0050

COLLABORATORS			
	TITLE : 0050		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		August 8, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

0050

1.1 Personal Fonts Maker - 11. PDM: The Preferences Menu

```
                11.       PDM: The Preferences Menu

   The "Preferences" menu of the Printer Driver Modifier, like the menu
having the same name in the Personal Fonts Maker, contains the functions
used to set the program parameters.


11.1       Code Table

   When this function is first selected, a help window with a table of the
first 33 ASCII code abbreviations is displayed. The constant-codes which
are displayed are the same of those which are described in section 2.7.1
("FFDL Constants") and in appendix B.

   The window can be dragged with the mouse when the title bar is
selected, just like the main window. The window remains visible until this
function is selected again, the close box of the window is selected, or
the program terminates. The Printer Driver Modifier can be used normally
while the window is displayed.


11.2       Icons

   This option is identical with the "Icons" parameter of the Personal
Fonts Maker, described in section 7.8. When this menu item is
"checkmarked" (section 1.9.8, "Menus"), the program associates a
graphical Workbench icon with the files saved through the "Save
Definitions" option (section 10.4).

   Icons are never associated with printer driver files. Unlike the
Personal Fonts Maker, the Printer Driver Modifier does not have a "current
character image", therefore it is only possible to use the program's
default icon image.


11.3       Comments

   This parameter determines the format used to store the printer control
```

sequences with the "Save Definitions" function (section 10.4). If the "Comments" option is set, a comment precedes each line containing a printer control sequence in the file with the printer definitions. This may look like the following example:

```
PDM DEFS
; Reset
    COMM 0 = DLAY ESC \@ DLAY
; Initialize
    COMM 1 = NOAV
; Line feed (LF)
    COMM 2 = NOAV
; CR+LF
    COMM 3 = CR LF

...

; ' '
    CHAR 160 = SP
; '¡'
    CHAR 161 = (173)
; '¢'
    CHAR 162 = (155)
```

The texts used to comment printer commands are the same as those which are displayed in the "Function" field (section 9.4.6). These texts may vary depending on the current user interface language (section
11.6
).
Comments are particularly useful if the definitions are to be loaded with a text editor (or word processor), without the aid of the "Function" field of the Printer Driver Modifier.

If the "Comments" option is not set, the printer definitions are saved without comments. Printer definitions saved in this format occupy much less space, but are less readable once they are loaded with a text editor.

11.4 Decoding Mode

As described in section 2.7.1 ("FFDL Constants"), there are several ways to represent constant codes. Some methods can be preferred to others for compactness or readability. The user can write an FFDL sequence in the editing gadget freely mixing all different formats.

This parameter determines how the Printer Driver Modifier is to represent the constant codes extracted from the printer drivers' control sequences. The decimal code 27, for example, may be written as:

```
ESC
```
or
```
(27)
```
or
```
(0x1B)
```

The parameter can also be modified after a printer driver has already been loaded. In this case, the program first verifies if the control sequences associated with the commands and characters are error-free (section 10.6). If this is the case, all codes are converted to the new format.

Different menu subitems can be used to select the different combinations of this parameter’s settings. The following subsections explain the options in detail.

11.4.1 Alphanumeric Symbols

If this option is selected, the Printer Driver Modifier uses single ASCII characters, preceded by a backslash, to represent the codes from 33 to 126 (decimal). The code 65, for example, would be represented as "\A" (without quotes) in this mode. Otherwise, if the decimal constants (section 11.4.4) were selected, the same code would be written as "(65)".

11.4.2 Codes 0-7

This flag determines whether the codes from 0 (zero) to 7 are to be represented with the associated ASCII names from NUL to BEL. By default this mode is not set, as the codes in the range from 0 to 7 are often used as numerical parameters to printer control sequences. These values are usually more readable if they are not converted to ASCII symbols.

11.4.3 Codes 8-32

If this flag is set, all codes from 8 to 32 are converted by the Printer Driver Modifier into the equivalent ASCII control names from BS to SP.

Section 2.7.1 ("FFDL Constants") and appendix B contain all the codes. Section 11.1 ("Code Table") explains how to display a help window containing these codes.

11.4.4 Decimal Constants

This option, like the following two, determines the (mathematical) base to be used to represent numbers between parentheses. Again, it should be noted that the user can write the constants in any format. This option only determines the format to be used by the program when codes are converted automatically into FFDL sequences.

The decimal format is the default mode of the Printer Driver Modifier.

11.4.5 Hexadecimal Constants

When this mode is set, constants represented as a number between parentheses are written in base 16 (sixteen). The number 27 (base 10), for example, becomes "(0x1B)" in this format. The "0x" prefix is used by the FFDL to identify hexadecimal numbers.

11.4.6 Octal Constants

This option causes numbers to be written in base 8 (eight). The number 27 (decimal), for example, becomes "(033)" in this format. The leading ’0’ (zero) character is used by the FFDL to identify octal numbers.

11.5 Encoding Mode

This parameter determines the format in which the tables containing the control sequences associated with the commands and characters are to be stored inside the printer driver when the driver is saved.

The Printer Driver Modifier, as the name implies, can be used to modify the control sequences of an existing driver. The sequences are stored inside the driver. Not all drivers reserve the same amount of memory for the tables of commands and characters. The Printer Driver Modifier cannot increase the total space dedicated to the control sequences, but different options allow the user to decide how to make best use of the existing space.

The editing gadget (section 9.4.2) allows the user to edit a control sequence as long as 256 characters. This is more than enough to describe a printer’s control using the FFDL syntax. Section 9.4.2 explains how much memory is occupied by the driver’s internal representation of the codes. The Printer Driver Modifier verifies whether the control sequences fit in the driver’s memory only when the "Check Definitions" function (section 10.6) is executed, either explicitly (selecting the command manually) or implicitly (before a "Save Driver" operation or a change to the "Decoding Mode" parameter"). If the memory of a printer driver is not sufficient to contain the new control sequences, a driver with more memory should be used as a point of departure.

The encoding mode can be changed at any time. The parameter is checked by the Printer Driver Modifier only when a driver is saved. If the program detects that the control sequences in the printer driver which was loaded are not stored contiguously, only the "Fixed Locations" mode (section

11.5.1
) can be selected.

The following subsections explain each of the possible options offered by the Printer Driver Modifier to use the printer driver’s memory.

11.5.1 Fixed Locations

In this mode, the maximum number of bytes which can be occupied by a control sequence is determined by the original size of the sequence. This means that a sequence associated with a command or a character can be edited with the Printer Driver Modifier as long as the size of that sequence is not increased. The size is not measured after the length of the FFDL sequence, but after the driver’s format in which the sequence is converted by the Printer Driver Modifier.

The name "Fixed Locations" means that even after the control sequences are modified, made shorter or cleared, the starting position (location) of each sequence within the driver is the same as it was before the driver was modified.

This technique does not modify the driver’s internal table of pointers to the control sequences. In some cases, for example if a control sequence in the middle of the table is made shorter, a "hole" is created, which may make it impossible to modify the driver again in the "Floating Locations" mode (section 11.5.2).

11.5.2 Floating Locations

A driver has one chunk of memory where the control sequences associated with the commands are stored, and one chunk for the control sequences associated with the characters. In the floating locations mode it does not matter how much memory is occupied by the single control sequences, as long as the sum of all the command sequences fits in the memory reserved for the command table, and the total memory occupied by all control sequences associated with the characters does not exceed the available memory for the character table.

In this mode, the program always stores the control sequences one after the other. If one sequence is reduced by the user, more space becomes available for the other sequences. The "Check Definitions" function (section 10.6) can be used to control whether the limits of either the command table or the character table are being exceeded. When the driver is saved, any free space remaining at the end of the tables is filled with zero-bytes.

11.5.3 Strings on Word Boundary

This flag affects the way control sequences are stored in the "Floating Locations" mode (section 11.5.2). Normally, the distance between control sequences in the same table (commands or characters) is an even number of bytes. The distance is measured from the beginning (or end) of one sequence to the beginning (or end) of the following (or preceding) control sequence. All sequences (starting with the first in each table) are stored on even byte positions, i.e. they are word-aligned.

In general, aligning data on word boundaries is a good programming practice on the Amiga computer. In some cases, this is the only way to store data, depending on what hardware is to process the data. In practice, printer drivers seem to work well even if the control sequences are not word-aligned. This saves the pad bytes which have to be inserted if a sequence would otherwise not be word-aligned.

If this option is disabled, the Printer Driver Modifier does not insert pad bytes to force all control sequences to be word-aligned. This gives the user some extra memory for control sequences.

11.6 Language

This is the same as the "Language" parameter of the Personal Fonts Maker, described in section 7.12.

11.7 File Requester

The file requester of the Printer Driver Modifier is the same file requester which is used by the Personal Fonts Maker, which is described in section 3.23. Three menu items allow the user to control some details of the file requester.

11.7.1 Expand Path

This option works like the one in the Personal Fonts Maker, described in section 7.11.1

11.7.2 List Icons

This option, identical to the one with the same name in the Personal Fonts Maker program, is described in section 7.11.2.

11.7.3 Double-Click

This is the same as the "Double-Click" option of the Personal Fonts Maker, described in section 7.11.3.