Printers

PJ Hutchison

COLLABORATORS

	TITLE : Printers		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	PJ Hutchison	July 25, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Printers

1.1 Contents

A Guide to Printers on the Amiga

by Peter J Hutchison

1.2 Workbench Preference Drivers

From the very first Workbench, printer drivers have been provided for the range of printers that were available at that time. In the 1980s, the majority of printers available were dot-matrix and laser printers.

Installation is very simple:

Workbench 1.3

Run a CLI or Shell and insert the Extras disk. Copy the files in devs/printers to devs:printers on your Workbench disk. Run the Prefs/ Printer program and select the printer required and Save the new preferences.

Workbench 2.0

Insert the Extras disk, open the disk icon to show its contents, from the Windows menu select Show/All Files. Find the Devs/Printers directory and drag the appropiate printer driver files on to your

Workbench disk in devs/Printers. Run the prefs Printer program to select the printer and save. Run the PrinterGfx prefs program to change the graphics preferences.

Workbench 3.x
Do the same as WB 2.0 but the printer drivers are located on the Storage disk.

Upto WB3.1, the following printers are supported:

```
  CalComp_ColorMaster              CalComp_ColorMaster2
  CanonBJ10                        CBM_MPS1000
  Diablo_630                       EpsonQ
  EpsonX                           EpsonXOld
  Howtek_Pixelmaster               HP_DeskJet
  HP_DeskJetOld                    HP_LaserJet
  HP_PaintJet                      HP_ThinkJet
  ImagewriterII                    NEC_Pinwriter
  Okidata_293I                     Okidata_92
  Okimate_20                       PostScript
  Seiko_5300                       Seiko_5300a
  Sharp_JX-730                     Tektronix_4693D
  Tektronix_4696                   Toshiba_P351C
  Toshiba_P351SX                   Xerox_4020
  CanonEpson                       StarLC10
```

Limitations of the preference drivers:

a) No new printers since 1992 have been added to the printer drivers esp. new Laser and Inkjet or Bubblejet printers.
b) Limited to a maximum of 4096 colours, so true colour pictures are not possible.
c) Workbench does not support PC GDI printers or other 'Windows' only printers.

1.3 Public Domain Drivers

Since no new printer drivers have been provided by Commodore or Amiga Technologies, then the Amiga developers and users have written their own Printer drivers. Information on writing a printer driver is available in the Amiga Rom Kernal Reference Manual : Devices (p.196).

Some new printer drivers have been released to the public and its a case of finding them. Be wary though, these drivers may not have been thoroughly tested and may cause unpredictable results. See docs with the drivers for further information.

The following suppliers are known suppliers of printer drivers:

Undergound PD, 54 Carmania Close, Shoeburyness, Essex SS3 9YZ.
Tel 01702 295887. (HP, Cannon, Epson and others)

PDSoft, 217-219 Hamstel Road, Southend-on-Sea, Essex SS2 4LB.
Tel 01702 306060 or 306061. (Panasonic, HP, Canon, Citizen, Epson etc)

Software 2000, 8 Falcon, Wincote, Tamworth B77 5DN. Tel 01827 287377
Software 2000, 9 Wills St, Lozells, Birmingham B19 1PP. Tel 0374 678068
(Star, Seiko, Panasonic, Citizen, Canon and others)

Ask your nearest or favourite PD supplier to see if they have the printer driver you require! If you have a modem, try calling your local BBS and see if they have any on their file base. If you have internet access, you can try Aminet sites (text/print) to download drivers or you can call the printer manufacturer’s sites such as:

http://www.europe.canon.com/
http://www.hp.com/
http://www.epson.com/
http://www.panasonic.com/

You can also download some of the popular drivers from my home page at:

http://www.blizzard.u-net.com/

1.4 Commercial Printer Drivers

The latest commercial software for the Amiga such as Word processors, Desktop Publishing and Paint packages do provide there own printer drivers. Some may be additional preference drivers or drivers specific to that program.

There are two software packages available that replace the preferences drivers altogether and provide new features for today’s high speed, colour printers:

a) Print Studio 2.15
Cost: 49.99
Suppliers: First Computer Centre, Power Computing
List of Supported Printers

b) Turbo Print 5 (new version)
Cost: 49.99
Suppliers: Wizard Developments
List of Supported Printers

c) EnPrint (for Epson Stylus Printers)
Cost: 29.99
Supplier: Eyetech
List of Supported Printers

These provide a wide range of drivers for old and new printers and have new features that overcome the limitations of Workbench printer drivers such as true colour printing and supports higher resolution output. Both of these cost fifty pounds (you can get it for less if bought with a printer).

1.5 Supported Print Studio Printers

```
The following is a list of currently supported printers.

AMS, Amstrad, Bizer, Brother, Bull, C.Itoh, Calcomp, Canon,
Chipset, Citizen, Commodore, Compaq, Craft, Dataproducts,
Datatrade, DEC, Epson, Facit, Fordata, Fujitsu, GCC, Genicom,
Hewlett Packard, IBM, Kodak, Kyocera, LAserMaster, Mannesmann,
MicroPlex, Minolta, NEC, NewGen, Oce, Okidata, Olivetti,
Panasonic, Philips, PrePress, QMS, Qume, Ricoh, Samsung, Sanyo,
Seikosha, Sharp, Siemens, Star, Texas, Triumph Adler, Vobis,
Unisys and Zygal.

Updates are freely available on Aminet in biz/patch and supply
drivers for other packages besides Workbench drivers.

Who to contact:
CompuServe: Wolf Faust 100116,1070
Internet:   100116.1070@compuserve.com
```

1.6 Supported Turbo Print Printers

```
The following is a list of currently supported printers.

Brother_24-Pin, Brother_9-Pin

Canon_BJ Serie, Canon_BJC210, Canon_BJC240 incl. Photo Cartridge,
Canon_BJC4000, Canon_BJC4100, Canon_BJC4200 incl. Photo Cartridge,
Canon_BJC600, Canon_BJC600e
Canon_BJC610, Canon_BJC620, Canon_BJC70, Canon_BJC800
Canon_LBP Laserdrucker, Canon_PJ1080A

Citizen_120D+, Citizen_120D, Citizen_PRINTiva600c
Citizen_Swift24,240, Citizen_Swift9

Epson_EX,FX,LX Serien, Epson_LQ,SQ Serien, Epson_Stylus
Epson_Stylus820, Epson_StylusColor, Epson_StylusColorII
Epson_StylusColorIIs, Epson_StylusPro(XL), Epson_StylusColor200,
Epson_StylusColor500

Facit_B3450
Fargo_Primera, Fargo_PrimeraPro, Fargo_FotoFUN!
Fujitsu_DL-1100

HP_DeskJet, HP_DeskJet1200, HP_DeskJet500, HP_DeskJet500C, HP_DeskJet520
HP_DeskJet540C, HP_DeskJet550C, HP_DeskJet560C, HP_DeskJet600C
HP_DeskJet660C, HP_DeskJet690C / 694C incl. Photo Cartridge
HP_DeskJet870Cxi,HP_DeskJet850C
HP_LaserJetII, HP_LaserJetIII, HP_LaserJetIV (l,p), HP_LaserJetV(l,p)
HP_PaintJet,PaintJetXL, HP_PaintJet300XL

Lexmark_ExecJetIIc
MannesmannTally_7400
Nec_Pinwriter
```

```
Okimate20, Oki_ML-38x, Oki_ML-39x
Panasonic_KX-P1124, Panasonic_KX-P1540
Seikosha_24-Pin, Seikosha_9-Pin, Seikosha_SL-80AI, Seikosha_SL-80IP
Star_9-Pin, Star_LC,XB-24, Star_LC-10, Star_SJ144
```

A Demo and updates are available from Irsee’s home page:

http://home.t-online.de/home/irseesoft/

1.7 Enprint Supported Printers

This is a list of supported Epson printers:

Epson Stylus COLOR, COLOR II, IIs, 820, Pro, Pro XL, 500.

Other printer drivers included are:
Epson Stylus 300, 400, 800, 800+, 1000, 1500

Updates are available for registered users.

Enprint 3.0 will soon be available for new Epson printers such
as the 600 and Photo.

Web page: http://www.endicor.com/

1.8 What do drivers do?

To print on a printer requires special codes which tell the printer about the text or graphics to print. There are many standards for these printer langauges such as Epson ESC sequences, Hewlett Packard’s PCL language and there’s the high end Postscript language for top of the range lasers.

Each printer has its own features and limitations and therefore each printer requires a driver to access these features from Workbench.

When printing something from a program, the output is usually sent to the PRT: device, this is handled by the printer.device located in the DEVS drawer.
Workbench supports a list of printer commands (see p.9-23 in the Workbench User Guide for these codes) and most Amiga programs use these codes for printing text or graphics.

Obviously, these codes are understood by Workbench only and do not relate to any specific printer. To convert them to a format the printer understands requires a driver for each different printer type. The driver then coverts the Workbench printer codes to real printer codes.
For example, if we had an Epson compatible printer (NB: <ESC> = Code 27):

Description	Workbench code	Epson code
Init Printer	<ESC>#1	<ESC>@
Boldface on	<ESC>[1m	<ESC>E
Italics on	<ESC>[3m	<ESC>4
Underline on	<ESC>[4m	<ESC>_1

and so on these are stored in a command table in the printer driver.
Special printer commands are dealt with a dospecial.c program which does specific things for certain commands.
To send graphics two other programs are written called render.c and transfer.c which sets up the printer, the graphics data and transfers this data to the printer.

1.9 Writing Printer Drivers

Writing a printer driver is more complex than you think and requires some programming knowledge, an assembler or compiler that can produce object code and access to the Amiga Development Kit on floppy or CD.
Information on writing a printer driver is available from:

Amiga ROM Kernel Reference Manual: Devices p196 - 245
Addison-Wesley 1992, ISBN 0-201-56775-X.
description of files and programs to write inc. examples of Epson & HP driver!

NDUK 3.1 Example source
You might be able to get hold of the example files in the NDUK kit to help you.

As information on writing drivers is scarce and new cheap drivers are even scarcer. I enclose some information for programmers wishing to write their own:

Files and functions needed for a driver

Starting to write a Driver

Text based Printer Driver section

Graphics based Printer Driver section

Testing a new Driver

1.10 Files and functions

To write a driver you need the following files:

macros.i	Contains printer device macros
printertag.asm	Contains printer specific characteristics such as density, char sets and colour.
init.asm	Open various libraries for the printer
data.c	Contains printer RAW commands and extended char set
dospecial.c	Printer specific special processing for commands such as aSLRM and aSFC
render.c	Processing for graphics output and fill buffer
transfer.c	Processing called by render.c to output the

```
                    buffer to the printer.
     density.c      Contructs proper printer density commands

Other requirements:

     PrinterSegment      Contains printer extended data structures
                         See devices/prtbase.h
     CommandTable        Converts ANSI data commands to printer
                         specific commands. See devices/printer.h
     ExtendedCharTable   Contains definitions for characters from
                         $A0 to $FF.
     ConvertChar()       Function to do character conversion for
                         one character to a combination of others.
     Render()            Graphics output function.
     Transfer()          Output buffer to printer.
     SetDensity()        Contructs density setting commands.

     Once all these data tables and functions are written they
can be combined to produce a printer driver that will reside in
devs:printers.
```

1.11 Starting a new Driver

```
  See device/prtdata.h which contains the PrinterSegment
and the PrinterExtendedData or PED. These are part of the
printertag.asm file.

  This consists of a printerName, an INIT function (run when the printer code
is loaded and used to open libraries and devices), an EXPUNGE function
(run when printer code is closed and resources can be freed), an OPEN function
(which is called by OpenDevice() call after prefs are read and the parallel/
serial port is opened. It must return 0 for ok or non-zero value for an error).
Finally, a CLOSE function (called by CloseDevice() and frees any resources).

  The pd_ variable passed to the init call is a pointer to the
PrinterData structure and is the same as the io_Device for
IO requests.

pd_SegmentData
   Points to the PrinterSegment containing PED.

pd_PrintBuf
   For use by the printer-dependant code.

(*pd_PWrite)(data,length);
    Interface to the primitive IO device where writes
    are double buffered. Data,length points to byte data
    to send and its length.

(*pd_PBothReady)();
    Waits for IO requests to complete, it is useful
    if the code does not use double-buffering. Use it in
    between successive pd_PWrites.

pd_Preferences
```

Copy of printer prefs obtained when printer opened.

1.12 Text based Printer section

This section is designed to convert ANSI x3.64 style commands to specific printer escape codes. There are two parts the CommandTable and the DoSpecial() routine.

The CommandTable is used to convert escape codes by simple substitution. The order of the codes must be in the same sequence as shown in devices/ printer.h.

If the code for your printer requires a decimal 0 use octal 376 (deciaml 256). Also, if no conversion is available, then place octal 377 (dec. 255) in its position. It also could mean that the function can be dealt with by DoSpecial.

The DoSpecial() function is used to implement functions that require more complex sequence of control characters sent to the printer. The DoSpecial() function requires these parameters:

command
  Points to the command table e.g. aRIN for initialise.

vline
  Points to value for current line position.

currentVMI
  Points to value for current line spacing.

crlfFlag
  Points to setting of add LF after CR flag.

Parms
  Contains parameters needed for ANSI command.

outputBuffer
  Points to buffer where converted command is stored.

All printers will require aRIN (init) command in DoSpecial(). Some commands occur in the CommandTable and DoSpecial such as superscript, subscript etc. When using the reset command (aRIS), then to avoid losing data it is best to define the PD->pd_PWaitEnabled = \375; in the DoSpecial() function which causes the printer to wait before reseting.

For the printertag.asm, the following values are required:

MaxColumns
  Max number of columns the printer can print across the page.

NumCharSets
  Number of character sets

8BitChars
  Pointer to Extended Character Table. If null, uses default table.

```
ConvFunc
  Pointer to character conversion routine. If null, no conversion req.

  The Extended Character table is for the codes $A0 to $FF. Valid expressions
are \011 for octal 11, \00 for null, \n where n is 1-3 digit number. A back
slash is represented by \\.

  The ConvFunc function may be needed to convert any character to a combination
of other characters. It should return -1 if no conversion done, 0 for no  ←
   characters
added or the number of characters added to buffer returned.

  ConvFunc requires the following parameters:

char *buffer - Pointer to a buffer
char c       - Character to be processed
int crlf_flag- CR/LF flag
```

1.13 Graphics Printer Section

```
  The graphics part of the driver consists of three parts: Render(), Transfer()
and SetDensity() functions. If the printer does not support graphics then
only the Render() function is required which returns an error (PDERR_NOTGRAPHICS).

Render()

  This is the main printer specific code which consists of seven parts or cases.
The Render() function requires four long parameters: ct, x, y and status.

Pre-Master Init (Case 5)

Parameters: ct    - 0 or pointer to IODRPReq struc passed to PCDumpRPort.
            x     - io_Special flag deom IODRPReq struc
            y     - 0

X contains the Density flags. Do not PWrite() during this case. When done return
PDERR_NOERR. SetDensity() is called here.

Master Initialisation (Case 0)

Parameters:  ct     - pointer to IODRPReq struc
             x      - width of picture in pixels
             y      - height of picture in pixels

X and y are used to determine buffer size, if alloc fails return PDERR_BUFFER
MEMORY. The buffer should also contains commands to set up the graphic dump
and for each colour pass. A reset should NOT occur in this case. If ct is used
then return PDERR_TOOKCONTROL to allow a graceful exit. Using two buffers with
AllocMem() allows double-buffering (see example).

Putting pixels in buffer (Case 1)

Parameters: ct - pointer to PrtInfo buffer
```

```
            x  - PCM Colour code (if printer is MCC_MULTI_PASS)
            y  - printer row number (0 to height-1)
```

This is passed a whole row of YMCB intensity values. These are passed to the Transfer() function. Return PDERR_NOERR at end. See device/prtgfx.h for PCM values.

Dumping a pixel buffer to printer (Case 2)

```
Parameters: ct, x - 0
            y     - Number of rows sent (1 - NumRows)
```

The data can be Run Length Encoded (RLE) if supported otherwise it should be white-space stripped (WSS). This involves scanning for the first non-zero value and data from this point is sent. This reduces print time.
The y value allows you to advance the paper a few pixel lines to prevent white lines appearing. Commands which appear in the data can be pre-processed eg. $03 to $03 $03. The error from PWrite() should be returned.

Clearing & init the pixel buffer (Case 3)

Parameters: ct, x, y - 0

Inits the buffer to value the printer uses for blank pixels (usually 0). This is printer specific and should include the control codes in the buffer before and after the data. This call is made before each Case 2 call.
Return PDERR_NOERR.

Closing Down (Case 4)

```
Parameters: ct - error code
            x  - io_Special flag from IODRPReq struct
            y  - 0
```

This is called at the end of the dump. Any buffer memory should be freed, you can check if any is allocated by checking the value of PD->pd_PrintBuf. You must wait for the buffer to clear by calling PBothReady before freeing them. Page orientated printers should be given an Eject command , this can be checked in SPECIAL_NOFORMFEED, if set don't eject page.
Also, PWrite() should be used to reset line spacing (to 6 or 8 dpi), set bidir. mode, set black text and restore any margins. Use PDERR_NOERR or error from PWrite() on return.

Switch to next colour (Case 6)

This call is made for printers that require colours to be sent in seperate passes ie has PCC_MULTI_PASS set.

Transfer() function

This dithers and renders an entire row of pixels passed to it by Render(). It has five parameters:

```
PInfo     - pointer to PrtInfo struct.
y         - row number
ptr       - buffer pointer
colors    - colour buffer pointer
```

```
BufOffset - buffer offset for interleaved printing.

Dithering may involve thresholding, grey-scale or colour dithering each pixel.
If the threshold is non-zero, the dither value is: PInfo->pi_theshold ^15.
otherwise it is *(PInfo->pi_matrix + ((y & 3) * 2) + (x & 3))
where x is PInfo->pi_xpos which is incremented for each pixel.

The function renders by placing a pixel in the buffer based on its dither value.
If intensity of the pixel is greater than the dither value (see above) then it
is placed in the buffer otherwise it is skipped and the pixel pixel is processed.

The ColourClass (PCC_BW, PCC_YMC etc), determines the rendering logic. Only PCC_BW
has no Colour dithering. For Thresholding the Black value should be
compared to the threshold value to see if it to be rendered. Printers with no
Black colour should use YMC colours to make black.

  For the Grey Scale Dithering, the black value is compared to the
dither value to render a black pixel (print Black or YMC colours). For the
Colour Dithering, black (if avail) then/or the YMC colours are compared
to the dither value to see if they should be rendered.

 Writing the transfer function in assembly can cut the printing time by half!

SetDensity() function

  This is called in the Pre-Master Init case and the density code is passed in
the density_code variable which is used for a user-specified or max
density. It should also handle narrow and wide tractor paper sizes.
  Densities below 80dpi should not be supported with the exception of the
original HP_Laserjet which has a minimum of 75dpi.

Printertag.asm

 For graphics printer the following values need to be filled:

MaxXDots  - Max number of dots across the page.
MaxYDots  - Max number of dots down the page (or 0 if using roll or form fed
            paper).
XDotsInch - Dot density in x (see SetDensity())
YDotsInch - Dot density in y (see SetDensity())
PrinterClass - type of printer (PPC_BWGFX etc)
ColorClass - Colour class (PCC_YMC etc)
NumRows   - Number of rows printed by 1 pass of the printer head.
```

1.14 Testing the driver

```
The following tests should be performed to ensure the driver is working
correctly.

1. B&W (threshold 8), grey scale and colour of the same picture.

2. Do a dump with DestX and DestY dots set to even multiples of XDotsInch
and YDotsInch eg a printer with a res of 120 x 144 dpi, a 480 x 432 dump
can be done. It should produce a 4 x 3 inch picture.
```

3. Do a colour dump at max wide at density 7.

4. Ensure dumps don’t use the text margins. To test this, set margins to 30 and 50 characters at 10 cpi and do a dump wider than 2 inches. If should dump a picture left justified.

5. A invisible setup can be used which involves printing non-printable text before starting the dump eg. CRs.

6. Finally, an image with lots of white space between objects should be dumped to test white-space stripping or RLE is working. The white space should be seperated by as many lines of white space as NumRows.

1.15 About the Author

Peter John Hutchison
Halifax, W Yorks.
Amiga 1200 WB 3.1
Blizzard 1230-IV (8Mb RAM)
Epson Stylus 500 (EnPrint)

If you wish to contact me about printers, printer drivers (or even other types of drivers) then send me a message to:

E-Mail: P.J.Hutchison@hud.ac.uk
          or PJHutch@blizzard.u-net.com

Fidonet: 2:250/366.24

Web page: http://www.blizzard.u-net.com/