Display Manager to FoxView conversion tool

Table of Contents

1 Overview

The application, unix scripts and information gathering provided here was motivated by the need to translate 1277 graphics from Display Manager to FoxView. The Invensys provided utility, `pdf_fdf` didn't quite solve all problems. These are outlined later in the document. Mind you, our version is at least 9 months old. A new version may leave part or all of this work obsolete.

The original development of the tools was purely aimed at my own use and was not intended to be a released application. It's designed to solve our specific set of issues at Lihir Gold based on our particular standards and conventions. It's almost guaranteed that you, as another user in a different part of the world will have different problems. However hopefully it will assist you with the problems we have in common.

When designing the application I knew that whenever I needed a new feature I could simply modify the code. After all the conversion is a one-off project. Hence I did not concern myself with many of the niceties of a properly designed application. So please keep this in mind.

If you know something about programming and have access to Visual Studio .NET 2003, make changes as you see fit. However I do ask that, if it's likely to be useful to others in the Foxboro community, let me know about your changes and I'll try to incorporate them into a later version.

1.1 Word of Caution – Please read

Getting things wrong here has the potential to stop your plant. This happened to us for reasons I still don't fully understand. In an earlier version the application failed to add the `.fdf` extension to the links. What I can best fathom is that when FoxView went to open the file and found that the link was not valid, it did a search for the file and then, finding that it was a Display Manager page, did a dynamic conversion to FoxView.

Anyway, it seemed to consume such a large number of resources that a watch-dog between a CP40 and an Allen Bradley Station timed out, tripping our grinding circuit. So please keep this in mind. This may have happened on another occasion when I accidentally rebooted an AW in the middle of a checkpoint.

1.2 Disclaimer

Of course this is a totally voluntary effort on my part to hand over my work and as I'm not a software guy (Elect Eng. actually) and only have 9 months experience with Foxboro (and little more with Unix) you ***must accept this code and information at your own risk***. Although I have undertaken some effort to test the code, ***I accept no legal or financial responsibility what-so-ever for any problems you experience as a result of using the program, code or documentation***. If this is not acceptable to you, please do not use the application or code and remove it from your system.

Unfortunately these seemingly obvious words seem to be a necessary part of our selfish and adversarial world. Apologises for the depression that may set it in, as it does with me when I see them.

Thanks and Regards
Adam Pemberton
Lihir Gold Limited
email: `adam.pemberton@lihir.com.pg`

2 Introduction

At Lihir Gold we had to convert approximately 190 pages and 1100 overlays from Display Manager. When we tried the `pdf_fdf` tool we found it did a lot of things right but not everything. In particular we had three main issues:

- Fonts – Some fonts are badly resized.
- Colours – not all colours are mapped correctly (and for those North American's reading this – Colour is the correct spelling!! ☺).
- Paths – The default location for files in FoxView is different from Display Manager.

2.1 Fonts

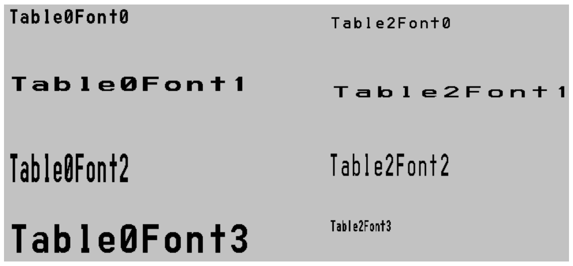


Figure 1: Display Manager font test

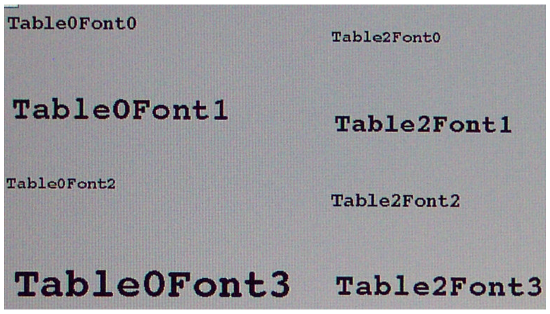


Figure 2: The font test in FoxView as generated by `pdf_fdf`.

Figure 1 and Figure 2 show the DM and FV equivalents for a font test. We use Table 2, Font 3 extensively as it's the smallest provided font. One can see that it ends up many times larger in FoxView. This was our first issue.

2.2 Colours

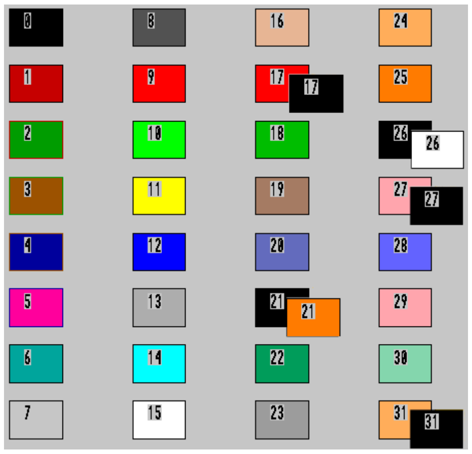


Figure 3: The 32 Display Manager colours. Both colours for the 5 blinking colours are shown.

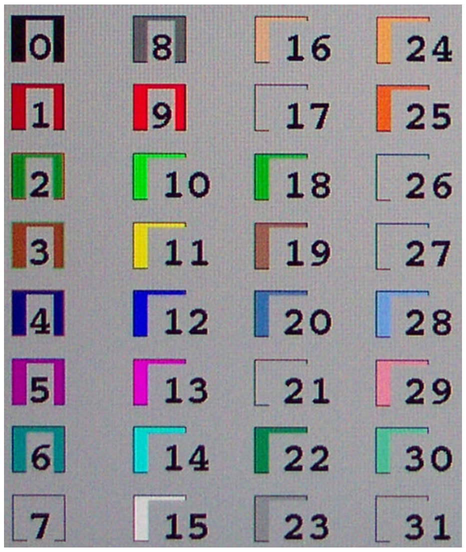


Figure 4: The test page from Figure 3 in FoxView as generated by `pdf_fdf`.

Colour 13 goes from grey to bright pink. And most of the blinking colours become incorrect although unfortunately you can't tell here.

2.3 Paths

The default location for Display Manager files for our site is `/usr/disp`, with overlays in directories such as `/usr/common`, `/usr/overlay`, `/usr/drive` and `/usr/valve`.

In FoxView it is recommended that pages go in `/opt/menus` and overlays etc go in `/opt/customer/`.

3 The Solution

Our system is I/A version 7.1.1 with 51F workstations (I think – they're Blade 150's). Hence if your using version 8, your actions will change slightly. But you should all be engineers…

The Invensys guys nicely provided the `.g` file format for editing every aspect of a display, including all the dynamic aspects. This file can be generated either by using the `-g` option (retain g file) in `pdf_fdf`, or using `fdf_g` (one can also use the FoxDraw utilities but we're scripting here so let's talk command line).

So our approach involves the following steps:

1. Generate the g files. This uses a unix scipt which you *will* have to modify in order to work at your site.
2. `tar`'ing and `ftp`'ing the files to a windows PC.
3. Running my conversion tool on the g files.
4. `tar`'ing and `ftp`'ing the files back to your AW/WP.
5. Generating the FoxView `.fdf` files from the modified `.g` files. Another script although I ended up using `g_fdf -r`. It's quicker.
6. Moving the new `.fdf` files to their final resting place.

3.1 *Generating the "g" files*

For this I wrote the script `do_pdf_g_all`. Credit goes to the FeedForward guys and their `xref` utility for teaching me a few basics about writing scripts.

1. Copy the script to your selected working directory.
2. Use `chmod` to turn it into an executable (`chmod 755 do_pdf_g_all).`
3. Create a directory, `g`, underneath your working directory.
4. Copy the file `dsps.dm` to the `g` directory. Edit this file to include the base path for all your Display Manager pages and overlays.
5. Execute the script

3.2 *Tar'ing and Ftp'ing*

3.2.1 Tar

For creating tar files on Windows, I downloaded GNU tar, which is readily available on the internet. For those a little rusty the relevant syntax is:

Creating a tar file:

`tar -cvf` <tar filename> <list of files or directories>

You can drop the – sign for unix.

eg: from the `/usr` directory

tar -cvf disp.tar disp

Extracting from a tar file

```
tar –xvf <tar filename>
```

eg: after placing the tar file in the directory you want to install the files:

```
tar –xvf disp.tar
```

3.2.2 Ftp

The only point to make here is remember to type `bin` before transferring tar files, and `ascii` before transferring the scripts.

3.3 The Conversion Tool – GConvert

3.3.1 Installation

Installation is simply a matter of extracting the `GConvert.exe` and `PathConvert1745.xml` files from the zip file to a suitable location.

The application does require .NET Version 1.1 or later. However if you are running XP, it should be installed.

3.3.2 Introduction

Starting GConvert, you see the window as shown.

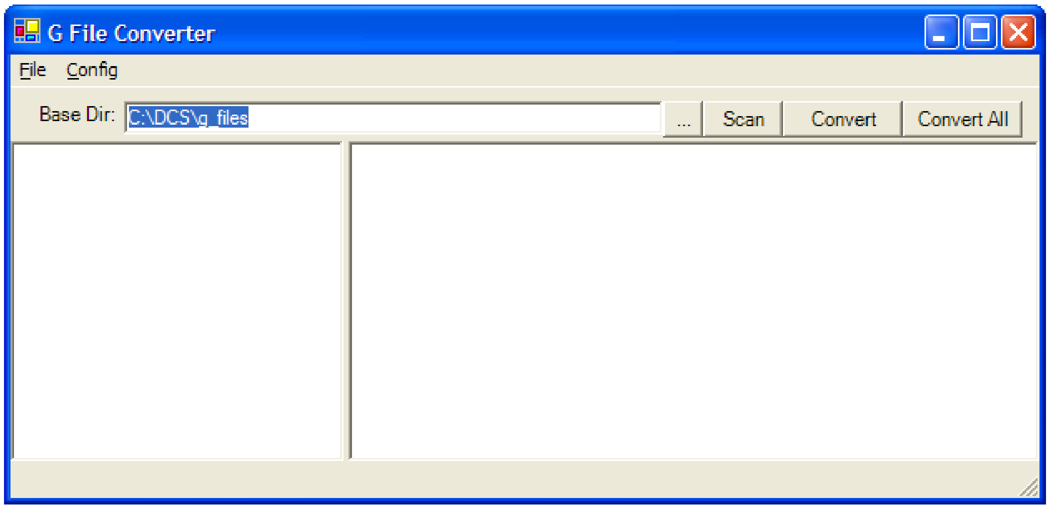


Figure 5: GConvert when started.

To get going, set the base directory for your `.g` files (type it in or browse) then hit the scan button. You will then see something equivalent to Figure 6. Initially you probably won't have an `opt` directory.

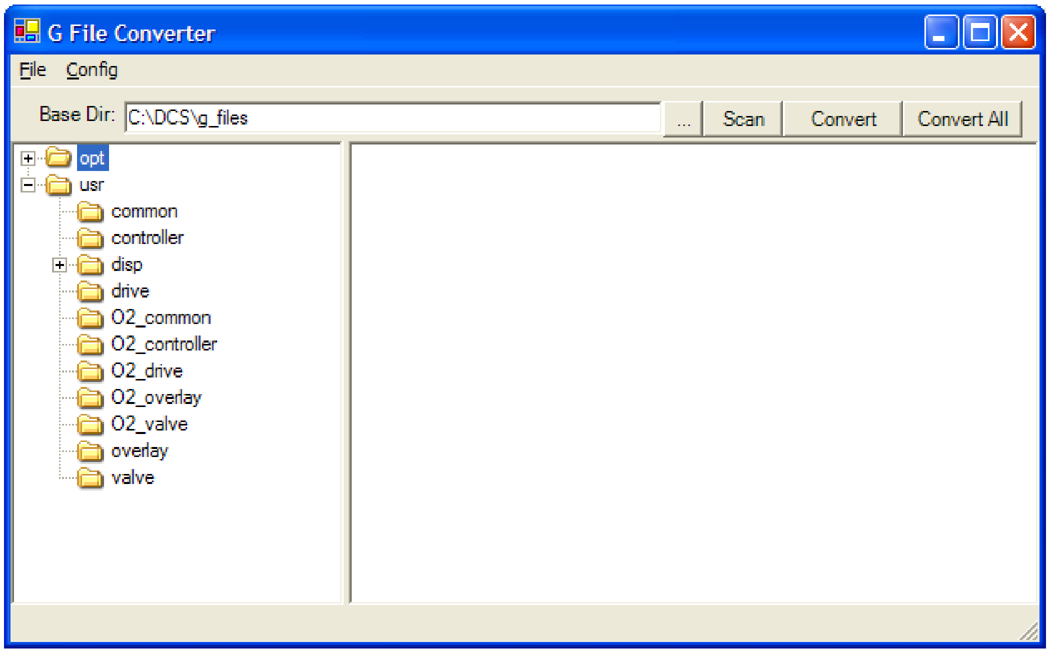


Figure 6: GConvert following a scan.

You then have the usual browsing capabilities.

After scanning one can choose to convert a single file ("Convert") or all files (Convert All). Pressing the Convert button will just convert the selected file, creating a new file in the same location with an extension `.1`, `.2` etc.

Pressing the Convert All button converts all `.g` files in the `usr` directory. My assumption here is that all who use this tool will be moving their files from the `/usr` location to the `/opt` location as recommended by the FoxView documentation. Hence conversion will only occur if there is a `usr` directory in the top level of the left pane following the Scan. If this is a problem with anyone, contact me.

Of course, we want to control how the conversion is done. For this we investigate the Config menu. However before we do that, it might be useful for some to state exactly what we are trying to achieve, in order to get some sort of context for the tedium of the next few pages.

3.3.3 The Purpose of `GConvert`

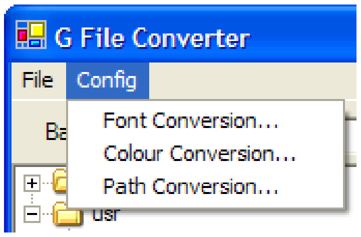
Remember our aim here is to tweak the `.g` files generated by `pdf_fdf` to rectify a few shortcomings such as incorrect colours and font sizes fonts and fixing up of paths. Rather than over-writing the `pdf_fdf` generated `.g` files, we'll generate new ones in a different location.

Part of this process involves identifying the fields in the `.g` files that correspond to particular aspects of the Display Manager pages and then altering them in a way that gives the best result in FoxView whatever that means to you. It's a mapping exercise. Mapping the Display Manager colours, fonts and paths to new values ready for FoxView.

What you will find unfortunately is that, at least in the case of font's, it's not a one-to-one mapping exercise. This is because, despite the fact that there are 8 unique alphanumeric font's in Display Manager, one can only identify six unique font's in the corresponding `.g` files.

3.3.4 Configuration

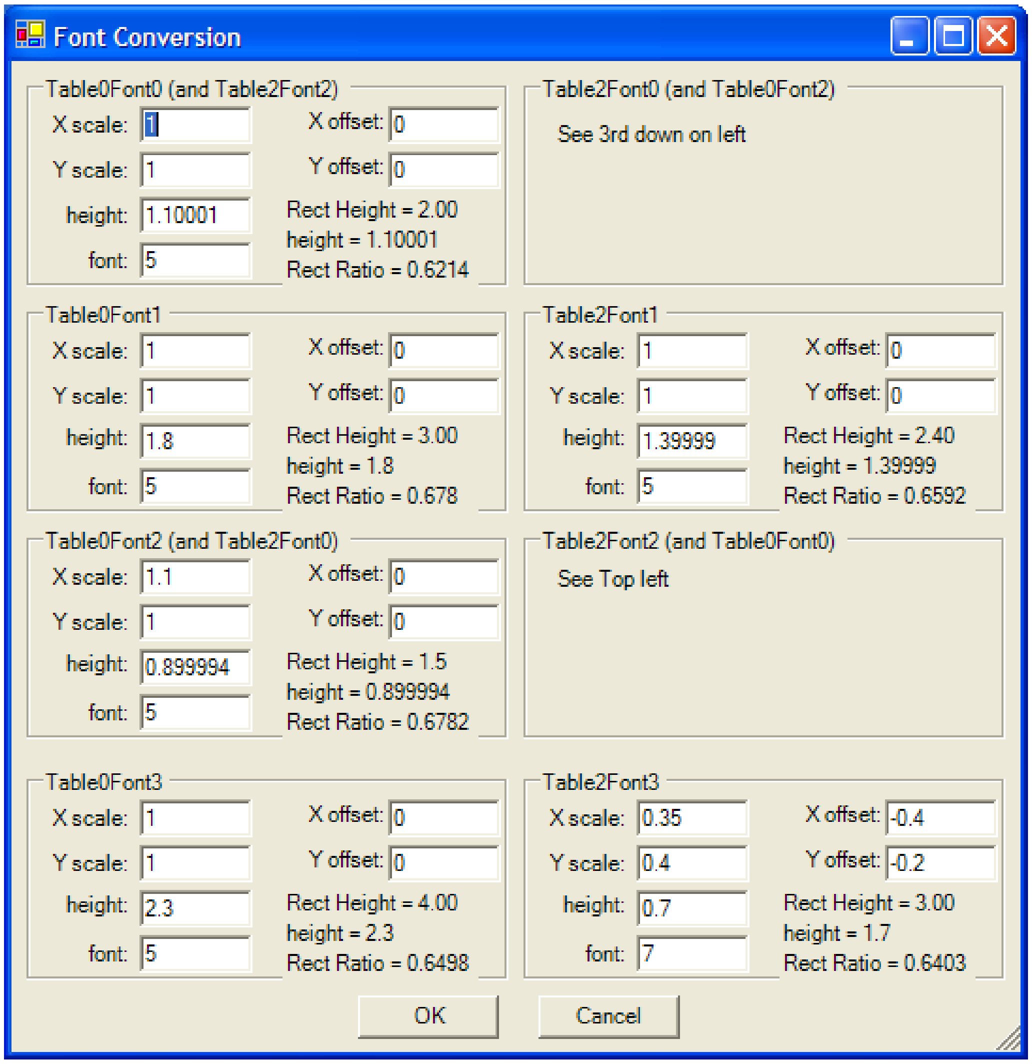
Examining the Config menu, you will find 3 options:



- Font Conversion – Controls how the font's are adjusted
- Colour Conversion – Controls colour conversion
- Path Conversion – You guessed it.

3.3.5 Font Conversion Configuration

Figure 7 shows the configuration parameters for fonts as configured for Lihir Gold.



Font Conversion

Table0Font0 (and Table2Font2)
X scale: 1 X offset: 0
Y scale: 1 Y offset: 0
height: 1.10001
font: 5
Rect Height = 2.00
height = 1.10001
Rect Ratio = 0.6214

Table2Font0 (and Table0Font2)
See 3rd down on left

Table0Font1
X scale: 1 X offset: 0
Y scale: 1 Y offset: 0
height: 1.8
font: 5
Rect Height = 3.00
height = 1.8
Rect Ratio = 0.678

Table2Font1
X scale: 1 X offset: 0
Y scale: 1 Y offset: 0
height: 1.39999
font: 5
Rect Height = 2.40
height = 1.39999
Rect Ratio = 0.6592

Table0Font2 (and Table2Font0)
X scale: 1.1 X offset: 0
Y scale: 1 Y offset: 0
height: 0.899994
font: 5
Rect Height = 1.5
height = 0.899994
Rect Ratio = 0.6782

Table2Font2 (and Table0Font0)
See Top left

Table0Font3
X scale: 1 X offset: 0
Y scale: 1 Y offset: 0
height: 2.3
font: 5
Rect Height = 4.00
height = 2.3
Rect Ratio = 0.6498

Table2Font3
X scale: 0.35 X offset: -0.4
Y scale: 0.4 Y offset: -0.2
height: 0.7
font: 7
Rect Height = 3.00
height = 1.7
Rect Ratio = 0.6403

Figure 7: The configuration page for Font Conversion.

To understand what the parameters are for, you'll need some background understanding.

3.3.5.1 Background

Consider a few lines from the `.g` file that came from the Display Manager page of Figure 1 and became the FoxView page of Figure 2.

```
    fcolor 23
    ecolor 23
    estyle 0
    tcolor 16
    height 1.10001
    path 1
    font 5
    prec 0
    align 1 3
    size 0 0
    ftrect 0.9961 71.6146 14.6691 69.6146  "Table0Font0"
    height 1.8
    ftrect 0.9961 64.2578 23.3701 61.2578  "Table0Font1"
    height 0.899994
    ftrect 0.9961 56.5104 12.1831 55.0104  "Table0Font2"
    height 2.3
    ftrect 0.9961 48.8715 29.5851 44.8715  "Table0Font3"
    height 0.899994
    ftrect 32.8711 70.2257 44.0581 68.7257  "Table2Font0"
    height 1.39999
    ftrect 32.8711 62.7604 50.2731 60.3604  "Table2Font1"
    height 1.10001
    ftrect 32.7148 55.4688 46.3878 53.4688  "Table2Font2"
    height 1.7
    ftrect 32.8711 48.1771 54.0021 45.1771  "Table2Font3"
```

This segment draws all of the lines of text. The way the g file protocol works, until a parameter reappears with a new value, its last value holds. Hence all the parameters prior to the first string hold for all strings, except `height`. To the best of my knowledge and crude testing, the relevant parameters to the text are:

- `fcolor` – Fill colour
- `tcolor` – Text color
- `height` – Text height
- `font` – Font number.
- `ftrect` – The background rectangle and text string itself.

We'll deal with the colours later. For the font itself, we are concerned with the font number, the height and the background rectangle dimensions.

The text line consists of the identifier `ftrect`, followed by four numbers representing the shape and location of the rectangle, and finishing with the actual text in quotes.

The rectangular dimensions are in the form `x1 y1 x2 y2` where the co-ordinate (`x1,y1`) is the upper left corner and (`x2,y2`) is the lower right corner of the rectangle. Units are percent of screen (nominally 100% is full width and 75% is full height).

For all of us who have worked with Table 2, Font 3 in Display Manager you will be well aware that the selection rectangle is, for what-ever reason, many times the size of the font itself. This immediately becomes apparent in the sizes of the rectangle from the test page, which is similarly very large.

Now our problem is that `pdf_fdf` screws up each of the DM font's in a different way. Hence we want to be able to identify which font was used for each text string in the .g file and adjust according to Figure 7. However this turns out not to be possible.

From the test of Figure 1 and Figure 2, we can determine the following statistics regarding how the DM font's are translated to the `.g` file.

Table 1: g file font statistics for files generated by pdf_fdf.

Font Table	***Font Number***	***height***	***Rect. Height***	***Rect. Ratio***
0	0	1.10001	2.00	0.6214
0	1	1.8	3.00	0.678
0	2	0.899994	1.50	0.6782
0	3	2.3	4.00	0.6498
2	0	0.899994	1.50	0.6782
2	1	1.39999	2.40	0.6592
2	2	1.1001	2.00	0.6214
2	3	1.7	3.0	0.6403

Note the following points:

- The height field represents the actual font height.
- The rectangular co-ordinates have no effect what-so-ever on the appearance of the text itself – only the background.
- The Rect Ratio is the ratio of the width to the height
- From the statistics **there is absolutely no way to tell the difference** between Font 0,0 (Table 0, Font 0) and Font 2,2 and also no way to tell the difference between Font 0,2 and Font 2,0. This is despite the fact that each of the two pairs of fonts appear very different in Display Manager.

We were fortunate at Lihir Gold in that our predecessors almost exclusively used two fonts: Font 2, 0 and Font 2, 3. So we configured the application to assume Font 2,0 when it comes across that statistics of Font 2,0/Font 0,2.

If you are less fortunate than us you will have to determine which font is more popular and manually adjust the less popular font.

3.3.5.2 Instructions

Hopefully the configuration parameters of Figure 7 are self explanatory.

You are able to adjust:

- The height of the font which is the single determining factor in it's size in FoxView.
- The offset. We found this was necessary with Table 2, Font 3.
- The size of the rectangle (X Scale, Y Scale)
- The Font number. The tool, `pdf_fdf`, substitutes font 5 for all fonts in DM.

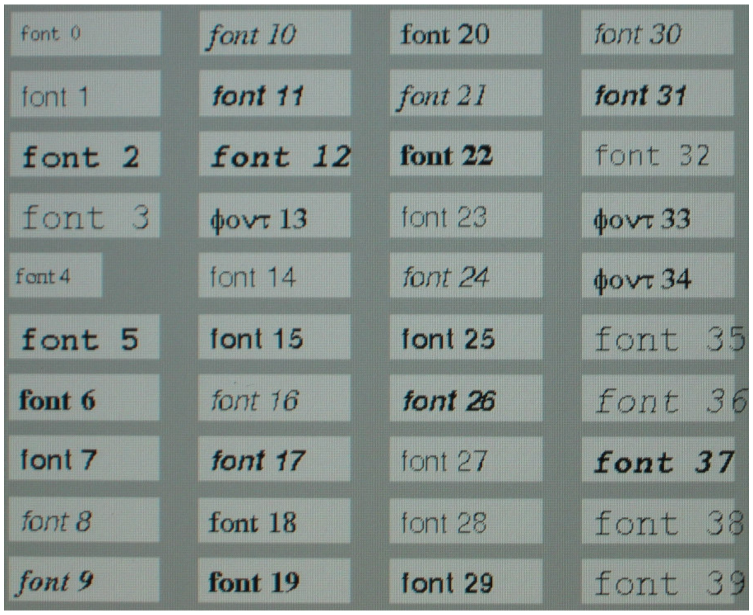
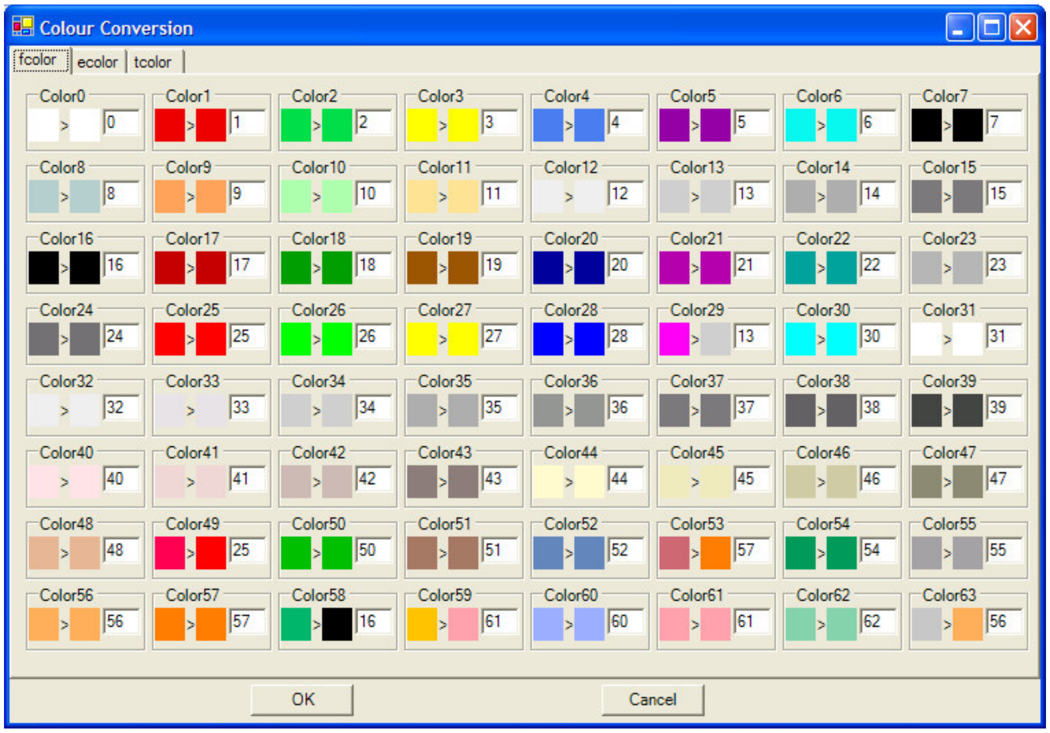


Figure 8: The first 40 `.g` file font's when converted to FoxView `fdf`.

As far as I know, you can't adjust the relationship between the top left corner of the text string and top left corner of the rectangle. So this ends up being the determining factor for setting "Y scale".

3.3.6 Colour Conversion Configuration

Figure 9 shows the configuration page for colour mapping. You will notice that there are 3 tab pages, one for fill colours (fcolor), one for edge colours (ecolor) and one for text colours (tcolor).



Color0 > 0	Color1 > 1	Color2 > 2	Color3 > 3	Color4 > 4	Color5 > 5	Color6 > 6	Color7 > 7
Color8 > 8	Color9 > 9	Color10 > 10	Color11 > 11	Color12 > 12	Color13 > 13	Color14 > 14	Color15 > 15
Color16 > 16	Color17 > 17	Color18 > 18	Color19 > 19	Color20 > 20	Color21 > 21	Color22 > 22	Color23 > 23
Color24 > 24	Color25 > 25	Color26 > 26	Color27 > 27	Color28 > 28	Color29 > 13	Color30 > 30	Color31 > 31
Color32 > 32	Color33 > 33	Color34 > 34	Color35 > 35	Color36 > 36	Color37 > 37	Color38 > 38	Color39 > 39
Color40 > 40	Color41 > 41	Color42 > 42	Color43 > 43	Color44 > 44	Color45 > 45	Color46 > 46	Color47 > 47
Color48 > 48	Color49 > 25	Color50 > 50	Color51 > 51	Color52 > 52	Color53 > 57	Color54 > 54	Color55 > 55
Color56 > 56	Color57 > 57	Color58 > 16	Color59 > 61	Color60 > 60	Color61 > 61	Color62 > 62	Color63 > 56

Figure 9: The Configuration page for colours.

The 32 Display Manager colours where mapped by my version of `pdf_fdf` as follows:

Table 2: The colour mapping pdf_fdf uses for Display Manager to .g/FoxView.

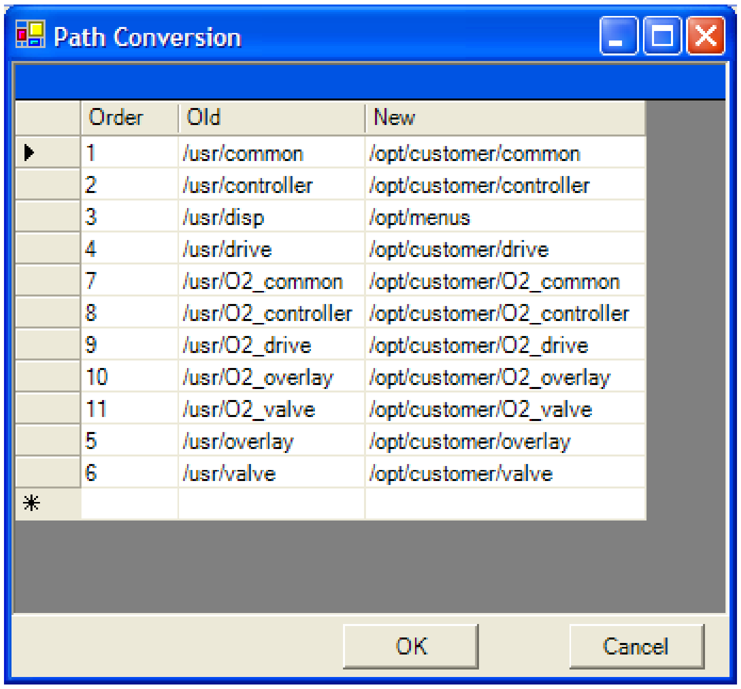
Display Manager Colour		***Blink***	***.g file/Fox View***	***Comment***
0	Black		16	
1	Dark Red		17	
2	Dark Green		18	
3	Brown		19	
4	Navy		20	
5	Bright Pink		21	
6	Aqua?		22	
7	Light Gray		23	
8	Dark Gray		24	
9	Red		25	
10	Light Green		26	
11	Yellow		27	
12	Blue		28	
13	Medium Gray		29	Incorrect - Should be FV13
14	Cyan		30	
15	White		31	
16	Beige		48	
17	0/10 (Light Green)	Yes	49	Incorrect. On Colour should be FV25
18	Darkish Green		50	
19	Pale brown		51	
20	Blue/Gray		52	
21	0/25 (Orange)	Yes	53	Incorrect. On colour should be FV57
22	Darker Green		54	
23	Gray (slightly darker than 13)		55	
24	Pale Orange		56	
25	Orange		57	
26	0/15 (White)	Yes	58	Incorrect. On colour should be FV16
27	0/29 (Pale Pink)	Yes	59	Incorrect. On colour should be FV61
28	Mid Blue		60	
29	Pale Pink		61	
30	Pale Green		62	
31	0/24 (Pale Orange)	Yes	63	Incorrect. On colour should be FV56

Perhaps the errors have been fixed with a later version.

3.3.6.1 Instructions

Usage is relatively trivial. To make selecting the appropriate colour (as it will be displayed in FoxView) a little easier, clicking on the "to" colour (right side of the > character) opens up the colour selector dialog.

3.3.7 Path Conversion Configuration



Order	Old	New
1	/usr/common	/opt/customer/common
2	/usr/controller	/opt/customer/controller
3	/usr/disp	/opt/menus
4	/usr/drive	/opt/customer/drive
7	/usr/O2_common	/opt/customer/O2_common
8	/usr/O2_controller	/opt/customer/O2_controller
9	/usr/O2_drive	/opt/customer/O2_drive
10	/usr/O2_overlay	/opt/customer/O2_overlay
11	/usr/O2_valve	/opt/customer/O2_valve
5	/usr/overlay	/opt/customer/overlay
6	/usr/valve	/opt/customer/valve

Figure 10: Path Conversion configuration dialog

Path conversion does three things:

1. Looks inside the `. userdata` fields in the `.g` file for strings of the form of the "Old" column and replaces them with the string of the "New" column.
2. Uses the same path mapping to "physically" relocate the output `.g` files.
3. Takes the opportunity to add an "`.fdf`" string to the end of the path since in every case the path is part of a full filename. Note that it handles pick variables – a recent bug fix. (For example `/usr/disp/$p5` will become `/usr/disp/$p5.fdf`).

The code searches for entries in the "Old" column in the order specified by the "Order" column. I figured this might prove useful to someone.

One downside is that it currently doesn't handle well any paths that have unusual characters. It will do the substitution of the "Old" with the "New" but then is likely to get into problems with steps 2 and 3. For example, we had about 8 trend files which where in a 3 subdirectories, each called `trends..`. When the application tries to create a directory called `/usr/disp/ore/trends..`, Windows complains. This wasn't a major issue for us as we fixed the problem by hand. If this is a major issue for anyone, let me know your specific requirements and I'll try to improve things.

If you want to know exactly what the code does here, see from line 654 (`#region Userdata`) of the file `FormMain.cs`.

3.4 Generating the FoxView fdf files

Use your favourite windows tar utility to copy the modified .g files back to your AW/WP.

Unix

To convert the `.g` files to `.fdf` you can choose to use `do_g_fdf_all` or just use the `-r` option with `g_fdf`. I ended up using the latter.

Windows

Sorry. Don't know what resources you have in version 8. Let us know so I can add to this document

3.5 Moving the FoxView files to their final location

Unix

You're all unix people. I used tar.

Windows

Even easier

4 Known Problems

This section contains the known issues we have at the time of writing this document. If I generate new versions, this document will be updated accordingly.

4.1 Colour blinking is slow

The `pdf_fdf` conversion utility leaves all blinking colours flashing in FoxView at about half the speed they did in Display Manager.

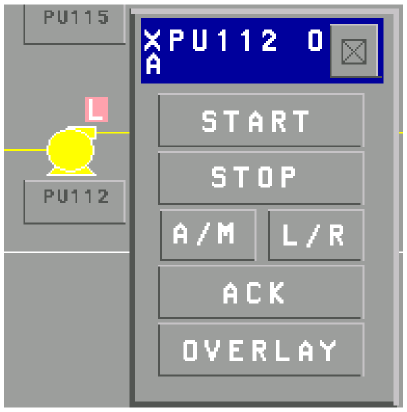
Olivier Prachar pointed out the particular parameter in the g file that controls blinking. Here's his comment:

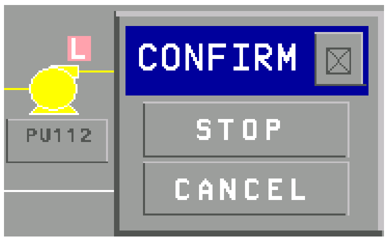
```
For blinking speed purpose, you can look in the g file and find
something like that: "0X400000E0,0X40000020,ALL,1" to be
"0X400000E0,0X40000020,ALL,2"
Where the last value is the speed frequency, with:
      0 = no blink,
      1 = blink slow,
      2 = blink fast.
OP.
```

Hence it's my intention to update the tool in a later version to allow people to choose fast or slow blinking.

4.2 Toggle commands don't work with $Pick and two layers of overlays

With our manually started and stopped pumps, we have an overlay for the pump, which has a start button and stop button as well as Auto/Manual, Local/Remote etc. When one clicks on the start or stop, a confirm dialog comes up as shown





What we have found is that, in FoxView, pressing the confirm START or STOP button does not do anything. Also the first overlay remains open. Still to chase this one up. Configuration looks OK. Seems to be a mismatch in behaviour between Display Manager and FoxView.

4.3 Tool doesn't handle non-simple characters in paths

See bottom paragraph of Section 3.3.7.

5 Appendix

5.1 Notes on generating "g" files

`pdf_fdf` obviously generates a FoxView `.fdf` file from a Display Manager pdf file. The interesting point is that it appears to do it by first converting to the generic `.g` text file format and then converting the `.g` file to an `.fdf` file. This seems to be the reason why there is a switch, `-g`, which allows one to "retain the g file".

To robustify things, I wrote the script to move all display manager files to a temporary directory (maintaining the directory structure), do the `pdf_fdf -g` on each file and then delete the `fdf` and `pdf` files that are also generated.

You can use the `-r` switch to recurse directories, but I found that it occasionally bombs out with a core dump if it comes across a bad file meaning that large numbers of files aren't processed. Given that it takes a while, this was a pain. You still then had to clean up the `.pdf` and `.fdf` files.