GhostPDL

This document discusses configuration, compilation and running of the GhostPDL [1] family of products:*XPS* [1], *PCL5E* [2], *PCL5C* [3], *PCLXL* [4], and *HPGL/2* with *RTL* [6], additionally a scaled down *PJL* [5] interpreter is provided. The PDL products use the Ghostscript Graphic Library for graphics, imaging and driver support. GhostPDL may be configured with PostScript and PDF support but these PDL's are not documented here, see the Ghostscript documentation at `www.ghostscript.com/doc/` for information about these languages. The relevant ghostscript version used by the PDL's can be found in gs/base/version.mak. The GhostPDL product may be configured with PostScript and PDF support but that is not documented here.

GhostPDL is not intended to be a finished software application but a collection of software components that will ultimately be included in a software application or a real time embedded system. Generally the GhostPDL languages are compliant with the Genoa (now QualityLogic) Functional Test Suite and the Genoa Application Test suite. Also each language should run the corresponding Genoa Comprehensive Evaluation Test (CET) without egregious errors, the following text files in the pcl and pxl source code directories should be consulted for discrepancy details: Anomalies.txt (PCL), pxcet.txt (PCLXL), and pxfts.txt (PCLXL).

In this document and the software, PCL6 refers to technology that supports both PCL5 and PCLXL languages.

Quick Start for Unix environment with GCC.

```
# unpack the release and go to the release directory.
tar zxvf ghostpdl-xxx.tar.gz; cd ghostpdl-xxx
./configure
make pcl  # build pcl
make xps # build xps
```

Supported development environments

- Windows Vista, NT and '95 with Microsoft Visual C/C++
- Linux with gcc
- Unix environments without GCC

Unpacking the source

The source code will either be on CD-ROM (without any archiving or compression) or in a archived compressed format. Zip is used for Windows environments and compressed tar format for Unix environments. If you have a CD-ROM release you will want to replicate the directory structure from the CD-ROM to a development directory on a hard drive. The compile process does not require a special directory name. If you have obtained the archived compressed version use:

For Windows:

```
pkunzip -d <archive_name>
```

For Unix::

```
gunzip <archive_name.gz>
tar xvf <archive_name>
```

---

[1] PDL stands for Page Description Language

After unpacking the result should be a directory hierarchy which is briefly described next.

Brief Overview of Directory Hierarchy

xps Source code for the XPS interpreter.

pcl Source code for the PCL interpreter. Files beginning with "pc" handle pcl state, text and rules. "rt" files implement pcl's raster language RTL and the "pg" source code files are for HPGL/2.

pxl Source code for the PCL-XL interpreter. This is an entirely different language than PCL; all files in this subdirectory are prefixed with "px"

pl This subdirectory contains code that can be shared by PCL and PXL. In particular font scaling code, the printer job language (PJL) and the language switching machinery are in this subdirectory. All files in this directory are prefixed with pl.

common PCL and PXL shared makefiles. The subdirectory is an historical artifact from when PCL and PXL had separate makefiles. It does contain the makefile to build the ghostscript graphics library.

tools miscellaneous tools and test files.

tools/GOT GOT means graphical object tagging. The system is able to classify high level graphical objects (text, images and vectors) and store this information in the framebuffer along with the output data. The tools in these directories are used to unpack a framebuffer that has been tagged for display in a viewable format.

tools/viewer The Java pcl viewer is a sample graphical user interface for the GhostPCL system.

language_switch top level makefiles for building a share language environment. This is where the shared language environment which includes *PostScript* and PDF can be built.

gs Ghostscript branch, refer to the documentation at `www.ghostscript.com/doc/` for more information about ghostscript.

main The "workhorse" makefiles of the system. These are also most likely to need modification to add/remove devices, choose a font scaler, and set directory locations.

urwfonts 80 URW TrueType fonts.

win32 Microsoft Visual C project files and instructions (ReadMe.txt) how to use them.

This directory structure is the default, but the directories can be rearranged with minor modifications to the makefiles.

Building with Microsoft Visual C/C++

The recommended way to build is to use the project files found in the subdirectory win32, there is a ReadMe.txt file containing instructions. The rest of this section may prove helpful if you wish to wrestle with Microsoft's NMAKE.

The GhostPDL tools are known to build with Visual C/C++ 4.0, 5.0 and 6.0, Visual Studio 2005 and 2008. [2]. Automatic makefile wrapping to projects file stopped working some time ago, we aren't sure which Visual Studio version deprecated the functionality.

---

[2] A subset of the components have been known to build with these tools

There are 2 ways of building the PCL components with Microsoft Visual C/C++: Convert the current makefile environment to a Visual C/C++ project; or compile the source directly using nmake. The following steps take you through converting the supplied makefile environment to a Visual C++ project:

- Unpack the source code, see Unpacking the source.
- Start MSVC++.
- Open main/pcl6_msvc.mak and MSVC++ should wrap the makefile automatically and create an MSVC project.
- Set the project setting (Alt F7). To create a debug:

  `NMAKE /f pcl6_msvc.mak DEBUG=1 DEVSTUDIO=c:\progra 1\micros 3`[3]

  Now set the name of the executable for debuggging to .\obj\pcl6.exe, and set the program argument to be any pcl6 options wanted and the name of the pcl file.
- Now PCL6 is set to be compiled, debugged and use other features of the MSVC++ IDE.[4]

This will build both the PCL and XL language and supporting language switching code.

Building with Linux and GCC

The pcl tools have been compiled on Linux using GCC. It is easiest to simply use the instructions provided in the section Quick Start For Unix environment with GCC.

Building on a generic UNIX platform

The PCL tools have been compiled on Solaris with the Sun Development Tools and SGI with the native SGI C compiler. We only provide makefiles for the gcc tools and assume users can customize the gcc makefiles such that they work with vendor's compilers. For the Sun Development tools the following workaround is sufficient to build the software:

- change to the main subdirectory.
- `make CC_=cc CCLD=cc \ CCAUX=cc CFLAGS=-g CC_NO_WARN=cc GCFLAGS=`

If you intend to do development using the software a dedicated makefile should be constructed.

Customizing the build process

The build process is completely configurable. Here is a list of things that are user will most like be interested in customizing. To change any of these see the appropriate makefile for your platform in the "main" subdirectory

- Directory location of source directories.
- Directory location of objects, executables, and other compile time generated files.

---

[3] DEBUG=1/0 on/off will require you manually clean; del main\obj\*.*

[4] Dos hackers can start up a dos window, set the DEVSTUDIO environment variable, and use the same make commands as above (the DEVSTUDIO variable should be set to the top level of the Microsoft Developer Studio, e.g. set DEVSTUDIO=c:\progra 1\devstu 1" or NMAKE /f pcl6_msvc.mak DEBUG=1 DEVSTUDIO=c:\progra 1\micros 3)

- Selection of devices.
- Directory location of PJL filesystem volume 0 and 1
- The font scaling technology.
    - Defaults to /tmp/pjl0 and /tmp/pjl1
    - Edit pl/pl.mak PJL_VOLUME_0 and PJL_VOLUME_1 to match desired root

All of these can be configured in the top level makefile or can be specified on the make command line.

Sample make targets include:

debug build tools with symbols and debugging information

product builds optimized code.

pdl-pg builds profiling.

For a complete list of targets see the top level makefile named Makefile. Each target is somewhat self explanatory.

Building only one Language

PCL or PCL-XL can be built together in a language switching environment or each can be built alone with the supporting PJL interpreter. The simplest way to implement one language is simply to remove the unwanted implementation from the pdl implementation table located in pl/plimpl.c:

For example, this is the default table with two implementations: PCL and XL.

```
/* Zero-terminated list of pointers to implementations */
pl_interp_implementation_t const * const pdl_implementation[] = {
        &pcl_implementation,
        &pxl_implementation,
        0
};
```

If you only wish to use one interpreter remove the unwanted one and recompile the code.

Running the products.

> Few things are harder to put up with than a good example. *Mark Twain*

Most ghostscript options, as described in the Ghostscript documentation `www.ghostscript.com/doc/`, have similar effect in the GhostPCL system. Of course, options specific to the PostScript or PDF language are not relevant and are ignored.

pcl6 mypcl.pcl

Interpret a pcl file called mypcl.pcl and render it to the default device, usually a simple display is the default, X11 on Unix like systems and the Window's display device on Windows.

pcl6 -dTextAlphaBits=4 mypcltext.pcl

When rendering pcl text on a low (screen) resolution display device, use the TextAlphaBits option to enable anti-aliasing.

```
pcl6 -sDEVICE=ljet4 -sOutputFile=”| lpr” -dNOPAUSE mypcl.pcl
```

Interpret mypcl.pcl and send the Laserjet 4 formatted output to the command lpr.

```
pcl6 -sDEVICE=pcxcmyk -sOutputFile=”pcxpage.%d” -dNOPAUSE mypcl.pcl
```

Interpret mypcl.pcl and generate CMYK output. Pages are to be put in files named pcxpage.1, pcxpage.2, pcxpage.3, etc.

```
pcl6 -r72 -sDEVICE=x11mono mypcl.pcl -r100 -sDEVICE=x11 mypcl.pcl
```

Render a pcl file at 72dpi on the monochrome X11 device, then render the same file at 100 dpi on color X11 device. This demonstrates on-the-fly device switching.

```
./pcl6 -J”@PJL SET SYMSET = ISOL1” mypcl.pcl
```

PJL or PCL Job control commands can be set directly on the command line. This example sets the default symbol set to ISO Latin 1.

```
pcl6 -sDEVICE=pdfwrite -sOutputFile=mypcl.pdf mypcl.pcl
```

Convert the pcl file mypcl.pcl to PDF with output written to mypcl.pdf.

```
pcl6
```

Simply running the interpreter should generate some useful information about the available options and devices.

PCL Personality

The PCL emulation comes in three flavors: PCL5E, PCL5C, and RTL. The PCL5E personality thresholds colors to black and white irrespective of the color parameters of the output device. PCL5C is the color personality, used with a monochrome device it will grayscale colors. The RTL personality can be used to print HPGL/2 RTL plot files.

```
pcl6 -lRTL myrtl.rtl
```

run the interpreter with the rtl personality.

```
pcl6 -lPCL5E -sDEVICE=ljet4 mypcl.pcl
```

run the interpreter with the pcl5e personality. This will threshold colors to black and white (ljet4 is a 1 bit device).

```
pcl6 -lPCL5C -sDEVICE=ljet4 mypcl.pcl
```

run the interpreter with the pcl5c personality. This will grayscale colors on the 1 bit output device. If not set on the command line the pcl interpreter personality will be set to PCL5E if the output device is 1 bit per pixel otherwise it is set to PCL5C. RTL must be explicitly set on the command line. RTL always grayscales and never thresholds colors to black and white.

```
pcl6 -H12x12x12x12 mypcl.pcl
```

apply hardware margins of 1/6 inch, the parameter’s units are points. Without this or specification of hardware margins from a Ghostscript device the margins will be 0 or full bleed. Most HP printers have a hardware margin greater than 0. Normally this would be expressed by setting a device parameter on the command line but PCL does not parse this type (array) parameter yet.

Fonts

The release is packaged with 80 high quality URW TrueType fonts. For commercial use of the GhostPCL technology these fonts can be licensed from Artifex. The fonts are searched for in either the fonts, /windows/fonts, or a directory specified with the PCLFONTSOURCE environment variable. For historical reasons the directory path must be specified using forward slashes and must include a trailing slash.

Fonts and a font scaler from a third-party vendor such as Agfa or Bitstream may also be used. There is an existing interface for integrating the AGFA Universal Font Scaler Technology, several Artifex customers currently use this solution. The software can use Hewlett Packard FONTSMART version 1.5 or Windows TrueType fonts, using either of these font solutions require minor PCL code modifications.

PCL Code changes required to use other TrueType fonts.

To use a new set of TrueType fonts requires modifying the C code in the file pl/plftable.c. The C structure resident_table contains a list of Windows TrueType font names. In the released package these names will correspond with the Windows True Type font names in the URW font set. To use a different font set these names must be replaced with the new font names and the code (at least the plftable.c module) should be recompiled and linked. The file tools/fontpage.pcl can be run to display font samples and the pcl escape sequences required to select the fonts in a pcl stream. This option is only recommended for advanced developers. It is easiest to simply create a downloaded font and embed it in the PCL stream, see the PCL Technical Reference Manual [2] for details.

Using the language switching build

For printer and embedded device users we provide a complete language switching solution consisting of PCL/HPGL2, PCLXL, PS, PDF, with Job Control. For host based user we strongly recommend that you use GhostPCL and Ghostscript separately as the shared language has particular feature well suited to printer environment but the same feature may produce unexpected results on host based environments.

Building the Language Switch Environment

At this time, we have makefile support for Microsoft Visual C and Linux with gcc. For the Microsoft Visual C the use the instructions in the section Building with Microsoft Visual C, but this time you will wrap the makefile pspcl6_gcc.mak which is located in the directory language_switch. For Linux the makefile targets for the language switch build are exactly the same except each target is prefixed with "ls_". So using the pattern from the "Quick Start" section we have:

```
tar zxvf ghostpcl-xxx.tar.gz; cd ghostpcl-xxx  # unpack the release and go to the release directory.
make ls_fonts # install the fonts.
make ls_product #compile and link pspcl6.
make ls_test # test pspcl6 (optional).
make ls_install # install it.
```

Reporting bugs

If you find a bug or have comments about this documentation, please send mail to bug-pcl@ghostscript.com.

Trademark Credits

PostScript is a registered trademark of Adobe Systems Inc. *PCL* is a registered trademark of Hewlett-Packard Company.

References

[1] *XML Paper Specification v.1.0*

[2] *PCL 5 Printer Language Technical Reference Manual*, HP Part No. 5961-0509, First Edition - October 1992

[3] *PCL 5 Color Technical Reference Manual* Copyright 1999, Hewlett-Packard Company.

[4] *PCL XL Feature Reference.*

[5] *Printer Job Language Technical Reference Manual* Edition 10, HP Part No. 5021-0380, October 1997.

[6] *The HP-GL/2 and HP RTL Reference Guide A Handbook for Program Developers*, Addison Wesley Publishing Company, 1993.

`http://www.microsoft.com/whdc/xps/xpsspec.mspx`