0010

COLLABORATORS

	TITLE : 0010	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		January 16, 2023	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

0010

1.1 Personal Fonts Maker - 9. The Printer Driver Modifier

9. The Printer Driver Modifier

The Personal Fonts Maker package contains a program called Printer Driver Modifier. The program is stored in the "PFM_Tools" drawer of the disk containing the Personal Fonts Maker program. This chapter contains some general information on the environment in which the Printer Driver Modifier works. Chapters 10 and 11 describe the program's functions in detail, while appendix H explains the messages which may be displayed by the program, and appendix J lists all the shortcuts to the commands.

9.1 How Printer Drivers Work

Printer drivers were developed to bring some unity to the babel of different printer standards, control codes, character sets and command sequences. If all printers could interpret the same data sent by the computer, there would be no need for printer drivers. Unfortunately, printers developed at different times, by different companies, and/or for different purposes cannot always be controlled in the same way.

To work properly with different printers, a program would need a special module for each printer. This means that each program would have to contain the information about the character sets, the commands and other codes of every printer that might be used. If a new printer "standard" is developed after the program is finished, a new version of the program has to be written. Printer drivers remove this burden from software working with printers.

On a system which uses printer drivers, like the Amiga, programs do not need to know about which printer is connected. The program's output goes to the printer driver, which in turn translates the data into the format required by the printer. The Amiga printer device governs all accesses to the drivers and controls the data flow. The user informs the system about the printer which is to be used through the Amiga Preferences. When the printer device receives data from a program, it re-routes it to the appropriate driver.

Programs wishing to communicate with the printer device only need to be

able to handle one, single, standard set of characters and control codes. The printer device acts as a filter which, depending on the printer selected by the user, either ignores the command sequences or perhaps translates them into entirely different sequences that this printer can actually understand and obey.

The character set recognized by the Amiga printer device is the Amiga set, as in appendix D. Most control sequences are based either on the ANSI x3.64 rules defined by the International Standards Organization (ISO), or on control sequences defined by Digital Equipment Corporation. The Amiga documentation describes every control code in detail. This is all a program using printer drivers to print text needs to know. All printer-specific codes are handled by the printer drivers.

When a program needs to print a text, it sends a flow of characters, with interspersed control codes wherever necessary. For example, if a word in the middle of a sentence needs to be underlined, the following sequence might be used.

This is a sentence with an ESC \[\4 \m underlined ESC \[\2 \4 \m word.

The standard FFDL syntax for constants was used in this example. Section 2.7 ("Programming the Output Format: the Cloanto FFDL") describes this format, which is used by the Printer Driver Modifier and in the examples regarding the program.

The "ESC \[\4 \m" sequence in the above example is interpreted by the printer driver as "Underline On", while "ESC \[\2 \4 \m" means "Underline Off". These commands could mean nothing to the printer being used, therefore they have to be translated by the driver. For Epson, IBM Proprinter, NEC Pinwriter and compatible printers, for example, the two sequences would be translated as "ESC \- (1)" and "ESC \- (0)" respectively. If the printer selected through the Amiga Preferences cannot underline text (or if the driver cannot), the driver does not translate this command (and no command is sent to the printer).

Single characters can also be translated by the driver. Most printers do not use the Amiga character set, but - for example - the IBM PC set (appendix C) or a variant thereof. The letters and punctuation signs normally used to write English text are shared by many different character sets, since they are defined by the original 7-bit ASCII character set standard (appendix E), incorporated by most 8-bit character sets. But for many non-English national characters, and signs like '½' (one half), things are not as straightforward. These codes need to be converted by the driver to the equivalent codes in the printer's character set. Section 2.8 has more on character sets.

There are some exceptions, of course. Some word processors do not use the Amiga drivers, but their own drivers and conversion tables. Other programs print text in graphics mode, which is normally slower and of lower quality compared to the letter quality modes of modern printers. The Personal Fonts Maker can download fonts to be used in text mode. Programs adopting the graphics mode can usually load these same fonts saved by the Personal Fonts Maker in the Amiga font format.

9.2 Problems with Standard Drivers

Many users experimenting with printer control codes may have noticed that the Basic programming examples in the printer's handbooks often do not work as expected on the Amiga. This is because the "LPRINT" Basic command does not send the control sequences directly to the printer, but to the Amiga printer device, which expects a code in the Amiga format rather than a code in the printer's format. It is out of the scope of this manual to explain how to use the Amiga's Basic programming language, but since such experiments may be useful for a better understanding of the printer and the programs described in this guide, some possible solutions are explained here.

There are at least three methods to send a control sequence from an Amiga Basic program. The easiest way is to consult the Amiga documentation and adopt the Amiga's control codes instead of the codes used by the printer. The second method is to bypass the driver, sending the data directly to the printer. To do this, the data has to be written to a file, named "PAR:" if the printer is connected to the parallel port, or "SER:" if the serial port is used instead. The third possibility is to use a special Amiga control sequence (to be sent to the driver), which means "Send the following n characters directly to the printer, without conversions", where 'n' indicates the number of characters of which the control sequence in the printer's format consists. The sequence is "ESC \[\n \" \r", followed by the data in the printer's format, where 'n' is written as a decimal number (one ASCII digit in this example).

For example:

```
LPRINT CHR$(27);"[3";CHR$(34);"r";CHR$(27);"-";CHR$(1)
```

sends the control sequence which activates the underlined print mode on an Epson, IBM Proprinter, NEC Pinwriter or compatible printer. The equivalent FFDL sequence is:

```
ESC \[ \3 \" \r ESC \- (1)
```

where it should be noted that "\3" means "The (Amiga) code associated with the character '3'", which is 51, while the value "(1)" already indicates a code (one).

Unfortunately, printer drivers cannot be changed as easily as a line in a Basic program. Sometimes a printer driver for a particular printer is not available, or the existing drivers do not work correctly. The Printer Driver Modifier was designed to help in similar situations. This section continues with the description of some problems which can be solved with the Printer Driver Modifier.

A common problem is that no printer driver exactly matches all commands and/or character codes of a particular printer connected to the computer. The documentation enclosed with the printer usually contains a list of printer drivers designed for other printers, in order of preference. Even when the closest matching driver is used, it is likely that some control sequences will not be translated as required by the printer.

Many of the drivers which come with the Amiga operating system were designed to work with a wide range of printers and data transmission

modes. This has in some cases limited the possibility of exploiting features unique to a more restricted number of printers. Unfortunately, there is not one perfect printer driver for each different printer model. The Printer Driver Modifier can be used to custom-tailor an existing driver to the control sequences and character codes of the printer being used.

A major limitation of some printer drivers is their inability to exploit the full set of 8-bit character codes used by most printers. This means that the characters which are coded by the ASCII 7-bit character set (appendix E) are sent correctly to the printer, while unexpeced transformations on characters whose code is greater than 127 (the maximum code representable with 7 bits) can be performed. This problem is well known to those Amiga users who need to write text in languages with national characters which are not part of the 7-bit ASCII set. Also, users who have printed some less-used characters like '¼' (one fourth) may have noticed that the printed output did not correspond to the characters shown in the printer's documentation or in the printer self-test.

Many printer drivers do a lot of work in order not to have to send data with the 8th bit set (i.e. a value greater than 127). This ensures compatibility with printers configured for 7-bit data transmission, but places unnecessary restrictions on the users of other printers. For example, to print an 'à' letter (lower case 'a' with grave accent), many drivers (at the time of writing Epson X, Epson Q, NEC Pinwriter and others) instruct the printer to switch to the French 7-bit character set (appendix E), print the character whose decimal code is 64 (an 'à', in the French set) and then switch back to the USA 7-bit set. If a downloaded font is being used, this can cause the accented letter to be printed in the default printer font, with the text surrounding it printed with the downloaded font. The simplest solution would be to set the printer so that only the IBM PC set needs to be used, and send a decimal 133 code to print the 'à' (as in appendix C). This can be done with the Printer Driver Modifier.

Exploring the drivers a bit more, many other interesting characteristics can be found. On the same drivers mentioned above, for example, the '\ensuremath{\pm}' (plus minus) sign is composed by printing a plus ↩ sign,
followed by a backspace code, and finally overwritten by an underscore character ('_'). There is even more overhead involved in printing the '½' (one half). First, the superscript mode is activated, then a '1' (one) is printed, the normal characters are reselected, a backspace is output, a '-' (minus) sign is printed, another backspace output, the subscript mode is activated, the '2' (two) is printed, and finally the normal print mode is restored. If proportional characters are used, the result is almost unreadable, as the three characters used have different sizes, and the backspaces cause them to be printed at different horizontal positions. The Personal Fonts Maker could replace this long sequence with a single code of the PC set (decimal 171, for example, to print '½', as in appendix C).

Some printer drivers, like the HP LaserJet driver (which works with the Roman8 character set), have conversion tables for most 8-bit characters. Unfortunately, the average Amiga user does not own such a printer. This is one of the reasons for which the Printer Driver Modifier was created. Perhaps after this handbook is printed many Amiga users will buy a new

printer, or new drivers will be released. The Printer Driver Modifier will remain a useful tool to analyse printer drivers, understand how they work, and - whenever necessary - make some quick changes.

Another problem may occur with some printers which do not accept the downloading of characters having codes greater than 127. The Personal Fonts Maker can be used to download the characters whose code is smaller than 128. In this case, the 7-bit philosophy of some drivers could be useful, if only the technique of switching to different 7-bit sets did not restore the printer's default font. The only solution is to assign a 7-bit code to those characters to be printed whose code would normally be greater than 127. This is exactly what the national 7-bit character sets (appendix E) do. Both the Personal Fonts Maker and the Printer Driver Modifier come with character set data files for 7-bit character sets.

9.3 The PDM: General Description

The Printer Driver Modifier gives the user full access to any driver's conversion tables for characters having a code greater than 127 and control sequences. A control sequence or a character received by a printer driver are translated into a (usually) different set of one or more codes. These codes are displayed in the standard FFDL format for constants (section 2.7.1) and can be modified by the user.

The Printer Driver Modifier can be used to modify existing control sequences of the printer driver, or add new ones. Switches to national 7-bit character sets to print a character can be replaced with a single 8-bit character code. The same can be done with those characters in the set available on the printer which are represented by the driver as a series of partial character images and backspaces. If a character is not available on the printer, it can be downloaded with the Personal Fonts Maker, and the printer driver can be updated with the PDM so that the new character is translated correctly. If a character cannot be downloaded because the printer being used does not accept the downloading of character having codes greater than 127, a new code (smaller than 128) can be assigned to the character, both in the downloaded font (with the Personal Fonts Maker) and in the printer driver (with the Printer Driver Modifier).

Some functions of a printer driver are not representable as simple data-to-data translations. The graphics modes routines, for example, are procedures built into the drivers. These cannot be modified with the PDM. Characters whose code is smaller than 128 are not re-mapped by the drivers, as the standard 7-bit codes are common among most character sets. The Printer Driver Modifier can be used to assign a 7-bit code to a character whose code is greater than 127, but not vice versa. As described in section 11.5 ("Encoding Mode"), it is not possible to increase the total size reserved by a driver for the conversion tables' memory. This is usually not a problem, as the control sequences are generally either modified or replaced with a single 8-bit code, therefore occupying less memory. These limitations do not reduce the power and flexibility of the Printer Driver Modifier when the program is used in the environment it was designed for: text processing with the aid of the Personal Fonts Maker. This is probably the ideal field for the Printer Driver Modifier, which is likely to become a precious tool on several other occasions.

9.4 The PDM: The Main Window

The Printer Driver Modifier can be loaded by double clicking on its Workbench icon, as described in section 1.11 for the Personal Fonts Maker. The program immediately presents itself with the main window, which is opened on the Workbench screen.

The window can be moved around the screen dragging the title bar with the mouse. The gadget on the left of the title bar can be used to close the window and terminate the program, as described in section 10.10 ("Quit").

The following subsections describe the functions of the gadgets and fields which appear on the program's main window. Section 1.9.8 ("Menus"), chapters 10 ("PDM: The Project Menu") and 11 ("PDM: The Preferences Menu"), appendix H ("PDM Program Messages") and appendix J ("PDM Command Key Shortcuts") contain additional documentation on the Printer Driver Modifier.

9.4.1 The "Number" Gadgets

This string gadget always displays the code associated with the current command or character. If the command mode is active, the code refers to an Amiga standard printer control code. Commands are numbered starting from 0. Each command has an associated name (section 9.4.3, "The 'Command' Field"), an extended function description (section 9.4.6, "The 'Function' Field"), and a control sequence in the printer format (section 9.4.2, "The Editing Gadget"). If the character mode is selected, the value displayed in the gadget refers to the code of the current character in the Amiga's character set (appendix D). The codes of the characters which can be accessed by the Printer Driver Modifier range from 160 (the first) to 255. The command codes are numbered from 0 (zero) to 75.

A new value can be defined by the user either by selecting the string gadget and modifying the number with the keyboard, or using the two arrow gadgets on the right of the string gadget, which increase or decrease the displayed value by one unit at a time, as described in section 3.3 ("The "Character #" Gadgets"). The <+> and <-> keys can also be used instead of the two arrow gadgets (the <Amiga> key does not need to be held down).

Section 10.7 ("Section") explains how to switch between the command mode and the character mode.

9.4.2 The Editing Gadget

This string gadget, on the right of the "Number" gadgets, contains the FFDL sequence which describes the data to be sent by the driver to the printer when the Amiga control sequence associated with the command

described in the "Function" field (section
9.4.6
) or the character which
appears in the "Character" field (section
9.4.4
) is received by the
driver.

The FFDL language provides different representations for constants. Section 2.7.1 ("FFDL Constants") describes the different formats. The Printer Driver Modifier always accepts constants written by the user in any format defined by the FFDL. Section 11.4 ("Decoding Mode") explains how the data extracted from a printer driver can be represented by the Printer Driver Modifier. Section 11.1 ("Code Table") explains how to display a help window containing all ASCII shortnames for the codes from 0 to 32.

The Printer Driver Modifier accepts two special constant codes: NOAV and DLAY. These codes may appear in sequences associated with printer commands (not characters). NOAV stands for "NOt AVailable", and means that a particular command cannot be represented with a sequence of constant codes. This can also mean that the function is handled by a special procedure in the driver, which is not accessible through the Printer Driver Modifier. If NOAV is written by the user in the string gadget, then no other codes may be associated with that function. If a given command has no equivalent command sequence, a NOAV instruction should be placed in the string gadget, rather than leaving it empty.

The other special code is DLAY, which means "DeLAY". DLAY may appear alone or in the middle of a sequence of codes. When the driver is sending the data associated with a function to the printer, it pauses before the remaining part of the control sequence or the following codes are output. The duration of the delay varies from printer driver to printer driver. Some printer drivers make intensive use of DLAY codes, especially in the printer’s "Reset" function. The use of this code is mysterious, since the data sent by the driver is usually not processed immediately, but temporarily stored in the printer’s buffer memory. This makes all delays useless, since the rate at which the printer reads and processes any data from its buffer is not linked to the timings defined by the computer, but depends on the speed at which the mechanical components of the printer can be moved. Delays are also lost when the data output by the driver is sent to a file, as can be done with the "Cmd" command supplied by Commodore, or when a printer spooler is buffering the data stream.

Three constant codes are represented by printer drivers in a very particular format in their internal command-sequence tables. The codes are NOAV, DLAY and NUL (which is equivalent with the "(0)" FFDL sequence). The three decimal values 253, 254 and 255 (hexadecimal FD, FE and FF, octal 375, 376 and 377) are used in the printer driver’s command table to represent DLAY, NUL and NOAV respectively. These codes were probably chosen by the designers of the drivers when 7 bits seemed sufficient to represent the data to be sent to the printer. Higher codes were reserved for the drivers’ internal uses. This may explain why so many drivers are still 7-bit oriented. Unfortunately, the values 253, 254 and 255 are now excluded from the 8-bit codes which can be output by a printer driver to send a command to the printer.

The user does not need to bother about these three special codes, as all conversions are automatically made by the Printer Driver Modifier when a driver is loaded or saved. The three FFDL sequences "(253)", "(254)" and "(255)" cannot, however, be written in the editing gadget associated with a printer command. These three values cannot be sent to the printer, even if the printer recognizes them as part of valid commands. The driver would interpret the three codes as one of either DLAY, NUL (or "(0)") or NOAV. To avoid any confusion, an error message is displayed by the Printer Driver Modifier if one of the three special codes is used in a control sequence associated with a command. The three values can, however, be used normally in any control sequence associated with a character.

The three special cases mentioned above concern the command table. There is another peculiarity regarding the format used by printer drivers to store the control sequences in the character table. In the printer driver's table, the sequence of bytes associated with each character (codes ranging from decimal 160 to 255) or command is terminated with a byte whose value is zero. The so-called "zero-termination" of strings is a common practice in the Amiga environment, and is usually never a source of trouble. However, printer control codes often contain zero bytes which must be transmitted to the printer, rather than be interpreted as a string-terminator. As explained before, in the command table a zero is replaced with a single code whose decimal value is 254. A similar substitution in the codes associated with characters would be a cause of major problems, as the code 254 references a valid character in the character sets of most printers, and needs to be used as such. This problem was solved by expanding all zero codes (decimal 0) into the sequence '\'+ '0' (decimal 92 followed by decimal 48). This, however, led to another problem: how should the driver distinguish between a '\'+'0' sequence which should be converted to a zero byte, and another sequence which had to be sent "as is"? After all, it is not uncommon to send a backslash and a zero character to the printer. This second problem was solved by transforming all '\' (backslash, ASCII decimal code 92) characters into a sequence of two backslashes, all couples of backslashes into sets of four backslash characters, and so on. Furthermore, a zero byte followed by one or more other digits (range of ASCII codes from 48 to 57) needs an additional treatment. This drawback is caused by the fact that the parser of the printer device considers up to three digits following a backslash as a single octal code. In a similar case, a zero must be expanded into "\000" before it can be followed by other digits. Fortunately, users of the Printer Driver Modifier do not need to be able to handle these conversions, the explanation of which may sound like a very complex joke. The Printer Driver Modifier expands zero codes and backslashes followed by digits automatically when a printer driver is stored. It is important to know, however, that there can be such an expansion, as this may lead to an overflow in a character or in the entire character table, depending on the current encoding mode (section 11.5).

The content of the string gadget can be edited by the user as described in section 1.9.6 ("String Gadgets"). The maximum size depends on whether the fixed location or the floating location encoding mode is selected (sections 11.5.1 and 11.5.2). In the fixed location mode, the "Maximum" field (section
9.4.5
) indicates the maximum number of characters (in the printer's internal format) which the printer driver may store for that character or command. If the limit is 0 (zero), i.e. no characters can be

written, the gadget is disabled by the program, and cannot be selected.

Section 10.6 ("Check Definitions") explains how the program can automatically search the FFDL sequences written by the user for any errors. This is also verified before the driver is saved and when a new decoding mode is selected. Error messages are explained in appendix H.

9.4.3 The "Command" Field

This field is displayed if the command mode (section 10.7.1) is active. The internal short name for the printer command currently being processed is displayed here. The Printer Driver Modifier uses the same names used in the Amiga's documentation and in the "include" files of different programming environments.

For example, "aSHORP4" means "Condensed Fine On". This is one of the commands used to define the horizontal print pitch. The name itself comes from "Set HORizontal Pitch".

The "Function" field (section 9.4.6) always contains an extended definition of the command.

9.4.4 The "Character" Field

When the character mode (section 10.7.2) is active this field replaces the "Command" field. The Amiga default character image associated with the character currently being processed is displayed here.

9.4.5 The "Maximum" Field

As described in section 9.4.2 ("The Editing Gadget"), the value displayed in this field indicates the maximum number of characters which the printer driver can associate with the current character or command. This only happens if the fixed location mode is selected (section 11.5.1). In the floating location mode (section 11.5.2) a '-' (dash) sign is displayed in the field, as the limit depends on the total number of characters used by all control sequences associated with the commands or characters, rather than on the single sequences.

Section 10.6 ("Check Definitions") contains some additional information on the size occupied by the driver's internal representation of the codes.

9.4.6 The "Function" Field

This field, which is displayed only in the command mode, contains the extended name of the printer function currently being processed. This name can be, for example, "Italics On" or "Condensed Fine On", just as it normally appears in the printer's documentation.