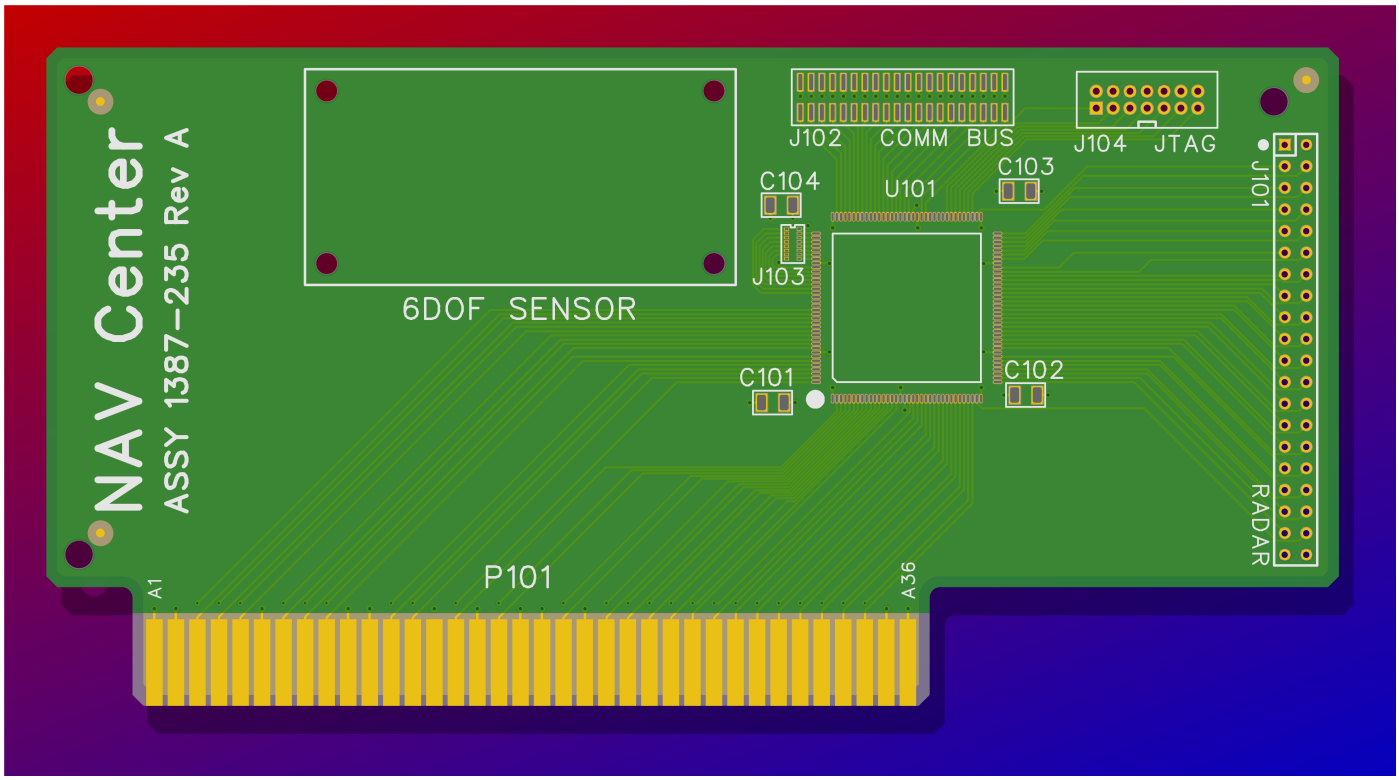
FpcPoly

A Gerber File Translator
for
FreePCB and ***PCB-Render***



User Guide

Version 1.13

This document is formatted for double sided printing.

Contents

FpcPoly End User License Agreement (EULA)

General

FpcPoly Software is distributed as Freeware. You may use the Software on any number of computers for as long as you like. The Software is NOT Public Domain software. You are allowed to freely distribute the Software, but Bruce Parham retains ownership and copyright of the Software in its entirety. You may use and/or distribute the Software only subject to the following conditions:

1. You may not modify the program or documentation files in any way.
2. You must include all the files that were in the original distribution.
3. You may not reverse engineer, decompile, or otherwise reduce the Software to a human perceivable form.
4. You may not sell the Software or charge a distribution fee, except to recover the media costs.
5. You understand and agree with this license and with the Disclaimer of Warranty and the Limitation of Liability printed below.
6. The Software may be bundled and distributed together with other software products provided the same conditions apply.

Agreement to this license

By using, copying, transmitting, distributing or installing the Software, you agree to all terms of this license. If you do not agree to all of the terms of this License, then do not use, copy, transmit, distribute or install the Software and immediately remove the Software from your storage device(s).

Warranty disclaimers and liability limitations

The Software and related documentation are provided "as is", without warranty of any kind. None whatsoever. Bruce Parham disclaims all warranties, express or implied, including, but not limited to, warranties of design, merchantability, or fitness for a particular purpose or any warranty of title or non-infringement. Bruce Parham does not warrant that the functions contained in the Software or documentation will meet your requirements, or that the operation of the Software will be error-free, complete, or that defects in the Software or documentation will be corrected.

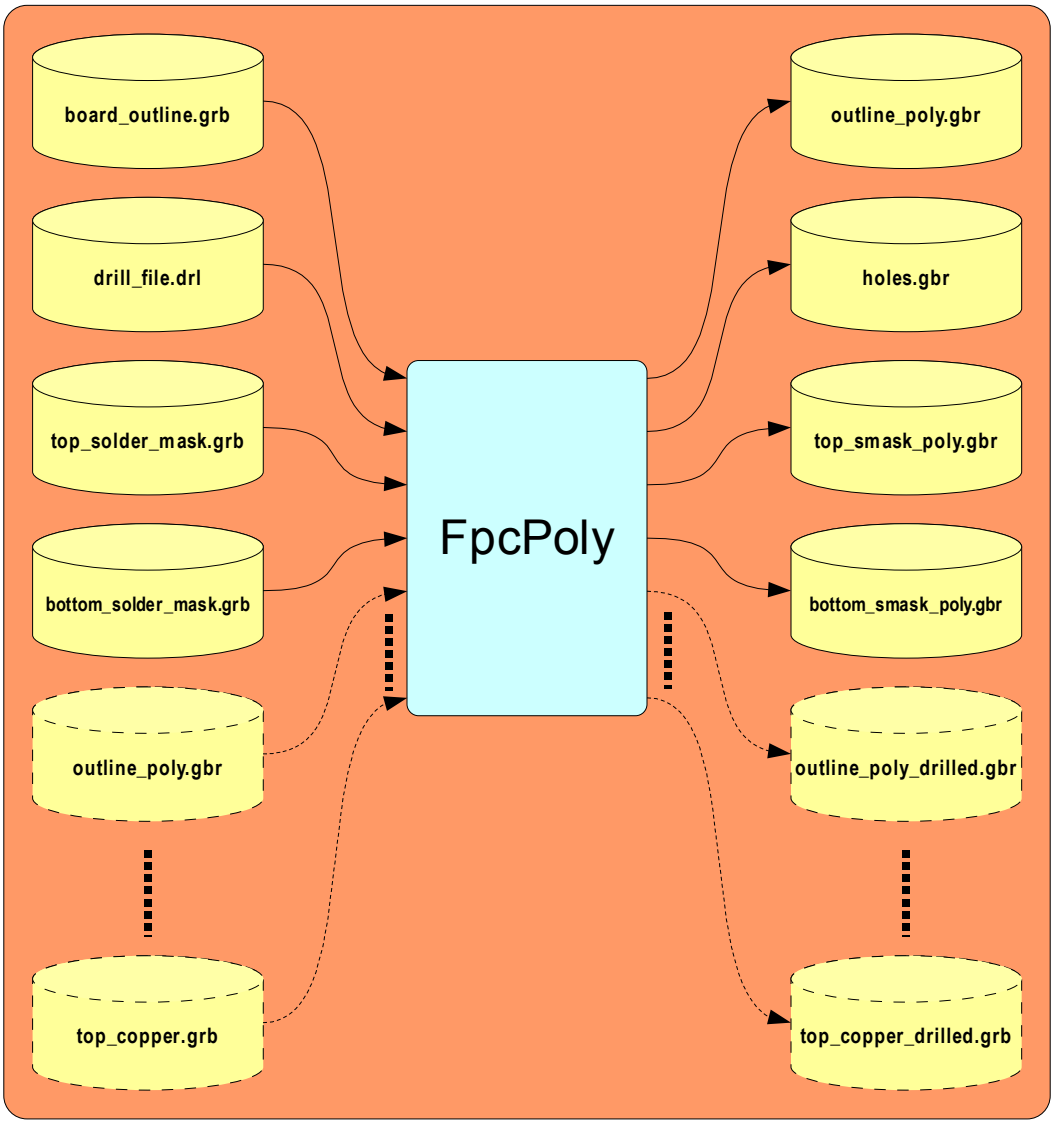
Limitation of Liability

Under no circumstances shall Bruce Parham be liable for any lost revenue or profits or any incidental, indirect, special, or consequential damages that result from the use or inability to use the Software or related products or documentation, no matter what legal theory it is based on.

Copyright and trademarks

Introduction

FpcPoly is a file translator that converts some of the standard Gerber files generated by ***FreePCB*** into a form usable by Guillaume Rosanis' ***PCB-Render*** in producing very photo-realistic PCB images.



The standard conversion process uses the board outline data found in the **board_outline.grb** file to create new files with the outline data converted to one or more solid polygons. The first file created, **outline_poly.gbr**, only contains the outline defined polygons. If solder mask data is available, additional files are created with solid, outline defined, polygons that use the data contained in the top and/or bottom solder mask files to create cutouts in the polygons. This process produces an outline limited image of the solder mask with the mask features in the correct polarity, i.e., as openings. Finally, the Excellon drill data is converted to a Gerber file with the drill data translated into pads.

Additional files may be specified, with optional offsets that use the drill data to form cutouts producing, in effect, a drilled version of the file.

Plane text command scripts may be used to simplify repeated execution with a large number of sommand switches.

This process produces a set of files that can be layered together by ***PCB-Render*** to create a photo-realistic picture of a regular or irregularly shaped board complete with one or more drop shadows and the background visible through the drilled holes and around the edges.

Installation and Setup

FpcPoly is distributed as a set of files in a zip archive that includes the executable, the user guide and a selection of example files.

To install ***FpcPoly***, extract the file **FpcPoly.exe** from the distribution archive to any handy directory such as C:\Program Files\ FpcPoly\ or C:\Program Files\FreePCB\bin\.

If the path to FpcPoly.exe is not already included in the environment **PATH** variable, it should be added.

For those unfamiliar with setting environment variables, assume for example that FpcPoly.exe is installed in the C:\Program Files\FreePCB\bin directory:

- Open the **System Properties** form by
  - Right clicking on **My Computer** and clicking on **Properties**.

  or
  - Open the **Control Panel**.
  - Open the **System** tool.
- Click the **Advanced** tab and open the **Environment Variables**.
- Scroll the variable list down to and highlight the **path=** entry and click **EDIT**.
- Click in the value field to clear the highlight.
- Scroll to the end of the field and add **;C:\Program Files\FreePCB\bin** to the existing value.
- Click **OK** to exit the edit window.
- Click **OK** to save and exit the Environment Variables window.
- Click **OK** to exit System Properties.

Operation

FpcPoly is a command line application; to use it, open a NT command shell (cmd.exe) and change the working directory to the directory where the Gerber files are located. At the command prompt, type *fpcpoly* and hit return to run the program in its default mode. In this mode, the following translations are performed:

1. The board outline file **board_outline.grb** is used to create **outline_poly.gbr** with all outline figures converted to solid polygons.
2. The Excellon drill data in **drill_file.drl** is used to create a Gerber format drill file in **holes.gbr**.
3. The file **top_smask_poly.gbr** is created with the outline data used to generate solid polygons and the solder mask data from **top_solder_mask.grb** added in as cutouts. This produces an outline limited version of the top solder mask with the mask features shown as clear openings.
4. The file **bottom_smask_poly.gbr** is created with the outline data again used to generate solid polygons and the solder mask data from **bottom_solder_mask.grb** added in as cutouts. This produces an outline limited version of the bottom solder mask with the mask features shown as clear openings.

```
D:\Demo1\CAM> fpcpoly

    FpcPoly - a RS-274X File Converter for Rendering
            Vers 1.00R, (c)2008 Bruce Parham

     outline_poly.gbr created
     holes.gbr created
     top_smask_poly.gbr created
     bottom_smask_poly.gbr created

  Done, 4 files created in 0.068 Sec.
```

At each step above, if the source file or files can not be located, a warning message is displayed and work proceeds to the next step:

```
D:\Demo1\CAM> fpcpoly

    FpcPoly - a RS-274X File Converter for Rendering
            Vers 1.00R, (c)2008 Bruce Parham

     outline_poly.gbr created
     holes.gbr created.
Warning: mask file "top_solder_mask.grb" not found, "top_smask_poly.gbr" not created.
     bottom_smask_poly.gbr created

  Done, 3 files created in 0.076 Sec.
```

FpcPoly execution is a two step process. In the first step all option switches, from the command line or script files, are processed to modify default setting and build option lists. Once that is completed, ***FpcPoly*** moves on to the second step where the actual file are created.

Syntax and Option Switches

FpcPoly can use a number of option switches to change file names or specify data offsets. If a command line switch parameter contains any embedded spaces, the entire switch must be enclosed within double quotes: "N2name with spaces.ext" but this quotation is not needed within a command script.

The command line syntax was designed for simple yet flexible operation. The full syntax consists of the program name followed by optional switches. The switches are not case sensitive and, for the most part, can appear in any order (order is significant only when specifying offsets and drilled files):

```
C:> fpcpoly [switch [switch [switch ...]]]
```

Defined Switches:

S*n*Fname (Source file name)

Change the default source ***n*** file name to *Fname*. The supplied file name must be complete; no default file extension will be added but the name may include any valid relative or absolute path. Duplicate entries are allowed with the last entry overriding any earlier entries. The digit **n** selects which source name is changed as follows:

S1*Fname* changes the default outline name from ***board_outline.grb*** to ***Fname***.

S2*Fname* changes the default drill name from ***drill_file.drl*** to ***Fname***.

S3*Fname* changes the default top mask name from ***top_solder_mask.grb*** to ***Fname***.

S4*Fname* changes the default bottom mask name from ***bottom_solder_mask.grb*** to ***Fname***.

SpSpath (Source file path)

Change the source path to ***Spath***. This string, empty by default, is perpended to the *outline*, *drill* and *solder mask* file names. If Spath does not end with a backslash (\), one will be added.

FpcPoly first attempts to open "drilled" files without ***Spath*** being added. If that fails, another attempt is made with ***Spath.*** Thus, a file just created in the current directory can be processed as well as those in the source directory specified by ***Spath***.

N*n*Fname (New file name)

Change the default new (destination) **n** file name to Fname. Again, the file name must be complete; no default file extension will be added but the name may include any valid relative or absolute path. Duplicate entries are allowed with the last entry overriding any earlier entries. A null, empty, name will cause processing of the selected file to be skipped. The digit **n** selects which destination name is changed as follows:

N1*Fname* changes the default new outline name from ***outline_poly.gbr*** to ***Fname***.

N2*Fname* changes the default new drill name from ***holes.gbr*** to ***Fname***.

N3*Fname* changes the default new top mask name from ***top_sm_poly.gbr*** to ***Fname***.

N4*Fname* changes the default new bottom mask name from ***bottom_sm_poly.gbr*** to ***Fname***.

Cfname (Command script)

Run poly command script *fname*. If *fname* does not include an extension, as defined by the presence of a period "**.**" character, ***.PCS*** will be appended. A command script is a plane text file that may contain command switches, comments and blank lines. Each command should be placed on separate line. Comments begin with a semicolon ";" character and continue to the end of the current line. Blank lines, comments and leading white space (spaces and/or tabs) are ignored. Scripts may be nested up to 8 levels deep. Each script inherits the parents current offset value but the parent offset is restored when the child script terminates.

Example script files:

```
; File: test1.pcs  -- Command Script Example --

; Note that parameters containing spaces do not need to
; be quoted in a script file.

sp..\                       ; Gerber source path

x.009                       ; exclude some drill sizes
x.014

n2                          ; skip drill and bottom mask
n4

  dtop_copper.grb           ; some drilled files w/o offset
  doutline_poly.gbr

o0.07 -0.12                 ; set the offset for this level
  doutline_poly.gbr         ; new offset version

o-0.09 -0.095               ; change offset
  doutline_poly.gbr         ; another new one

r.05 .03                    ; offset relative to previous
  doutline_poly.gbr         ;

ctest2                      ; nesting level test

end                         ; restore parent offset

Anything after the end command is ignored...
```

```
; File: test2.pcs    -- Nesting Level Demo --

r.005 -.006           ; relative to parent
  doutline_poly.gbr

ctest2                ; recursive call fails after 8 levels

end                   ; restore parent offset
```

(Found at Dictionary.com: **re·cur·sion** -*noun* ... see also recursion, ...)

Running the test1.pcs produces the following:

```
D:\Code\FpcPoly\Demo1\CAM2\Assembly> fpcpoly ctest1

    FpcPoly - a RS-274X File Converter for Rendering
            Vers 1.13R, (c)2008 Bruce Parham

  -- Running "test1.pcs" at nested level 1 --
  Source path changed to "..\"
  1  0.0090 dia drills suppressed
  2  0.0140 dia drills suppressed
  -- Drill file creation skipped --
  -- Bottom mask file creation skipped --
  "top_copper.grb" gets drilled (1)
  "outline_poly.gbr" gets drilled (1)
  Data Offset (0.070, -0.120)
  "outline_poly.gbr" gets drilled (2) with offset (0.070,-0.120)
  Data Offset (-0.090, -0.095)
  "outline_poly.gbr" gets drilled (3) with offset (-0.090,-0.095)
  Offset moved by (0.050, 0.030) to (-0.040, -0.065),
  "outline_poly.gbr" gets drilled (4) with offset (-0.040,-0.065)
  -- Running "test2.pcs" at nested level 2 --
  Offset moved by (0.005, -0.006) to (-0.035, -0.071),
  "outline_poly.gbr" gets drilled (5) with offset (-0.035,-0.071)
  -- Running "test2.pcs" at nested level 3 --
  Offset moved by (0.005, -0.006) to (-0.030, -0.077),
  "outline_poly.gbr" gets drilled (6) with offset (-0.030,-0.077)
  -- Running "test2.pcs" at nested level 4 --
  Offset moved by (0.005, -0.006) to (-0.025, -0.083),
  "outline_poly.gbr" gets drilled (7) with offset (-0.025,-0.083)
  -- Running "test2.pcs" at nested level 5 --
  Offset moved by (0.005, -0.006) to (-0.020, -0.089),
  "outline_poly.gbr" gets drilled (8) with offset (-0.020,-0.089)
  -- Running "test2.pcs" at nested level 6 --
  Offset moved by (0.005, -0.006) to (-0.015, -0.095),
  "outline_poly.gbr" gets drilled (9) with offset (-0.015,-0.095)
  -- Running "test2.pcs" at nested level 7 --
  Offset moved by (0.005, -0.006) to (-0.010, -0.101),
  "outline_poly.gbr" gets drilled (10) with offset (-0.010,-0.101)
  -- Running "test2.pcs" at nested level 8 --
  Offset moved by (0.005, -0.006) to (-0.005, -0.107),
  "outline_poly.gbr" gets drilled (11) with offset (-0.005,-0.107)

-- Maximum nesting level reached, "ctest2" ignored --


  -- "test2.pcs" done --
  -- "test2.pcs" done --
  -- "test2.pcs" done --
  -- "test2.pcs" done --
  -- "test2.pcs" done --
  -- "test2.pcs" done --
  -- "test2.pcs" done --
  -- "test1.pcs" done --

     outline_poly.gbr created
     top_smask_poly.gbr created
     top_copper_drilled.grb created
     outline_poly_drilled.gbr created
     outline_poly_drilledOf1.gbr created
     outline_poly_drilledOf2.gbr created
     outline_poly_drilledOf3.gbr created
     outline_poly_drilledOf4.gbr created
     outline_poly_drilledOf5.gbr created
     outline_poly_drilledOf6.gbr created
     outline_poly_drilledOf7.gbr created
     outline_poly_drilledOf8.gbr created
     outline_poly_drilledOf9.gbr created
     outline_poly_drilledOf10.gbr created

  Done, 14 files created in 0.156 Sec.
```

Dfname.ext (Drilled files)

Create a new, "drilled" version of file ***fname.ext*** by copying the contents of ***fname.ext*** to the new file and then adding the drill data as cutouts. Both steps add the current offset to the data as it is transferred.

If the current X and Y offsets are both zero, the newly created file name is formed by appending ***_drilled*** to the source name, i.e., ***fname.ext*** produces ***fname_drilled.ext***. If either offset is not zero, the appended suffix is changed to ***_drilledOf***n, where **n** is a number, starting with 1, to identify uniquely offset versions of source file, i.e., ***fname_drilledOf1.ext, fname_drilledOf2.ext*** and so on up to the limit of ***fname_drilledOf32.ext***.

Note that this option can be applied to files just created, like the board polygon. It can be included up to 32 times to specify a number of files and that each file may have up to 32 unique offsets.

FpcPoly first attempts to open ***fname.ext*** in the current directory, if the open attempt fails, a second attempt is made using ***Spath\fname.ext***.

E (End script)

Command scripts are normally processed until the end-of-file is reached but may be terminated with the ***End*** command. Any text, in a script file, following this command is ignored. The End command is only effective within a script; if found as a command line parameter, it is silently ignored.

Ox.xx**,**y.yy (Offset)

Initially, drilled files are generated with no offset but this option can be used to set the X,Y offset for subsequent drilled files. This option can be entered any number of times and the resulting offset values will remain in effect until changed by another offset entry. Note that the same file can be output multiple time with different offsets (see **D** option, above). Additionally, the X and Y values may be separated by a comma or a space if the whole switch in enclosed within quotes: "**o**0.123 -0.25" or even "**O** 0.5 .2". The quotes are not needed in script files.

Rx.xx,y.yy (Relative offset)

Move the current offset by adding ***x.xx***, ***y.yy*** to it. Quotation requirements are the same as the **O** option, above.

Xd.dddd (Exclude drills)

Exclude all drills of diameter ***d.dddd*** from drilled files. The diameter entered is rounded to four decimal places as are the Excellon tool sizes when tested for exclusion. This option can be entered any number of times to exclude multiple drill sizes but multiple entries of the same size will be ignored. (Suggestion: use this option to hide holes that will be filled and capped and thus not be visible on the actual board.)

H or **?** (Help)

This switch displays a short summary of the syntax and valid switches.

Command Line Examples

1. Add holes to "drill" the top copper file:

```
D:\Demo1\CAM> fpcpoly  dtop_copper.grb

    FpcPoly - a RS-274X File Converter for Rendering
            Vers 1.00R, (c)2008 Bruce Parham

  "top_copper.grb" gets drilled

     outline_poly.gbr created
     holes.gbr created
     top_smask_poly.gbr created
     bottom_smask_poly.gbr created
     top_copper_drilled.grb created

  Done, 5 files created in 0.087 Sec.
```

2. Suppress drill file generation, change the source path to the parent directory and drill three file including the newly created outline:

```
D:\Demo1\CAM> fpcpoly  n2  sp.. doutline_poly.gbr  dtop_copper.grb  dbottom_copper.grb

    FpcPoly - a RS-274X File Converter for Rendering
            Vers 1.00R, (c)2008 Bruce Parham

  -- Drill file creation skipped --
  Source path changed to "..\"
  "outline_poly.gbr" gets drilled
  "top_copper.grb" gets drilled
  "bottom_copper.grb" gets drilled

     outline_poly.gbr created
     top_smask_poly.gbr created
     bottom_smask_poly.gbr created
     outline_poly_drilled.gbr created
     top_copper_drilled.grb created
     bottom_copper_drilled.grb created

  Done, 6 files created in 0.114 Sec.
```

3. Change the new drill name to something with embedded spaces by enclosing the whole switch in quotes:

```
D:\Demo1\CAM> fpcpoly  "n2drilled holes.gbr"

    FpcPoly - a RS-274X File Converter for Rendering
            Vers 1.00R, (c)2008 Bruce Parham

  New drills changed to "drilled holes.gbr"

     outline_poly.gbr created
     drilled holes.gbr created
     top_smask_poly.gbr created
     bottom_smask_poly.gbr created

  Done, 4 files created in 0.0103 Sec.
```

4. The drill source is changed to something invalid. The missing file causes warnings to be issued:

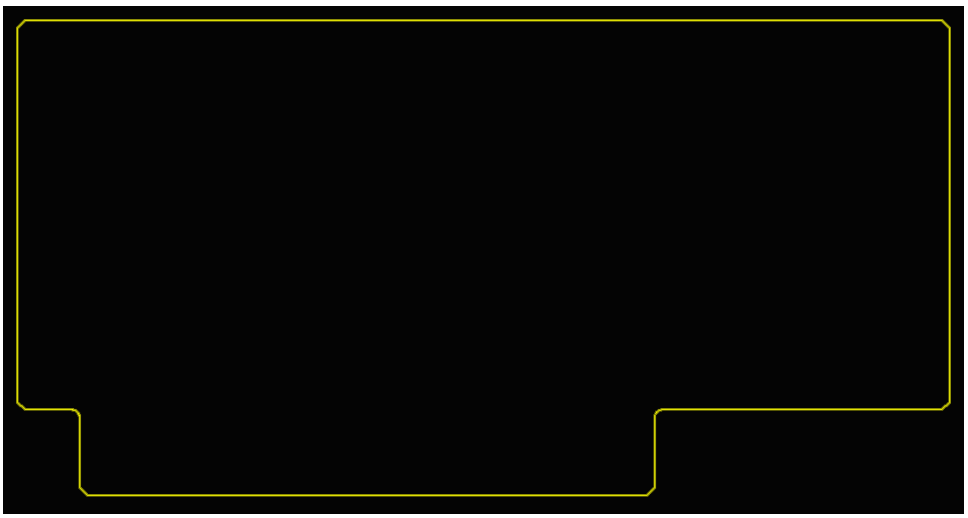
```
D:\Demo1\CAM> fpcpoly  s2junk  doutline_poly.gbr

    FpcPoly - a RS-274X File Converter for Rendering
            Vers 1.00R, (c)2008 Bruce Parham

  Drill source changed to "junk"
  "outline_poly.gbr" gets drilled

     outline_poly.gbr created
Warning: drill file "junk" not found, "holes.gbr" not created.

     top_smask_poly.gbr created
     bottom_smask_poly.gbr created

Warning: drill file "junk" not found, no drilled files created.

  Done, 3 files created in 0.132 Sec.
```

5. Generate drilled files with a two copies of the outline generated with offsets to use as a drop shadows.

```
D:\Demo1\CAM>fpcpoly doutline_poly.gbr o0.07,-0.13 doutline_poly.gbr o-0.07,-0.16 doutline_poly.gbr

    FpcPoly - a RS-274X File Converter for Rendering
            Vers 1.00R, (c)2008 Bruce Parham

  "top_copper.grb" gets drilled
  Data Offset (0.070, -0.130)
  "outline_poly.gbr" gets drilled with offset (0.070,-0.130)
  Data Offset (-0.070, -0.160)
  "outline_poly.gbr" gets drilled with offset (-0.070,-0.160)


     outline_poly.gbr created
     holes.gbr created
     top_smask_poly.gbr created
     bottom_smask_poly.gbr created
     outline_poly_drilled.gbr created
     outline_poly_drilledOf1.gbr created
     outline_poly_drilledOf2.gbr created

  Done, 7 files created in 0.124 Sec.
```

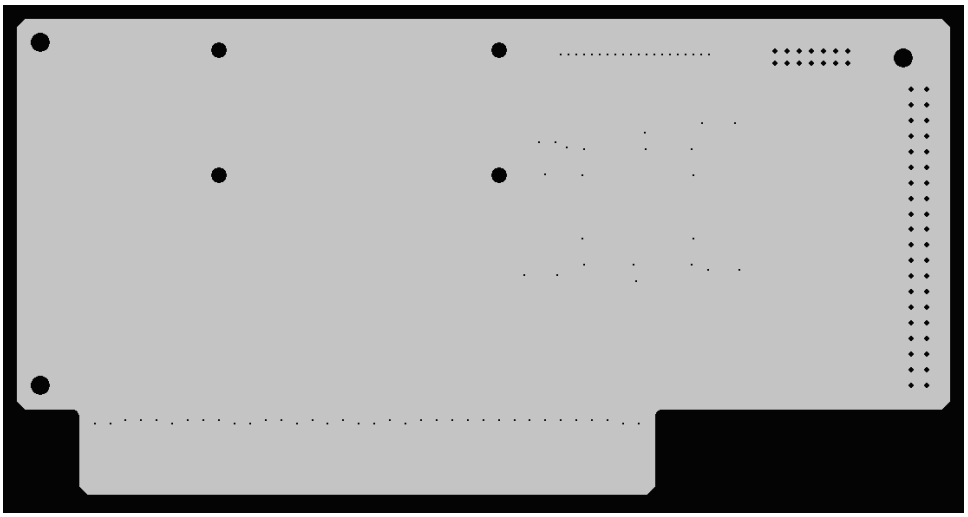
Gerber Graphics

This section examines some of the individual Gerber files produced by ***FreePCB*** and modified by ***FpcPoly*** as seen in the free Gerber viewer ***GC-Prevue***[1].



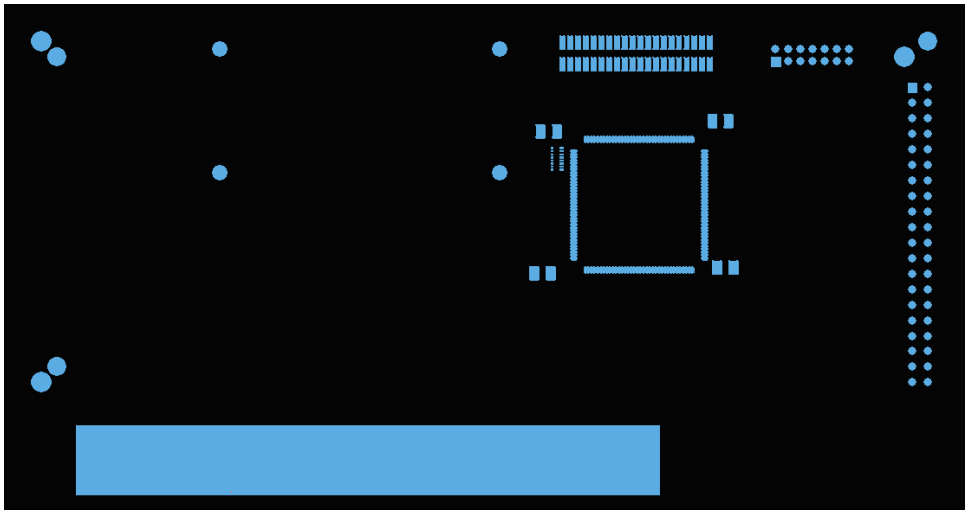
The board outline produced by ***FreePCB***.

The same outline converted to a solid polygon by ***FpcPoly***.

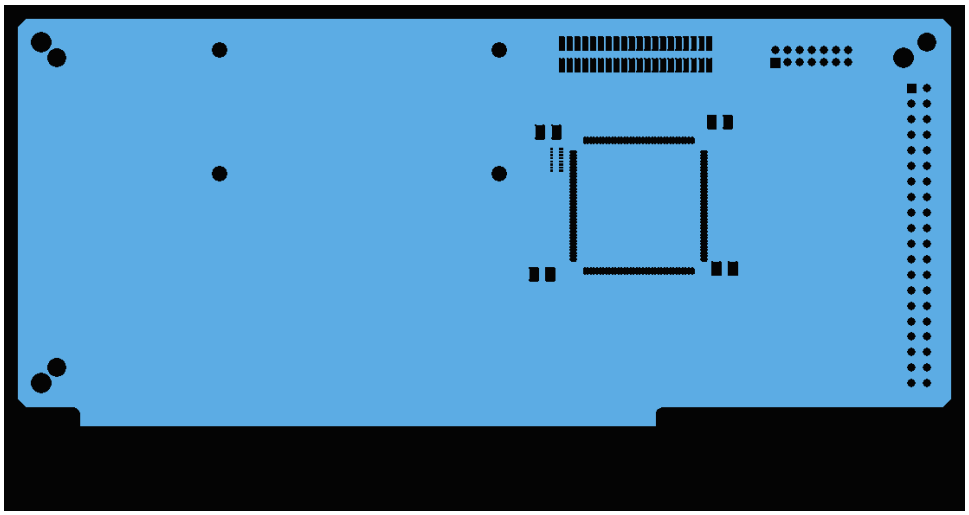


The outline polygon further modified by ***FpcPoly*** to add the drill data as cutouts.

1 See the **Handy Links** section.

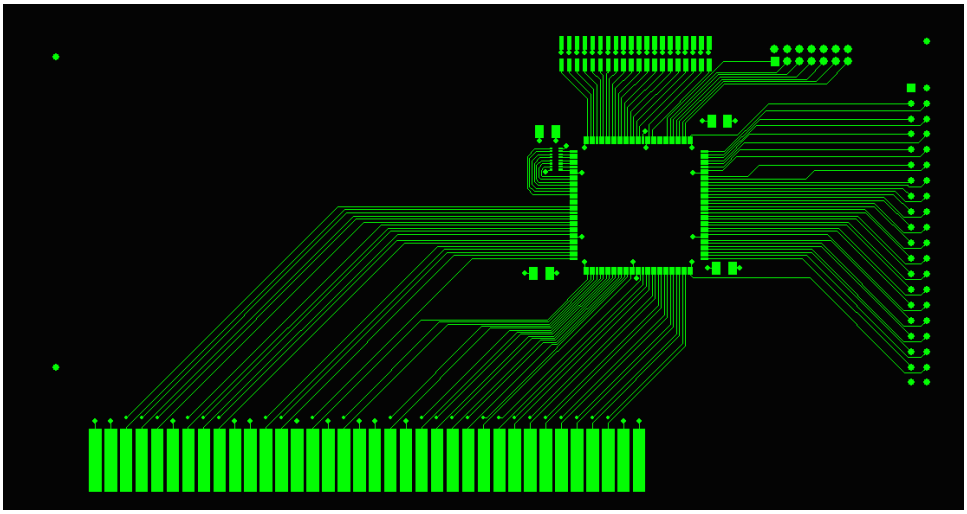


The standard top solder mask produced by ***FreePCB***.

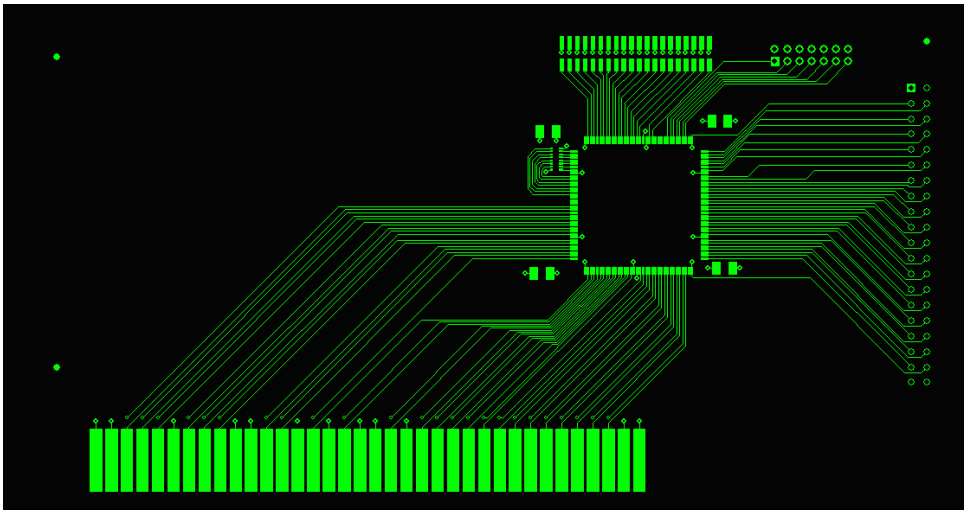


The outline defined polygon created by ***FpcPoly*** with the solder mask data used to create cutouts.

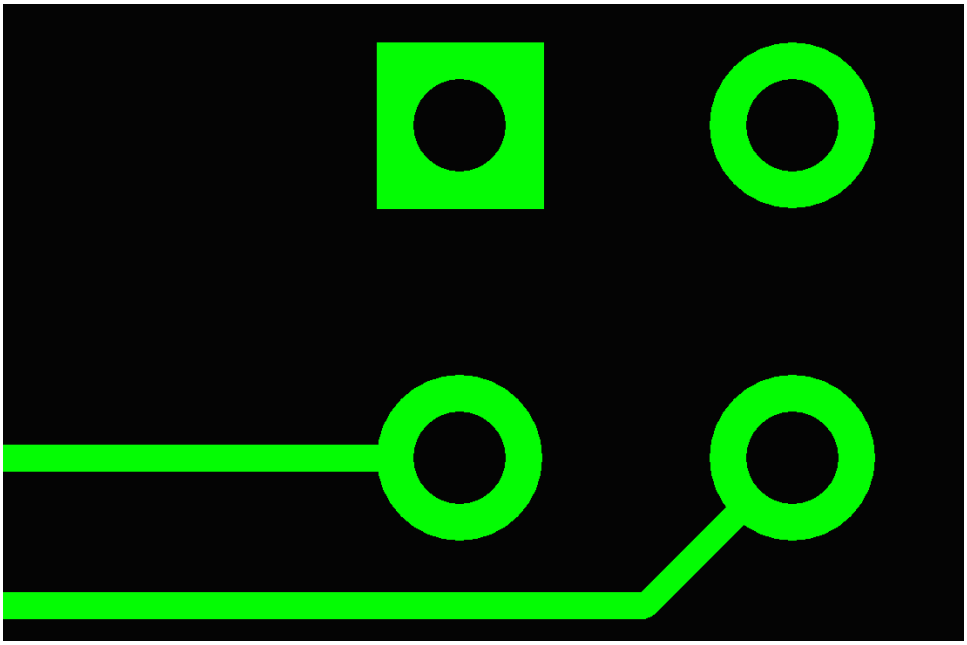
A detailed view of part of the the connector located near the top edge of the board above.



The standard top copper layer created by ***FreePCB***.



The top copper layer with drill data added as cutouts by ***FpcPoly***.



A detailed view of part of the header located near the right edge of the board above.

Graphic Rendering Tutorial

Successful photo-realistic image rendering depends on a number of factors. The primary factors, of course, are layer color and transparency but the choice of background as well as the use of holes and a drop shadow help create the illusion of reality.

The following tutorial will focus on the use of the freeware tool ***Paint.NET***[1] for image editing and manipulation. Many tools like ***Photoshop***, ***Paint Shop Pro*** and others are available but ***Paint.NET*** has proven to be both simple to use and powerful *and* the price ($0.00) is right.

Create Gerber Files

The first step in rendering is to use ***FpcPoly*** to create a set of modified Gerber files. This file set should include drilled versions of the board outline polygon and upper two copper layers as well as the solder masks. Additionally, the set should include an offset version of the drilled outline that will be used as a drop shadow. If the board is a two sided design, drilled versions of the top and bottom copper should be generated. For the top view of a multi-layer design, generate drilled versions of the top and inner1 layers.

Ex.:

```
C:\>fpcpoly N2 Dtop_copper.grb Dinner_copper_2.grb Doutline_poly.gbr o0.07,-.13 Doutline_poly.gbr
```

Determine Final Image Size

Next, the overall image size, in pixels, needs to be determined. One can measure the board, add in the drop shadow offsets and borders and multiply by the DPI to get the size or just run ***PCB-Render*** on a couple files using the real settings. The latter method is actually simpler and less error prone. With a text editor, create a short job file for rendering:

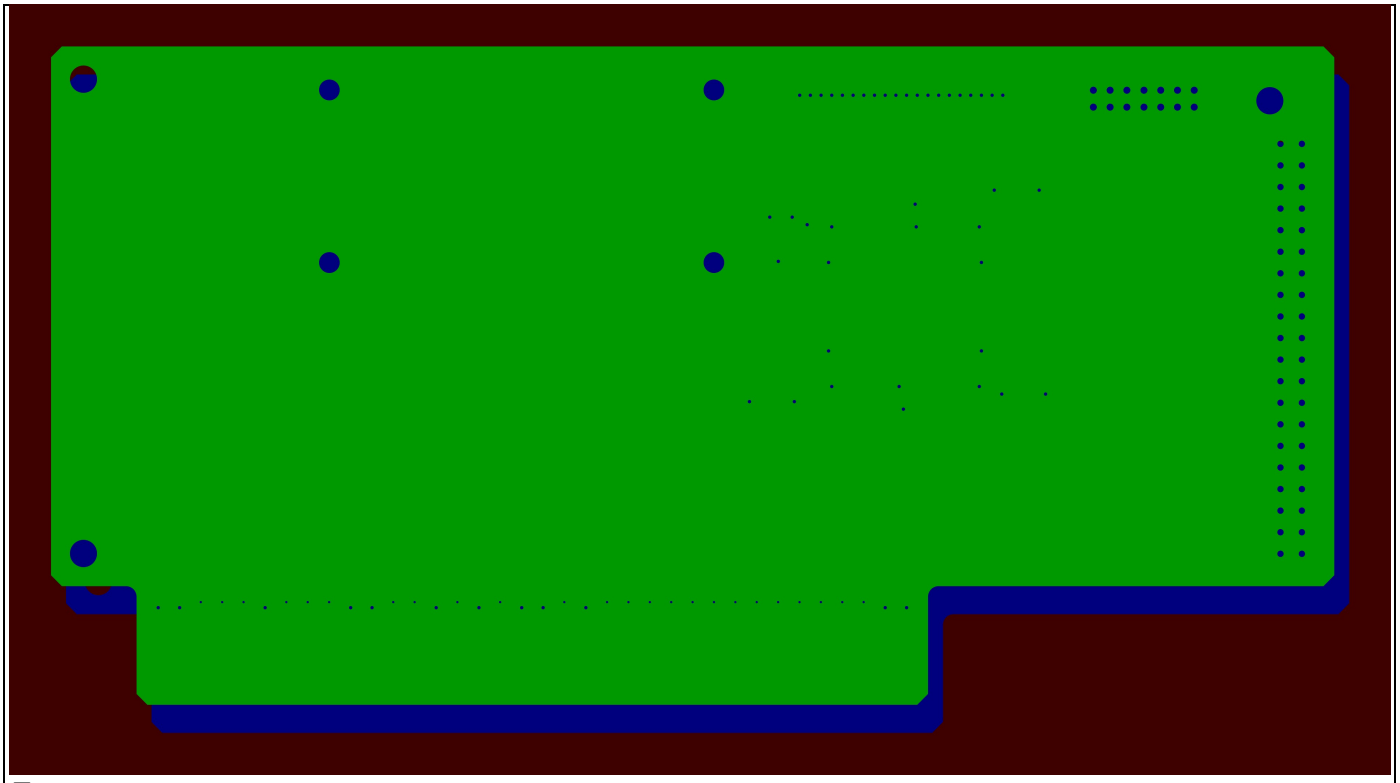
test.pcbjob:

```
GLOBAL
{
    DPI = 600;                              # some standard settings
    MARGIN = 5.0;
    AA = .001;
}

BACKGROUND
{
    COLOR = 0.25, 0.0, 0.0;                 # a dark red background
}

LAYER
{
    GERBER = outline_poly_drilledof1.gbr;   # medium blue for the shadow
    COLOR = 0.0, 0.0, 0.5;
}

LAYER
{
    GERBER = outline_poly_drilled.grb;      # medium green for the board
    COLOR = 0.0, 0.6, 0.0;
}
```

and run it with PCB-Render to generate **test.png**:

```
D:\Demo1\CAM> pcb-render test.job -o test.png
PCB dimensions: 154.18 x 80.77 mm
Resolution: 600.00 DPI, AA: 0.0428 mm, Margin: 5.00 mm
Rendering: 100 %   [4 of 4]
Writing PNG file...
```

1 See the **Handy Links** section.



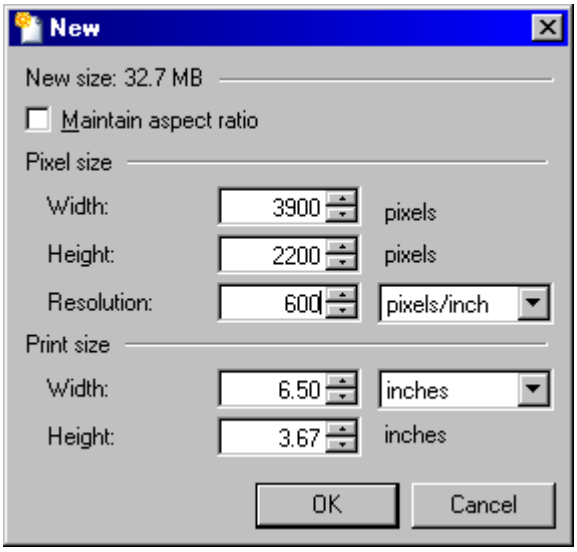
Test.png

Importing test.png into Paint.NET reveals that the final image is 3878 pixels wide by 2144 high.

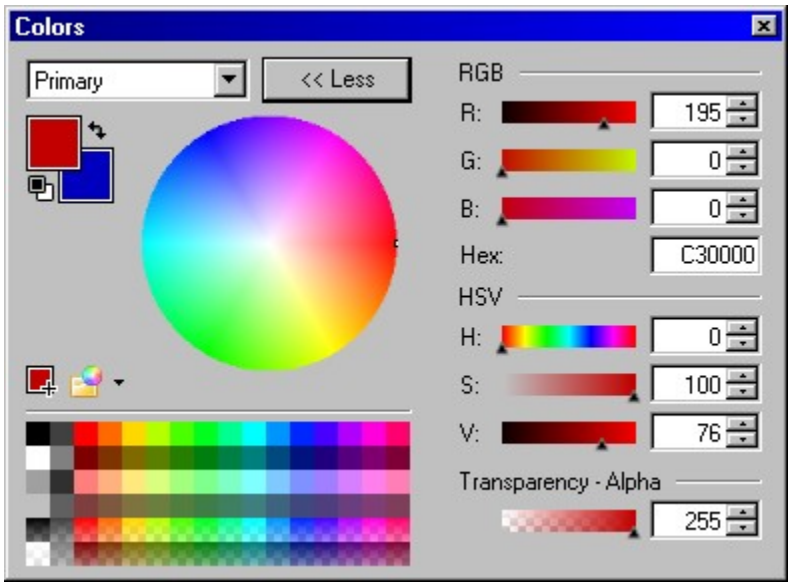
Create a Background

As an alternative to a fixed color, PCB-Render can use an image to form the background. If the selected background image is smaller then the rendered image, multiple copies are used to tile the background. To avoid background tiling, the image used should be as large as or larger then the final image.

In the next step, we'll use Paint.NET to create a background gradient image that is slightly larger then the rendered board image.



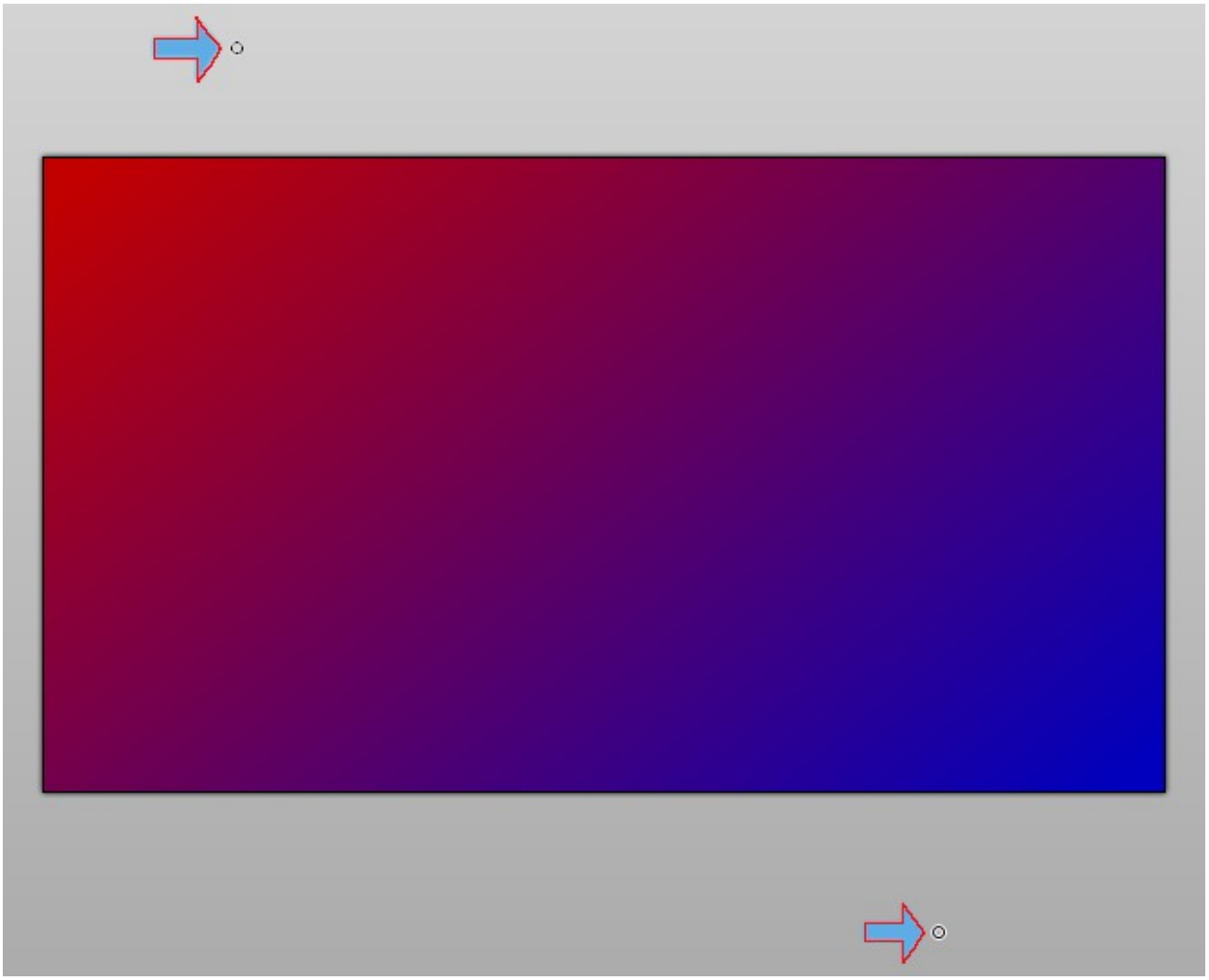
In Paint.NET, click **File** → **New** to create a new canvas and set is size to 3900 pixel wide by 2200 pixels high.



In the Colors window, click on More>> to gain access to the direct RGB settings. Set the primary color to value to Red 195 and the secondary color to Blue 190.

On the Tools bar, select the Gradient tool. 

Click above the canvas and drag the cursor down below it to generate a gradient between the two points. The two small *nubs* can be dragged around, even outside the canvas, to adjust the gradient. As shown below, the nubs were positioned so that the gradient covers the full canvas at an angle without saturating:



When the gradient is adjusted to your liking, save it as a .PNG file with a meaningful name like **Background.png**. It should be placed in the same directory where the Gerbers are located.

Job File

Next, we need to create the real job file to be used with PCB-Render. To maintain the image size, the global settings should match what was used in the earlier test file.

board_top.pcbjob:

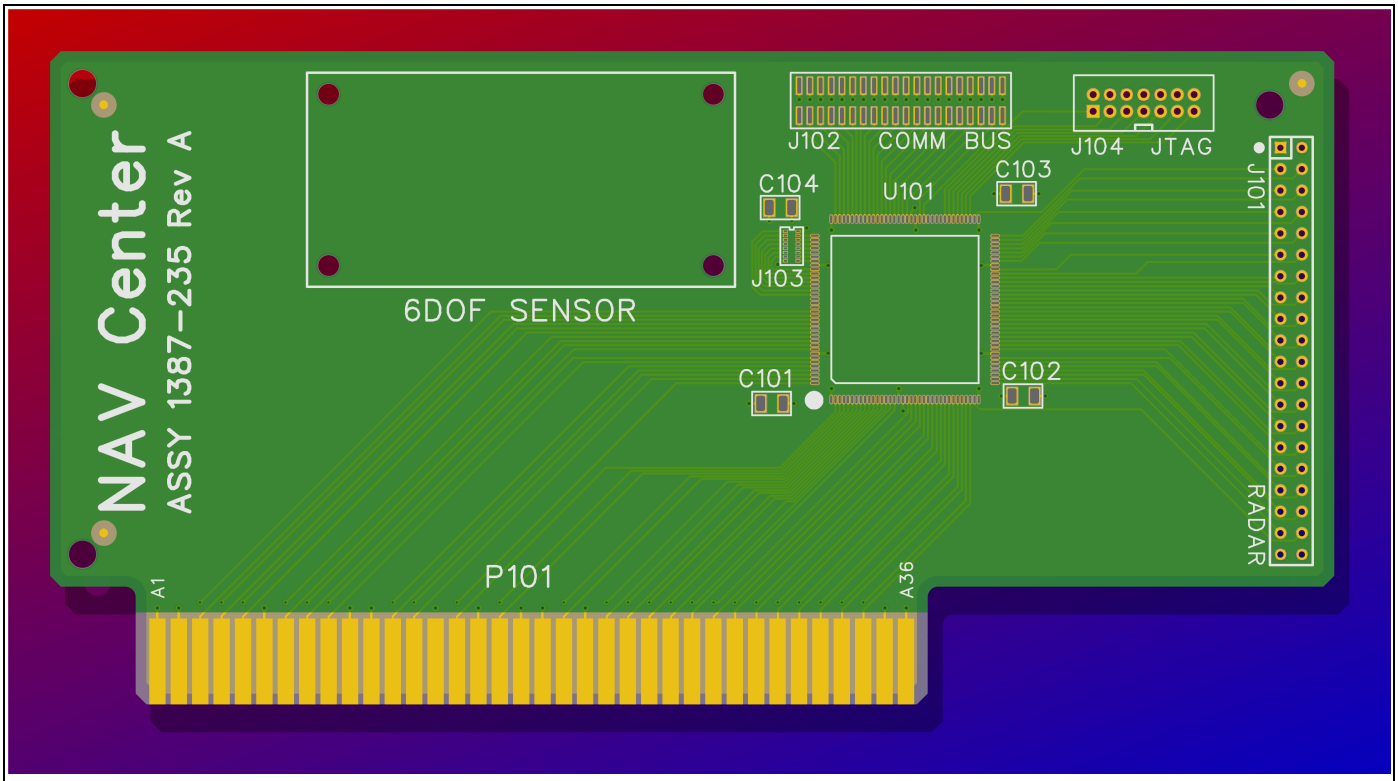
```
GLOBAL
{
    DPI = 600;
    MARGIN = 5.0;
    AA = .001;                              # a value of zero is ignored...
}

BACKGROUND
{
    IMG = Background.png;                   # background first
}

LAYER                                       # stack up the image layers from bottom to top
{
    GERBER = outline_poly_drilledof1.gbr;   # drop shadow next
    COLOR = 0.0, 0.0, 0.0;                  # black
    MIRROR = FALSE;                         # all FALSE for top view, TRUE for bottom
    ALPHA = 0.3;                            # 70% transparent
}

LAYER
{
    GERBER = inner_copper_1_drilled.grb;    # lower copper layer next
    COLOR = 0.84, 0.59, 0.04;               # copper color
    MIRROR = FALSE;
}

LAYER
{
    GERBER = outline_poly_drilled.gbr;      # top dielectric
    COLOR = 0.6, 0.6, 0.6;                  # for Rogers 4000, for FR-4 use 0.4, 0.6, 0.4
    MIRROR = FALSE;
    ALPHA = .75;                            # 25% transparent to show lower copper
}

LAYER
{
    GERBER = top_copper_drilled.grb;        # upper copper
    COLOR = 0.92, 0.75, 0.09;               # for ENIG Gold, for HASL use 0.6, 0.6, 0.6
    MIRROR = FALSE;
}

LAYER
{
    GERBER = top_smask_poly.gbr;            # top solder mask
    COLOR = 0.05, 0.5, 0.1;                 # light green LPI, for Vacril dry film use 0.4, 0.7, 0.4
    MIRROR = FALSE;
    ALPHA = .7;
}

LAYER
{
    GERBER = top_silk.grb;                  # top silk screen
    COLOR = 0.9, 0.9, 0.9;
    MIRROR = FALSE;
}

LAYER                                       # optional paste
{
    GERBER = top_paste_mask.grb;            # shows well on ENIG but not on HASL
    COLOR = 0.4, 0.4, 0.4;
    MIRROR = FALSE;
}
```

Rendering

With all of the pieces now in place, we can run PCB-Render to create the final image:

```
D:\Demo1\CAM> pcb-render board_top.pcbjob -o top_view.png
PCB dimensions: 154.18 x 80.77 mm
Resolution: 600.00 DPI, AA: 0.0428 mm, Margin: 5.00 mm
Rendering: 100 %   [9 of 9]
Writing PNG file...
```

and examine the results...



top_view.png

Bottom View

For a bottom view of the board, we need to run ***FpcPoly*** again to generate drilled versions of the bottom two copper layers. Since all layers will be mirrored in the bottom view, we'll also need to recreate the drop shadow file with the opposite X offset. Rename the top shadow file, used above, to something like **Top_Shadow.gbr** to save it or it will be overwritten.

```
C:\>fpcpoly N2 Dbottom_copper.grb Dinner_copper_2.grb o-0.07,-.13 Doutline_poly.gbr
```

And we'll also need another job file. An edited version of the one just used will work nicely:

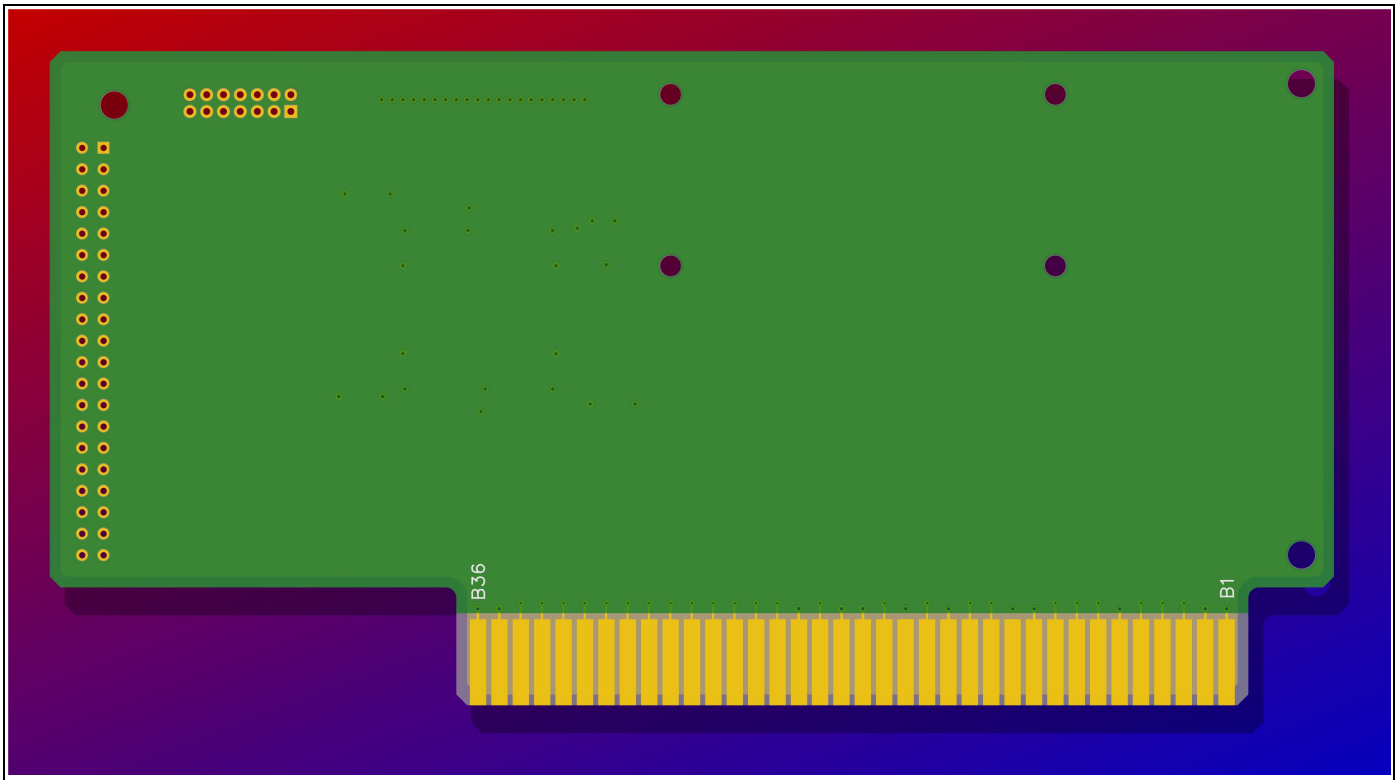
board_bottom.pcbjob:

```
GLOBAL
{
    DPI = 600;
    MARGIN = 5.0;
    AA = .001;                              # a value of zero is ignored...
}

BACKGROUND
{
    IMG = Background.png;                   # background first
}

LAYER                                       # stack up the image layers from bottom to top
{
    GERBER = outline_poly_drilledof1.gbr;   # drop shadow next
    COLOR = 0.0, 0.0, 0.0;                  # black
    MIRROR = TRUE;                          # all FALSE for top view, TRUE for bottom
    ALPHA = 0.3;                            # 70% transparent
}

LAYER
{
    GERBER = inner_copper_2_drilled.grb;    # lower copper layer next
    COLOR = 0.84, 0.59, 0.04;               # copper color
    MIRROR = TRUE;
}

LAYER
{
    GERBER = outline_poly_drilled.gbr;      # bottom dielectric
    COLOR = 0.6, 0.6, 0.6;                  # for Rogers 4000, for FR-4 use 0.4, 0.6, 0.4
    MIRROR = TRUE;
    ALPHA = .75;                            # 25% transparent to show lower copper
}

LAYER
{
    GERBER = bottom_copper_drilled.grb;     # bottom copper
    COLOR = 0.92, 0.75, 0.09;               # for ENIG Gold, for HASL use 0.6, 0.6, 0.6
    MIRROR = TRUE;
}

LAYER
{
    GERBER = bottom_smask_poly.gbr;         # bottom solder mask
    COLOR = 0.05, 0.5, 0.1;                 # light green LPI, for Vacril dry film use 0.4, 0.7, 0.4
    MIRROR = TRUE;
    ALPHA = .7;
}

LAYER
{
    GERBER = Bottom_silk.grb;               # bottom silk screen
    COLOR = 0.9, 0.9, 0.9;
    MIRROR = TRUE;
}

LAYER                                       # optional paste
{
    GERBER = bottom_paste_mask.grb;         # shows well on ENIG but not on HASL
    COLOR = 0.4, 0.4, 0.4;
    MIRROR = TRUE;
}
```

Run PCB-Render:

```
D:\Demo1\CAM> pcb-render board_bottom.pcbjob -o bottom_view.png
PCB dimensions: 154.18 x 80.77 mm
Resolution: 600.00 DPI, AA: 0.0428 mm, Margin: 5.00 mm
Rendering: 100 %   [9 of 9]
Writing PNG file...
```

and examine the results.



bottom_view.png

Use a Batch File

Long commands are prone to error and quickly become tedious. The whole process, with the exception of the background creation, can be quickly done with the aid of a short batch file:

build.bat:

```
@echo off
rem  Example batch file to automate the rendering process.
rem  If the background image does not exist, a test image is rendered and
rem  then execution will pause to allow background image creation.

fpcpoly N2 Dtop_copper.grb Dinner_copper_1.grb Doutline_poly.gbr o0.07,-0.13 doutline_poly.gbr
if exist Background.png goto render
pcb-render test.pcbjob -o test.png
echo.
echo  Examine test.png to determine image size, create
echo  the background image and when it is saved as
echo  Background.png ...
pause
:render
pcb-render board_top.pcbjob -o top_view.png
if exist top_shadow.gbr del top_shadow.gbr
rename outline_poly_drilledOf1.gbr top_shadow.gbr
fpcpoly N2 Dbottom_copper.grb Dinner_copper_2.grb o-0.07,-.13 Doutline_poly.gbr
pcb-render board_bottom.pcbjob -o bottom_view.png
echo on
```

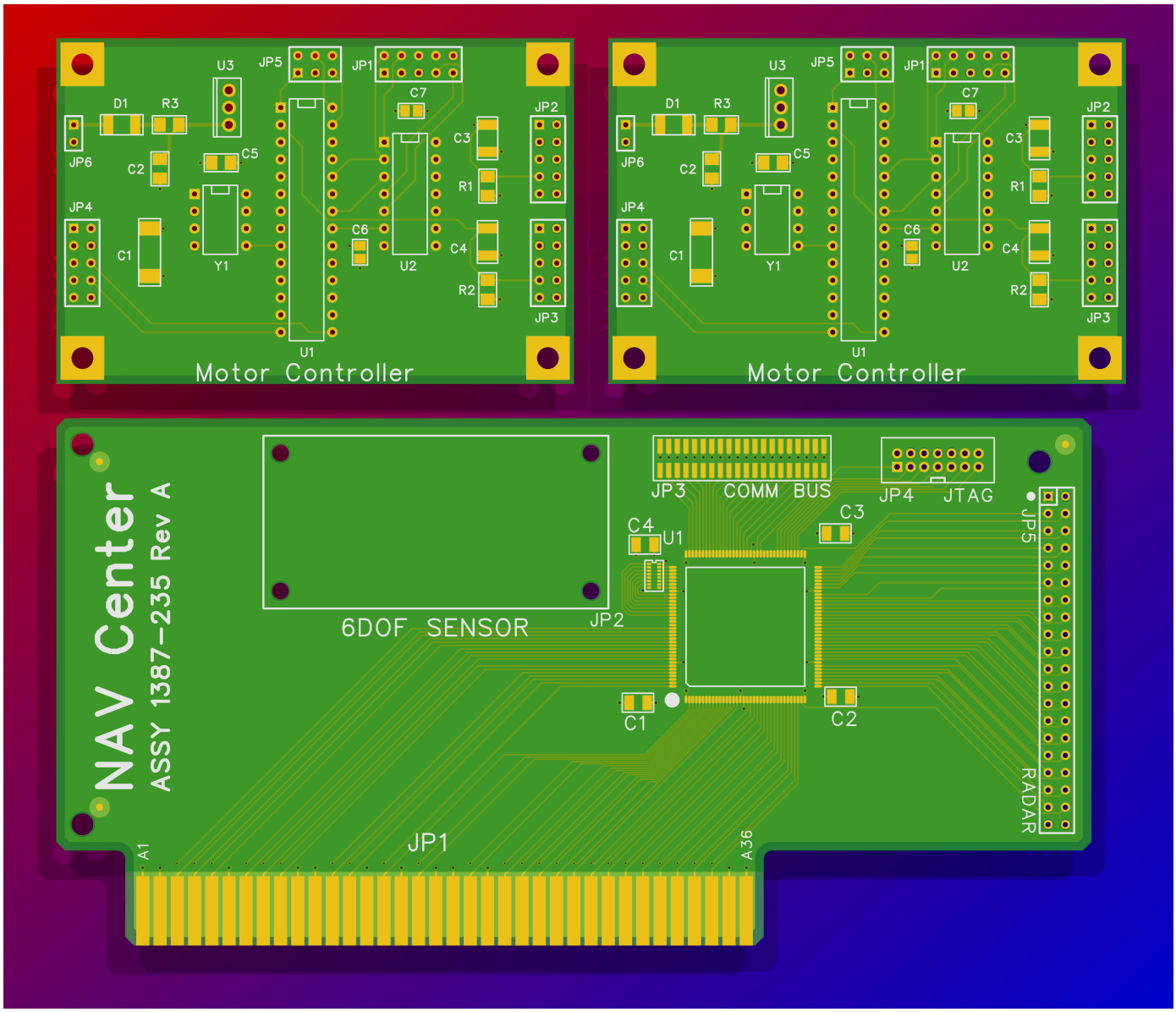
Final Thoughts

FpcPoly came about because I needed to impress my boss, in a design review, with a picture of my current project. After hand editing a few Gerber files *two or more time*, I realized that, with just a little code, I could simplify the process immensely. It's always *just a little code*. In any event, FpcPoly and the techniques outlined in this document are presented in the hope that some may find it useful.

Please note that the command and rendering scripts shown in this document and an expanded version of build.bat, with error checking, are all included in the Examples section of the distribution archive.

The examples presented here are by no means the only way to get the job done and should be considered as only a starting point.

Many variations are possible. For example, using a cropped photo of your work bench for the background or as show below, using two drop shadows to simulate dual light sources. Experiment. Have some fun!



FpcRef *Panelization Demo rendered with two drop shadows*

Handy Links

The links below can be used to download any of free tools that were referred to in this document or used in its preparation.

Gerber Viewers

GC-Prevue by Graphicode

ViewMate by PentaLogix

Image Editor

Paint.NET

Misc

FreePCB and PCB-Render

OpenOffice

History

Version 1.00	5 July 2008	Initial release with documentation and examples.
Version 1.10	8 July 2008	Added ***Sp*** (source path) switch option. User Guide updated.
Version 1.11	9 July 2008	Bug fix to handle solder mask files that included board outline data.
Version 1.12	11 July 2008	Minor bug fix in the help menu: Source Path switch was incorrect.
Version 1.13	10 Aug 2008	Added command script (C), script end (E), relative offset (R) and drill size exclusion (X) option switches. Capability to output the same drilled file with different offsets added. Example scripts updated.