CHAPTER

18

The GKEYMAP Procedure

Overview

The GKEYMAP procedure creates key maps and device maps that compensate for differences between the way that characters are encoded internally bySAS/GRAPH software and the way that they are encoded by different operating environments and output devices.

In addition, the GKEYMAP procedure can create SAS data sets from existing key maps and device maps, either Institute-supplied or user-generated. This capability is useful when you want to make minor alterations in a large key map or device map and you do not want to or cannot re-create the original data set with a DATA step.

The Institute supplies key maps for many keyboard configurations and operating-environment character representations. Your SAS Software Consultant should have selected the appropriate key map for your site. If the Institute-supplied device maps and key maps do not meet your needs, you can use this procedure to modify an existing map or create a new one.

Concepts

About Key Maps and Device Maps

The characters A through Z (upper- and lowercase), 0 through 9, and many symbols and national characters are represented by a set of hexadecimal codes. However, a

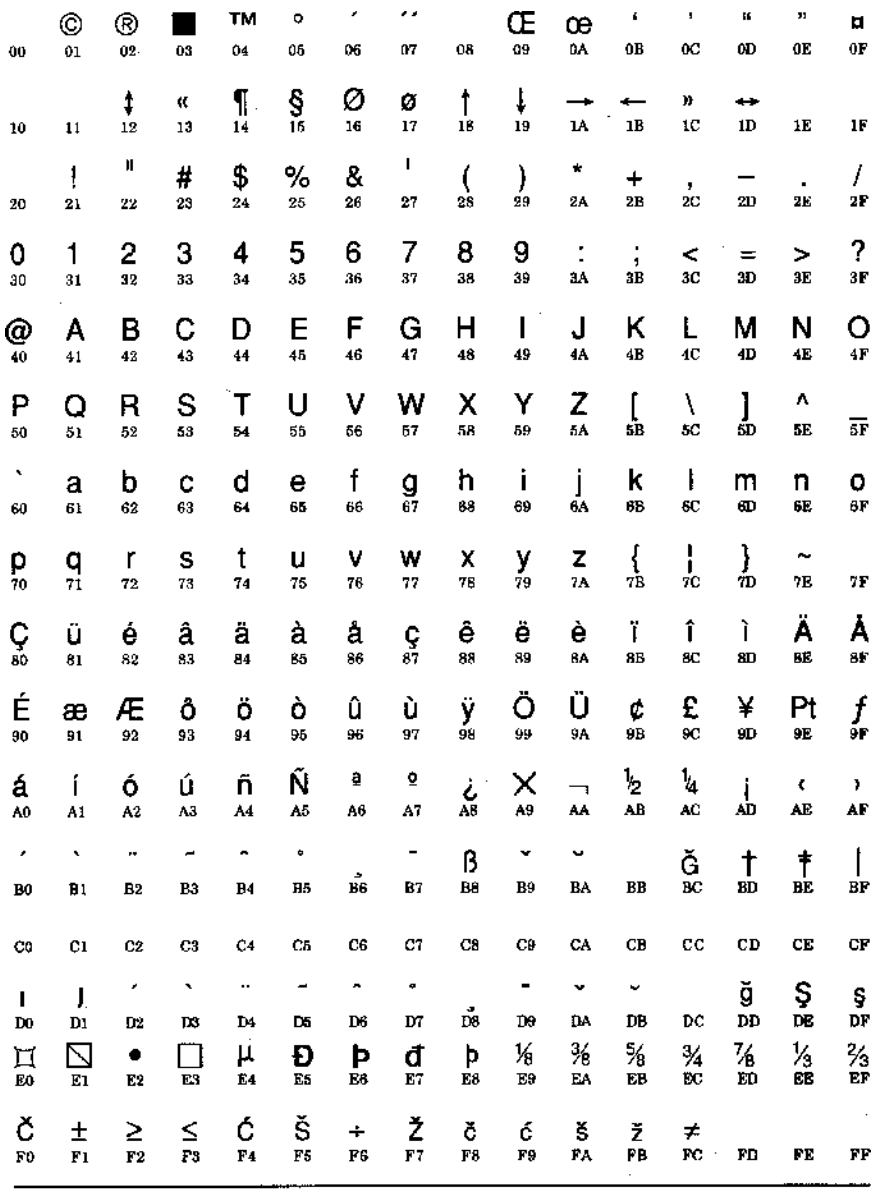
character may be represented by one code for the keyboard, another code for the operating environment, and yet another for the output device. To resolve these differences, SAS/GRAPH software stores all characters using its own internal encoding scheme, which is a set of hexadecimal values that are associated with all supported characters. Figure 18.1 on page 721 shows these internal character encoding (ICE) codes.

To accommodate differences in the encoding of characters, you must be able to translate the hexadecimal codes generated by your keyboard or operating environment into the corresponding SAS/GRAPH internal encoding. A key map gives you this ability.

You also must be able to convert the internal encoding that is used by SAS/GRAPHsoftware to the codes required to produce the corresponding hardware characters on your output device. A device map gives you this ability.

Key maps and device maps are SAS catalog entries. Institute-supplied key maps and device maps are stored in the catalog SASHELP.FONTS. User-generated key maps and device maps are stored in the catalog GFONT0.FONTS. Key maps are stored with the extension KEYMAP (for example, GERMAN.KEYMAP), and device maps are stored with the extension DEVMAP (for example, DEFAULT.DEVMAP).

Figure 18.1 SAS/GRAPH Internal Character Encoding



Note: Positions 00-1F are reserved.
Note: SAS Institute reserves the right to change, at any time, the character displayed and the hexadecimal code returned for all undefined codes.

What Key Maps Do

A key map changes the code generated by a keyboard key to the value corresponding to the SAS/GRAPH internal character encoding. Otherwise, a different character (or no character) may be drawn when the character is requested in a SAS/GRAPH software font.

Key maps are required when the code that is sent to the operating environment does not match the SAS/GRAPH internal encoding for the character corresponding to the key that is pressed. They are useful for generating a character in a software font that is not available on your keyboard or when the same key on different keyboards sends a different character to the operating environment. They are also useful for creating new characters by combining existing characters with accent characters (called *diacritics*).

Note: In Figure 18.1 on page 721, the diacritic characters specified by the codes D2 through DB are backspaced before being drawn and can be used to create new characters (characters resulting from codes B0 through B7, B9, and BA are not backspaced before being drawn). See Example 1 on page 726 for an example of using a diacritic character as an accent. Two commonly used characters have already been created for you: the character located in position F0 of the ICE table could be created by combining DA with an uppercase C, and the character located in position BC could be created by combining DB with an uppercase G. △

What Device Maps Do

A device map maps the code stored in the SAS/GRAPH internal encoding to the code required to reproduce the character on the output device when a particular hardware character is requested in a SAS/GRAPH program.

You usually use device maps in these two situations:

- reversing the translation performed by key maps (if needed). To display the proper hardware character, you must use a device map to convert the SAS/GRAPH internal encoding of the character back to the encoding that the device expects.
- accounting for differences between the code that represents a character on the operating environment and the code or codes required to generate the same character as a hardware character on an output device. The problem can be further complicated if you have multiple output devices, each with its own way of generating a particular character using hardware text.

Using Key Maps and Device Maps

You use key maps and device maps by specifying them with the KEYMAP= or DEVMAP= options in a GOPTIONS statement. You also can specify a device map by filling in the DEVMAP field in the Detail window of the device entry for the device driver that you are using.

For example, if you use the GKEYMAP procedure to generate a key map called MYKEYMAP, you can specify it with a statement like this:

```
goptions keymap=mykeymap;
```

Once you specify MYKEYMAP as your current key map, you can press a key and the code it generates is translated by MYKEYMAP into the ICE code that is specified by the key map.

When you specify a device map with the DEVMAP= graphics option and you use a hardware character set, mapped characters are converted from their SAS/GRAPH internal encoding to the codes required to display the corresponding characters on your device. See Chapter 9, “Graphics Options and Device Parameters Dictionary,” on page 301 for more information on the KEYMAP= and DEVMAP= graphics options.

Asymmetrical Maps

It is possible, and sometimes necessary, to define a key map or device map that is not symmetrical (that is, two or more input character codes map to the same output character code). For example, if you define a key map to map the keyed character A to the internal encoding for B, the keyed characters A and B both map to the internal encoding for B, but no code maps to A. This situation may make it impossible for you to display certain characters defined in software fonts.

Seeing What Characters in a Font are Available

To see what characters in a font can be displayed if a particular key map is used, do the following:

1. Use the KEYMAP= option in a GOPTIONS statement to specify the key map that you are interested in.
2. Then, use the GFONT procedure with the ROMHEX option to display the font that you want to use.

The hexadecimal values and corresponding font characters that are displayed are the ones available under the specified map. If the map is not symmetrical, a warning is issued. See Chapter 16, "The GFONT Procedure," on page 675 for more information on using hexadecimal values to display special characters.

About the GKEYMAP Data Set

To generate a key map or device map, you must create a data set that contains the mapping information and use that data set as input for the GKEYMAP procedure. The mapping information is specified as values for the variables in the data set, which should contain one observation for each character or key to be mapped. Any characters not specified in the data set are passed through the map unchanged.

GKEYMAP Data Set Variables

To provide information on the character mapping that is to be performed for a key map or a device map, you must use a variable named FROM to specify the character that you are mapping from, and a variable named TO to specify the character to map to. For key maps, these are the only variables in the data set. For device maps, you may also need variables named CHARTYPE and TOLEN.

Here are definitions for these variables:

CHARTYPE
specifies which hardware character set to use when a device requires that you select an alternate character set in order to display certain characters. CHARTYPE is a numeric variable.

All of the characters in the TO string for a particular FROM value must use the same character set. The CHARTYPE variable is required if you use the MULTFONT option in the PROC GKEYMAP statement; otherwise, it is ignored. (The CHARTYPE variable is always ignored when you are creating a key map.) The CHARTYPE value must match a value listed in the Chartype field in the Chartype window of the device entry for the device to which the map is applied. However, you can set the CHARTYPE variable to a missing value to specify that the character can be drawn in any hardware character set.

FROM
specifies the character you are mapping from. FROM is a character variable. For each observation, the FROM variable should contain a single character value. Any characters after the first are ignored. The data set must be sorted by the FROM variable.

Featured in: Example 1 on page 726

TO
specifies the string that the character in the FROM variable is mapped to. TO is a character variable.

For device maps, if the TO variable contains more than one character, you must also specify TYPE=MAP1N in the PROC GKEYMAP statement to indicate that a single FROM character is being mapped to multiple TO characters. In addition, you must include the TOLEN variable in the data set to specify the length of each TO string. If you specify TYPE=MAP11 in the PROC GKEYMAP statement or if you do not use the TYPE= option, only the first byte of the TO string is recognized.

Featured in: Example 1 on page 726

TOLEN
specifies the length of the string in the TO variable. TOLEN is a numeric variable. The TOLEN variable is used only with device maps and is required if you specify TYPE=MAP1N in the PROC GKEYMAP statement; otherwise, it is ignored.

Procedure Syntax

Requirements: The NAME= argument is always required. To create a key map or device map, the DATA= argument is required. To output a data set, the OUT= argument is required.

PROC GKEYMAP NAME=*map-name*
data-set-argument
<*option(s)*>;

PROC GKEYMAP Statement

The PROC GKEYMAP Statement names the key map or device map to be created or output as a data set. If the procedure creates a key map or a device map, it identifies the data set that is used as input. If it outputs a map, it identifies the data set to which the map is written.

Syntax

PROC GKEYMAP NAME=*map-name*
data-set-argument
<*option(s)*>;

data-set-argument must be one or more of the following:

DATA=*keymap-data-set*

OUT=*output-data-set*

option(s) can be one or more of the following:

DEVICE=*device-name*

DEVMAP | KEYMAP

TYPE=MAP11 | MAP1N

MULTFONT

Required Arguments

NAME=*map-name*
identifies the map that is to be created or converted to a SAS data set. Key maps are stored as *map-name*.KEYMAP, and device maps are stored as *map-name*.DEVMAP. The value of the KEYMAP or DEVMAP option determines the type of map and the

extension added to *map-name*. It is possible to use the same *map-name* value for both a key map and a device map.

If you create a key map or device map, the map is stored as an entry in the catalog GFONT*n*.FONTS where *n* is a number from 1 to 9, and you must use a LIBNAME statement to specify a libref for GFONT*n*. See "About the Libref GFONT0" on page 677 for details.

If you specify an existing key map or device map, SAS/GRAPH software searches for the map using the same search path that it uses to search for fonts. See "Font Locations" on page 127 for details.

Featured in: Example 1 on page 726

DATA=*keymap-data-set*

identifies the input data set for the GKEYMAP procedure. Used only when you are creating a key map or device map.

See also: "SAS Data Sets" on page 25 and "About the GKEYMAP Data Set" on page 723

Featured in: Example 1 on page 726

OUT=*output-data-set*

identifies the output data set to which the data from a key map or device map are to be written. Used only when you output an existing key map or device map as a SAS data set.

Featured in: Example 1 on page 726

Options

You can specify as many options as you want and list them in any order.

DEVICE=*device-name*

specifies the device driver that a device map is associated with, where *device-name* is the name of an entry in a device catalog. DEVICE= is not required when creating a device map, but it can be used if you want to limit the use of the device map to one particular driver. If you do not use DEVICE=, the device map can be used with any device. DEVICE= is valid only if you are creating a device map.

DEVMAP | KEYMAP

specifies whether you are working with a device map or a key map. The default is KEYMAP unless you use an option that can be used only with DEVMAP. This option also specifies the type of map you are outputting as a data set.

Featured in: Example 1 on page 726

TYPE=MAP11 | MAP1N

specifies whether you are mapping characters in a device map one-to-one or one-to-many. If you specify TYPE=MAP11 (the default), each character in a graphics text string is mapped to only one character on the output device. If you specify TYPE=MAP1N, a single character in a graphics text string can be mapped to multiple characters on the output device. For example, if two characters have to be sent to the graphics output device to display a single hardware character, specify TYPE=MAP1N. Specify TYPE=MAP1N only when you create a device map.

MULTFONT

specifies that an alternate hardware character set is required to display one or more characters in the device map. Specify the MULTFONT option only when you create a device map.

Creating a Data Set from an Existing Key Map or Device Map

To generate a data set from an existing key map or device map, follow these steps:

1. Specify the name of the key map or device map with the NAME= argument. If the map is user generated, you must first submit a LIBNAME statement to associate the libref GFONT0 with the location where the map is stored, and NAME= must specify the name that was specified for the key map or device map when it was created. If the map is an Institute-supplied map, it is located in the catalog SASHELP.FONTS, and you do not need to submit a LIBNAME statement to access it.
2. In the OUT= argument, specify the name of the data set to which the data are to be written. By default, the data set is written to the temporary library WORK.
3. Use the DEVMAP option if a device map is selected.
4. Optionally, use the PRINT procedure to display the newly created data set (most values will be unprintable, so you should use a $HEX2. format for the FROM and TO variables).

Creating and Using Key Maps and Device Maps

To create and use a key map or device map, follow these steps:

1. Submit a LIBNAME statement that associates the libref GFONT0 with the location where your map is to be stored.
2. Create a data set that contains the mapping information you need. You can use a DATA step to create all of the mapping information for the key map or device map, or you can create a data set from an existing key map or device map, then update that data set with the mappings that you need. This process is illustrated in Example 1 on page 726.
3. Use the GKEYMAP procedure to create the key map or device map, using as input the data set that contains the mapping information. The GKEYMAP procedure stores the map in the catalog GFONT0.FONTS.
4. Use the KEYMAP= or DEVMAP= option in a GOPTIONS statement to assign the key map or device map in your SAS session. The specified map is used automatically in your SAS/GRAPH programs. (The device map is used only when you use a hardware character set.)

Examples

Example 1: Modifying a Key Map

Procedure features:

GKEYMAP options:

DATA=
KEYMAP
NAME=
OUT=

Other features:

DATA step
GOPTIONS procedure
GOPTIONS statement
LIBNAME statement

SORT procedure

Sample library member: GR18N01

This example shows how to change multiple characters in an existing key map. It assumes that the national characters ß and ã are not on your keyboard, so you want to create a key map that provides them.

To provide the ß character, this example's key map converts the @ character into the SAS/GRAPH internal encoding for ('B8'x). Whenever the @ character is typed in text that is displayed with a software font, the character ß is drawn instead. In this case, the replacement character uses the text position that would have been used by the typed character.

Note: Once you have modified your key map so that @ is mapped to ß, you can no longer generate @ in a software font from your keyboard when the key map is in effect. △

To provide the ã character, which is not on the keyboard or in the ICE table, this example's key map converts the asterisk (*) into the SAS/GRAPH internal encoding for the accent character 'D5'x (a tilde). In this case, when the character * is typed, the resulting tilde does not take up a text position but is backspaced and used as an accent over the character preceding it in the text. To create the ã character, therefore, the text must contain the two characters a*.

Note: The example updates the current key map rather than creating a new key map so that all of the other character mapping in the key map remains in effect. △

Assign the libref and set the graphics environment. LIBNAME associates the libref GFONT0 with the location of the SAS data library where your device maps and key maps are stored.

```
libname gfont0 'SAS-data-library';
goptions reset=global gunit=pct border cback=white
         colors=(black blue green red)
         ftext=swiss ftitle=swissb htext=6;
```

Determine the name of the current key map. The SAS log in Output 18.1 on page 729 shows that the keymap name is DEFAULT.

```
proc goptions
   option=keymap;
run;
```

Copy the DEFAULT key map to a temporary SAS data set. NAME= specifies the DEFAULT key map as input to the procedure. OUT= specifies the data set TEMP, which is created from the specified key map.

```
proc gkeymap name=default
   out=temp;
run;
```

Create data set NEW. NEW will be used to create the key map for the character conversions. Values for the FROM variable are the keyboard characters to be converted. Values for the TO variable are hexadecimal codes from the SAS ICE table. OUTPUT is required to write a separate observation for each character to be mapped.

```
data new;
   from='@';
   to='b8'x;
   output;
   from='*';
   to='d5'x;
   output;
run;
```

Sort data set NEW and update data set TEMP with the mapping information. The data set NEW must be sorted by the FROM variable before its observations can be used to update data set TEMP.

```
proc sort data=new;
   by from;
data temp;
   update temp new;
   by from;
run;
```

Create a new key map from the modified data set. NAME= assigns a name to the new key map. DATA= specifies the data set TEMP as input to the procedure. KEYMAP specifies that the map being generated is a key map (the default).

```
proc gkeymap name=mykeymap
             data=temp
             keymap;
run;
```

Specify the new key map in a GOPTIONS statement. KEYMAP= specifies the name of the new key map so that when the characters @ and a* are specified in TITLE statements, the characters ß and ã are displayed in the output.

```
goptions keymap=mykeymap;
```

Print two titles with the special characters. The character @ is typed where the character ß should print, and the character * is typed after the character it will accent.

```
title1 'Kaiserstra@e';
title2 'Sa*o Paulo';
footnote j=r 'GR18N01 ';
proc gslide;
run;
quit;.
'
```

Output 18.1 Log from GOPTIONS Procedure

```
                SAS/GRAPH software options and parameters
                  (executing in DMS Process environment)
KEYMAP=DEFAULT               Input character map for hardware and software
                             text
```

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS/GRAPH® Software: Reference, Version 8*, Cary, NC: SAS Institute Inc., 1999.

SAS/GRAPH® Software: Reference, Version 8

ISBN 1-58025-525-6

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, October 1999