Scripted Component Placement in PCB Files

John C. Luciani Jr.

October 6, 2005

1 Overview

The script `pcb-place` arranges components (elements) in a printed circuit board (PCB) layout using placement data from a configuration file. `pcb-place` uses the Perl module `Pcb_parse` which contains parsing and placement functions. These programs are meant to be used prior to performing manual component placement.

N.B. This script may not work for all layouts. I wrote sufficient code to work for one of my PCB layouts not for all possible PCB layouts.

Conventions

⟨*parameter*⟩	Replace with the value of *parameter*
`verbatim text`	Verbatim text when associated with a command or contents of a file.
`pcb-place`	The current version of the PCB component placement program.
`pcb-place_`⟨*version*⟩	Version ⟨*version*⟩ of the PCB component placement program.

2 Creating a Design

To create a design:

create a PCB file	run `gschpcb` on a schematic and create a new PCB file but do not perform any edits on the new file.
create a configuration file	create a configuration file using the commands defined in section 3.
create an output PCB	run the command `pcb-place` ⟨*configuration filename*⟩

3 Creating Configuration Files

Each file can contain comments, empty lines, lines that create global definitions and lines that define component placement parameters. Comments start with a `#` and proceed to the end of the current line. Comments and empty lines are ignored by the parser.

Lines that create global definitions are in the format:

`@def@` ⟨*definition name*⟩ `=` ⟨*definition value*⟩

Lines that define component placement parameters use the syntax:

⟨*component placement line*⟩ := ⟨*component placement definition*⟩ (| ⟨*component placement definition*⟩)⋆
⟨*component placement definition*⟩ := ⟨*parameter name*⟩ = ⟨*parameter value*⟩

where ⟨*parameter name*⟩ is one of the names listed in Table 1.

Name	Type of Value	Description
x	Integer	the x coordinate (hundreths of a mil) of the component reference point in the output file.
y	Integer	the y coordinate (hundreths of a mil) of the component reference point in the output file.
input_filename	String	the name of the file containing the initial PCB layout.
output_filename	String	the name of the file containing the new PCB layout.
refdes	String	the reference designator of the component
element_side	String	the side of the PBC layout to place the component on. Recognized values are `top`, `bottom`, `component` and `solder`.
rotation	Integer	The counter-clockwise angle (in degrees) to rotate the component by. The value must be either `0`, `90`, `180`, `270`.

Table 1: Configuration File Parameters

4 A Simple Example

Listing 1 contains a basic program that parses a PCB layout (line 12), places a component (line 16) and then prints the PCB layout to STDOUT (line 18).

Listing 1: A Simple Example that uses Pcb_parse

```
 1  #!/usr/bin/perl
 2
 3  use strict;
 4  use warnings;
 5  use Carp;
 6  use Data::Dumper;
 7
 8  use Pcb_parse_1;
 9
10  # Create a PCB parse object and load the file ’test.pcb’ into a hash.
11
12  my $Pcb = Pcb_parse_1 -> new(filename => ’test.pcb’);
13
14  # Place R1 at (1in,1in)
15
16  $Pcb -> component_place(refdes => ’R1’, x => 100000, y => 100000);
17
18  print $Pcb -> pcb_layout; #
```

5 A Real Example

Figure 1 shows the top and bottom layers of the PCB created using `pcb-place` and the configuration file in listing 3. `D1-D48` are on the top layer, `J1` and `J2` are on the bottom layer. Alternatively the Perl script in listing 2 can be used to produce the same layout.

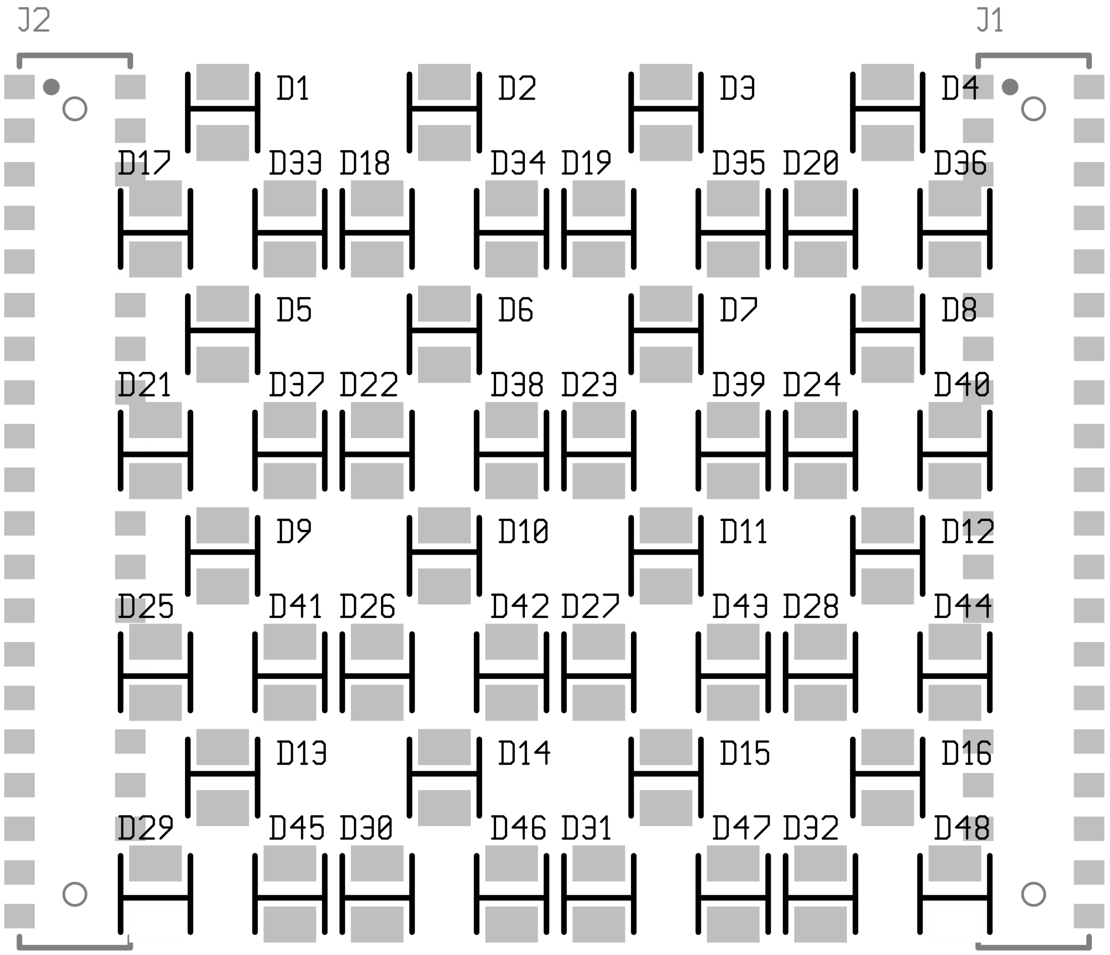

Figure 1: RGB LED Matrix

5.1 Perl Program

Listing 2: Perl program to create an RGB LED Matrix

```perl
#!/usr/bin/perl

use strict;
use warnings;
use Carp;
use Data::Dumper;

use Pcb_parse_1;
use Math::Trig;


# Initialize a PCB parsing object and read the file 'led.pcb' into a
# hash.

my $Pcb = Pcb_parse_1 -> new(filename => 'led.pcb');


# There 16 groups of LEDs and each group of LEDs has a red, green and
# blue LED.  The LEDs in a group are placed along a circle at 120
# degree angles.

my $space  = 40000; # spacing between groups
my $radius = 14000; # radius of the circle
my $led_dx = $radius * sin(deg2rad(60));
my $led_dy = $radius * cos(deg2rad(60));



my ($x0, $y0) = (105000, 87750); # starting position
my ($x, $y)   = ($x0, $y0);      # current placement position

# create a 4x4 matrix of diodes.

foreach (1..16) {

    # diodes 1-16 are placed at the top of the circle,
    # diodes 17-32 are placed on the left side of the circle
    # diodes 33-48 are placed on the right side of the circle

    # (x,y) centroid of the component
    # (text_dx, text_dy) offset from (x,y) of the text
    # rotation angle (degrees) that the component is rotated

    $Pcb -> component_place(refdes => "D$_",
                            x => $x,
                            y => $y - $radius,
                            text_dx => 10000,
                            text_dy => -7500,
                            rotation => 270);
    $Pcb -> component_place(refdes => sprintf("D%i", $_+16),
                            x => $x - $led_dx,
                            y => $y + $led_dy,
                            text_dx => -6500,
                            text_dy => -15000,
                            rotation => 90);
    $Pcb -> component_place(refdes => sprintf("D%i", $_+32),
                            x => $x + $led_dx,
                            y => $y + $led_dy,
                            text_dx => -3500,
                            text_dy => -15000,
                            rotation => 90);

    # After four LEDs are placed start a new row.

     if ($_ % 4) {
        $x += $space;
     } else {
        $x = $x0;
        $y += $space;
     }
}
```

```

# Place all of the components specified in the data section

while (<DATA>) {
    s/\#.*//; # Remove comments
    s/^\s*//; # Remove leading spaces
    s/\s*$//; # Revove trailing spaces
    next unless length;     # Skip empty lines
    if (s/\\\s*$//) {       # Remove the continuation backslash and
        $_ .= <>;           # append the next line to $_ then
        redo unless eof;    # restart the loop block after the conditional
    }
    my @values = split /\s*\\s*\s*=\s*/;
    $Pcb -> component_place(@values);
}


# open a new file and output the PCB layout

open(OUT, ">led-new.pcb")    die "could not open led-new.pcb for output";
print OUT $Pcb -> pcb_layout;
close(OUT)    die "could not close led-new.pcb";


__DATA__

# These are the two connectors that are on the bottom of the RGB matrix.

refdes = J1   element_side = bottom   x = 251330   y = 143942   rotation = 270
    text_dx = -10000   text_dy = -90000
refdes = J2   element_side = bottom   x = 78330    y = 143942   rotation = 270
    text_dx = -10000   text_dy = -90000
```

5.2 Configuration File

Listing 3: Example Configuration File

```
#
# configuration file to place two connectors
#

@def@ input_filename=test.pcb
@def@ output_filename= test-new.pcb

refdes= D1   x=105000   y=73750    text_dx=10000   text_dy=-7500    rotation=270
refdes= D2   x=145000   y=73750    text_dx=10000   text_dy=-7500    rotation=270
refdes= D3   x=185000   y=73750    text_dx=10000   text_dy=-7500    rotation=270
refdes= D4   x=225000   y=73750    text_dx=10000   text_dy=-7500    rotation=270
refdes= D5   x=105000   y=113750   text_dx=10000   text_dy=-7500    rotation=270
refdes= D6   x=145000   y=113750   text_dx=10000   text_dy=-7500    rotation=270
refdes= D7   x=185000   y=113750   text_dx=10000   text_dy=-7500    rotation=270
refdes= D8   x=225000   y=113750   text_dx=10000   text_dy=-7500    rotation=270
refdes= D9   x=105000   y=153750   text_dx=10000   text_dy=-7500    rotation=270
refdes=D10   x=145000   y=153750   text_dx=10000   text_dy=-7500    rotation=270
refdes=D11   x=185000   y=153750   text_dx=10000   text_dy=-7500    rotation=270
refdes=D12   x=225000   y=153750   text_dx=10000   text_dy=-7500    rotation=270
refdes=D13   x=105000   y=193750   text_dx=10000   text_dy=-7500    rotation=270
refdes=D14   x=145000   y=193750   text_dx=10000   text_dy=-7500    rotation=270
refdes=D15   x=185000   y=193750   text_dx=10000   text_dy=-7500    rotation=270
refdes=D16   x=225000   y=193750   text_dx=10000   text_dy=-7500    rotation=270

refdes=D17   x=92875    y=94750    text_dx=-6500   text_dy=-15000    rotation=90
refdes=D18   x=132875   y=94750    text_dx=-6500   text_dy=-15000    rotation=90
refdes=D19   x=172875   y=94750    text_dx=-6500   text_dy=-15000    rotation=90
refdes=D20   x=212875   y=94750    text_dx=-6500   text_dy=-15000    rotation=90
refdes=D21   x=92875    y=134750   text_dx=-6500   text_dy=-15000    rotation=90
refdes=D22   x=132875   y=134750   text_dx=-6500   text_dy=-15000    rotation=90
refdes=D23   x=172875   y=134750   text_dx=-6500   text_dy=-15000    rotation=90
refdes=D24   x=212875   y=134750   text_dx=-6500   text_dy=-15000    rotation=90
refdes=D25   x=92875    y=174750   text_dx=-6500   text_dy=-15000    rotation=90
refdes=D26   x=132875   y=174750   text_dx=-6500   text_dy=-15000    rotation=90
refdes=D27   x=172875   y=174750   text_dx=-6500   text_dy=-15000    rotation=90
refdes=D28   x=212875   y=174750   text_dx=-6500   text_dy=-15000    rotation=90
refdes=D29   x=92875    y=214750   text_dx=-6500   text_dy=-15000    rotation=90
refdes=D30   x=132875   y=214750   text_dx=-6500   text_dy=-15000    rotation=90
refdes=D31   x=172875   y=214750   text_dx=-6500   text_dy=-15000    rotation=90
refdes=D32   x=212875   y=214750   text_dx=-6500   text_dy=-15000    rotation=90

refdes=D33   x=117124   y=94750    text_dx=-3500   text_dy=-15000    rotation=90
refdes=D34   x=157124   y=94750    text_dx=-3500   text_dy=-15000    rotation=90
refdes=D35   x=197124   y=94750    text_dx=-3500   text_dy=-15000    rotation=90
refdes=D36   x=237124   y=94750    text_dx=-3500   text_dy=-15000    rotation=90
refdes=D37   x=117124   y=134750   text_dx=-3500   text_dy=-15000    rotation=90
refdes=D38   x=157124   y=134750   text_dx=-3500   text_dy=-15000    rotation=90
refdes=D39   x=197124   y=134750   text_dx=-3500   text_dy=-15000    rotation=90
refdes=D40   x=237124   y=134750   text_dx=-3500   text_dy=-15000    rotation=90
refdes=D41   x=117124   y=174750   text_dx=-3500   text_dy=-15000    rotation=90
refdes=D42   x=157124   y=174750   text_dx=-3500   text_dy=-15000    rotation=90
refdes=D43   x=197124   y=174750   text_dx=-3500   text_dy=-15000    rotation=90
refdes=D44   x=237124   y=174750   text_dx=-3500   text_dy=-15000    rotation=90
refdes=D45   x=117124   y=214750   text_dx=-3500   text_dy=-15000    rotation=90
refdes=D46   x=157124   y=214750   text_dx=-3500   text_dy=-15000    rotation=90
refdes=D47   x=197124   y=214750   text_dx=-3500   text_dy=-15000    rotation=90
refdes=D48   x=237124   y=214750   text_dx=-3500   text_dy=-15000    rotation=90

refdes=J1   element_side=bottom   x=251330   y=143942   rotation=270
refdes=J2   element_side=bottom   x=78330    y=143942   rotation=270
```

References

Eaton, H., & Nau, T. (2002). Pcb [Computer software and manual]. (Retrieved February 6, 2005 from http://pcb.sourceforge.net/pcb-20050127.html/index.html)

6 License

No-Fee License Version 0.1

Intent

The intent of this license is to allow for distribution of this documentation without fee. Usage of this documentation, other than distribution, is unrestricted.

License

This documentation may be distributed provided that no fee is charged. Modification is allowed provided that the resultant work is released with the same license.

This documentation is provided by the author **"AS IS"** and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the author be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.