size
Set window size	size of graph area
Select pattern	select fill pattern
Character slant	
Set line width	
Rotate	

This is a complete list of commands presently supported by the Metagraphics driver facility. The user controls the commands which are called by using the attribute file. For example, unless the attribute specifies a hardware polygon fill, SAS/GRAPH software will generate the draw and move command to draw a solid polygon.

Each of the commands has a command code and parameters. The number of parameters is dependant on what is needed to generate that command.

EXAMPLE 1:

A device expects a draw command and the two endpoint coordinates in hexdecimal characters. A process to translate the metafile command to the device specific driver code may look like the following:

```
/* ---  PL/1 Example -- DRAW --*/
/* --- operation code structure -- */
   NEXTCODE: CALL INPUT(CODE);
      /* -- ERROR CHECK FOR RANGE OF CODE --*/
      /* -- PRINT ERROR MESSAGE --*/
      IF ERROR THEN DO;
      ELSE GOTO COMMAND(CODE);
      . . . . . .
/* ---       DRAW        ------ */
   COMMAND(41):
      /*- INPUT FORMATS CODE INTO INTEGER VALUE */
      CALL INPUT(X);
      CALL INPUT(X);
      /*  PUTXY SENDS DRAW COMMAND AND HEX X & Y */
      CALL PUTXY(X,Y);
      /*- RETURN TO BEGINNING OF COMMAND TABLE   */
      GOTO NEXTCODE:
```

EXAMPLE 2:

Sometimes more than one metafile command is necessary to execute a device function. Or more often a device will require a textsize command each time a text string is drawn. In this case until the graph is being drawn, now text size should be sent. But the text size is sent to the external driver even before the device is initialized. The situation could be resolved by the following example.

```
/* ----        SET TEXT SIZE          ----- */
    COMMAND(64):
       CALL INPUT(HEIGHT);
       CALL INPUT(WIDTH);
       /* SEND TEXTSIZE WHEN GRAPH IS BEING DRAWN*/
       IF GRAPH THEN DO;
       . . . . . .
/* ----        SET GRAPHIC WINDOW ------ */
    COMMAND(65):
       CALL INPUT(XSIZE);
       CALL INPUT(YSIZE);
       /* SEND DEFAULT TEXT SIZE */
       IF GRAPH='0'B THEN DO;
```

In this routine the GRAPH flag is turned on when the screen is cleared.

SUBLIB ROUTINES

Many vendors will write a set of routines that can be called to generate graphics on their devices. The Metagraphics driver facility can be used with these routines. The routines will have to be linkedited by the user, but once linked the external driver can call those routines directly and the routines will handle the input/output with the graphics device.

```
/* ----        DRAW                   ------ */
    COMMAND(41):
       CALL INPUT(X);
       CALL INPUT(Y);
   /* CONVERT INTEGER TO REAL NUMBER    */
       CALL CONVERTI(X,XR);
   /* CONVERT INTERGER TO REAL NUMBER   */
       CALL CONVERTI(Y,YR);
   /* SUBLIB ROUTINE DRAW               */
       CALL DRAWA (XR,YR);
   /* DRAWING FLAG                      */
       IF DRAWIN=FALSE THEN DRAWING=TRUE;
       GOTO NEXTCODE;
```

CONCLUSION

The Metagraphics driver facility provides a means of writing a customized device driver for whatever device is in an installation. This facility is especially useful for a minicomputer environment. The facility also greatly reduces the time lapse between the introduction of a new graphics device and its support by SAS/GRAPH software. Installations with the SAS/GRAPH software can write their own customized device drivers soon after the installation of a new device, which may not be currently supported by SAS/GRAPH. Also, because the internal driver is maintained by SAS Institute, the user will have current enhancements to SAS/GRAPH software without having to change their device driver.