Creating PCB Elements with Perl

John C. Luciani Jr.

June 24, 2007

1 Pcb_9

This document describes a set of Perl routines that can be used to create component footprints for the circuit board layout program PCB. These routines reside in a file called `Pcb_`⟨n⟩`.pm` where ⟨n⟩ is the current revision number of the package. Only the new format of PCB elements is output. The differences (that I am aware of) between the old and new formats are:

- Dimensions are in hundreths of a mil.
- The argument delimiters are square brackets `[]`
- The element command adds the `mark_x` and `mark_y` parameters
- The pin and pad command add clearance and mask parameters.

Requirements

These routines should run with a standard Perl distribution. The only packages used are `POSIX` and `Carp`.

Usage

These routines are object oriented. A PCB object is created using `new` and all subsequent method calls use this object. `element_begin` starts a new element. `element_output` outputs the element file. `element_add_mark` sets the component centroid. `element_set_text_xy` sets the text position for the reference designator. The names of the methods used to draw elements all start with the string `element_add`. Arguments for the method calls are key-value pairs. The keys are parameter strings defined in pcb.html.

To use these routines in a Perl script to create a PCB element:

1. Include the PCB routines `use Pcb_`⟨n⟩`;`
2. Create a PCB object using `new`
3. Begin an element using `element_begin`
4. Add copper to the element using `element_add_pin` or `element_add_pad`
5. Add silkcreen elements using `element_add_line`, `element_add_arc`,
6. Mark the centroid using `element_add_mark`. The mark can also be set using parameters of the `element_begin` method.
7. Add the text location for the reference designator using `element_set_text_xy` The text location can also be set using parameters of the `element_begin` method.

8. Output the element to a file using `element_output`

The simple example in Listing 1 creates a quarter watt through-hole resistor. The example in Listing 3 creates a variety of two terminal SMD footprints ranging in size from `0402` to `2512`. The example in Listing 5 creates Molex 8624 series header connector footprints. The example in Listing 6 creates TQFN footprints for a variety of Maxim parts. These examples place files in the directory `./tmp`. This can be easily changed by changing the `element_begin` call.

Listing 1: 1/4 Watt Resistor Example

```
#!/usr/bin/perl

# Copyright (C) 2005 John C. Luciani Jr.

# This program may be distributed or modified under the terms of
# version 0.2 of the No-Fee Software License published by
# John C. Luciani Jr.

# Creates a 1/4 Watt resistor

use strict;
use warnings;

use Pcb_8;

my $Pcb = Pcb_8 -> new(debug => 1);

$Pcb -> element_begin(description => 'resistor',
                      output_file => '025W',
                      dim   => 'mils');

# the resistor centroid is at (0,0) and the pins are placed 400 mils
# apart

my $Body_width = 70;           # y direction
my $Body_length = 240;         # x direction
my @Pins = (-200, 0, 200, 0); # x,y pin centers
my $Pin_num = 1;

while (@Pins) {
    my ($x, $y) = splice @Pins, 0, 2;
    $Pcb -> element_add_pin(x => $x, y => $y,
                            thickness  => 55,
                            drill_hole => 35,
                            mask => 10,
                            clearance => 10,
                            pin_number => $Pin_num++);
}

$Pcb -> element_add_rectangle(width => $Body_width,
                              length=> $Body_length,
                              thickness => 10,
                              x => 0,
                              y => 0);

foreach my $sign (-1, 1) {
    $Pcb -> element_add_line(x1 => $sign * $Body_length / 2,
                             y1 => 0,
                             x2 => $sign * ($Body_length / 2 + 30),
                             y2 => 0,
                             thickness => 10);
}


$Pcb -> element_set_text_xy(x => -$Body_length/2,
                            y => -$Body_width/2 - 20);


$Pcb -> element_output();
```

2 new

Usage

`Pcb_9->new(` ⟨*parameter list*⟩ `)`

Description

Creates an object that is used to make PCB element files. Default parameters for the various element drawing commands can be initialized using a key-value parameter list. The valid keys and default values are in Table 1

Parameter Name	Default Value	Notes
line_thickness	10	thickness used in drawing silkscreen lines
arc_thickness	10	thickness used in drawing silkscreen arcs
thickness	10	thickness used in drawing any silkscreen line
pin_flags	0	flags used in creating element pins (See Table 16)
pad_flags	PAD_SQUARE	flags used in creating pads
font_size	50	size in ??? of the silkscreen found
clearance	10	separation of pad from other conductors on the layer .
mask	10	distance between the edge of the solder mask and the copper pad. This definition differs fromthe **PCB** definition of mask.
debug	0	debug messages. no messages (0). object methods (1). object methods + internal subroutines (2)

Table 1: Keys for Method new

Example

To create a new object that will display object method debugging messages:

```
my $Pcb = Pcb_9 -> new(debug => 1);
```

3 element_begin

Usage

`Pcb->element_begin(` ⟨*parameter list*⟩ `)`

Description

Initializes a new Pcb element. If an element was previously created but not output a call to `element_begin` will remove it. The valid keys and default values are in Table 2

Parameter Name	Default Value	Notes
flags	0	Element flags. See Table 13
description	''	Text description of the footprint
layout_name	''	Reference designator used on the PCB.
value	''	Value of component in the PCB. Leave blank.
mark_x	0	X location of the footprint mark (in mils)
mark_y	0	Y location of the footprint mark (in mils)
text_x	0	X location of the refdes text (in mils)
text_y	0	Y location of the refdes text (in mils)
direction	0	Text direction flags. See Table 14
scale	100	Text scale.
text_flags	0	See Table 15
output_file	'PCB_ELEMENT.TMP'	Element filename
pin_one_square	0	Sets a default value that is used when creating a pin.
dim	'mils'	units default to mils

Table 2: Keys for Method element_begin

Example

To begin a 1/4 Watt resistor element with dimension values in mils:

```
$Pcb -> element_begin(description => 'resistor',
                      output_file => '025W',
                      dim   => 'mils');
```

4 element_output

Usage

`Pcb->element_output(` ⟨*parameter list*⟩ `)`

Description

`element_output` outputs the element drawing commands to a file. At this time there are no parameters that are valid for the ⟨*parameter list*⟩.

5 element_add_line

Usage

`Pcb->element_add_line(` ⟨*parameter list*⟩ `)`

Description

Creates a silkscreen line of a specified thickness (`thickness`) between two points (`x1`, `y1`) and (`x2`, `y2`).

Parameter Name	Default Value	Notes
x1		X coordinate of the first point.
y1		Y coordinate of the first point.
x2		X coordinate of the second point.
y2		Y coordinate of the second point.
thickness		Width of the line.

Table 3: Keys for Method element_add_line

Example

To create a 200mil long silkscreen line that is centered at `(0,0)` that is 10 mils thick

```
$Pcb -> element_add_line(x1 => -100, y1 => 0,
                         x2 =>  100, y2 => 0,
                         thickness => 10);
```

6 element_add_arc

Usage

`Pcb->element_add_arc(` ⟨*parameter list*⟩ `)`

Description

Creates a silkscreen arc with a specified `width` and `length` centered at a point (`x1`, `y1`).

Parameter Name	Default Value	Notes
x		X coordinate of the point.
y		Y coordinate of the point.
width		horizontal width of the arc
height		vertical length of the arc
start_angle		Starting angle of the arc (degrees)
delta_angle		Angle swept by the arc (degrees)
thickness		line thickness

Table 4: Keys for Method element_add_arc

Example

To create a silkscreen circular arc centered at `(0,0)` with a line thickness of 10 mils, radius of 200 mils that starts at 45° and sweeps for 135°:

```
$Pcb -> element_add_arc(start_angle => 45,
                        delta_angle => 135,
                        x => 0,
                        y => 0,
                        width => 200,
                        height => 200,
                        thickness => 10);
```

For an ellipse set the width and height to unequal values.

7 element_add_pin

Usage

`Pcb->element_add_pin(` ⟨*parameter list*⟩ `)`

Description

Adds a pin to an element

Parameter Name	Default Value	Notes
x		X coordinate of the point.
y		Y coordinate of the point.
thickness		width of the copper pad
clearance		separation of pad from other conductors on the layer .
mask		distance between the edge of the solder mask and the copper pad. This definition differs fromthe **PCB** definition of mask.
drill_hole		diameter of the hole that is drilled at the center of the pad
name		string
pin_number		The pin number of the component pin that will inserted at this position.
flags		See Table 16

Table 5: Keys for Method element_add_pin

Example

To place a pin with a round pad at `(-100,0)` with a pad diameter of 55 mils, a drill hole diameter of 35 mils, soldermask clearance of 10 mils, a copper clearance of 9 mils, and a pin number of one:

```
$Pcb -> element_add_pin(x => -100, y => 0,
                        thickness  => 55,
                        drill_hole => 35,
                        mask => 10,
                        clearance => 9,
                        pin_number => 1);
```

8 element_add_pad

Usage

`Pcb->element_add_pad(` ⟨*parameter list*⟩ `)`

Description

Pads are created by drawing a line, with a specified thickness, between two points. The line is drawn with a square nib and extends beyond each end point by a distance of $\frac{\texttt{thickess}}{2}$.

Parameter Name	Default Value	Notes
x1		X coordinate of the first point.
y1		Y coordinate of the first point.
x2		X coordinate of the second point.
y2		Y coordinate of the second point.
thickness		Width of the line.
clearance		separation of pad from other conductors on the layer .
mask		distance between the edge of the solder mask and the copper pad. This definition differs fromthe **PCB** definition of mask.
name		Identification string for the pad
pad_number		The pin number of the component that will reside on the pad.
flags		See Table 17

Table 6: Keys for Method element_add_line

Example

To create a pad that is centered at `(0,0)` that is 100 mils long and 50 mils thick
has a soldermask clearance of 10 mils, a copper clearance of 9 mils and is numbered one:

```
$Pcb -> element_add_pad(x1 => -25, y1 => 0,
                        x2 => 25,  y2 => 0,
                        thickness  => 50,
                        mask       => 10,
                        clearance  => 9,
                        pad_number => 1);
```

9 element_add_pad_rectangle

Usage

`Pcb->element_add_pad_rectangle(` ⟨*parameter list*⟩ `)`

Description

Create a pad with a specified width and length that is centered at a point `(x,y)`. The length is in x-direction and the width is in the y-direction.

Parameter Name	Default Value	Notes
x		X coordinate of the point.
y		Y coordinate of the point.
width		The pad width (y direction)
length		The pad length (x direction)
clearance		separation of pad from other conductors on the layer .
mask		distance between the edge of the solder mask and the copper pad. This definition differs fromthe **PCB** definition of mask.
name		Identification string for the pad
pin_number		The pin number of the component pin that will inserted at this position.

Table 7: Keys for Method element_add_pad_rectangle

10 element_add_pin_oval

Usage

`Pcb->element_add_pin_oval(` ⟨*parameter list*⟩ `)`

Description

Create a pad with a specified width and length that is centered at a point `(x,y)`. The length is in x-direction and the width is in the y-direction. The corners of the pad are rounded.

This is actually a hybrid object consisting of a component side pad, a solder side pad and a pin placed at the same center point.

Parameter Name	Default Value	Notes
x		X coordinate of the point.
y		Y coordinate of the point.
width		The pad width (y direction)
length		The pad length (x direction)
drill_hole		diameter of the hole that is drilled at the center of the pad
name		Identification string for the pad
pin_number		The pin number of the component pin that will inserted at this position.

Table 8: Keys for Method element_add_pin_oval

Example

To place a pin with an oval pad at `(-100,0)` with a pad diameter of 55 mils, a drill hole diameter of 35 mils, soldermask clearance of 10 mils, a copper clearance of 9 mils, and a pin number of one:

```
$Pcb -> element_add_pin_oval(x => -100, y => 0,
                             thickness  => 55,
                             drill_hole => 35,
                             mask       => 10,
                             clearance  => 9,
                             pin_number => 1);
```

11 element_add_mark

Usage

`Pcb->element_add_mark(` ⟨*parameter list*⟩ `)`

Description

The mark is a positioning hint. `element_add_mark` places the mark at at a point (`x1`, `y1`).

Parameter Name	Default Value	Notes
x		X coordinate of the point.
y		Y coordinate of the point.

Table 9: Keys for Method element_add_mark

12 element_add_lines

Usage

`Pcb->element_add_lines(` ⟨*parameter list*⟩ `)`

Description

Draws silkscreen lines using the specified line end points. Lines are drawn from point to point until all the points are connected.

Parameter Name	Default Value	Notes
points		reference to a list containing x,y coordinates for line end points.
thickness		Width of the line.

Table 10: Keys for Method element_add_lines

13 element_add_rectangle

Usage

`Pcb->element_add_rectangle(` ⟨*parameter list*⟩ `)`

Description

Draws a silkscreen rectangle with a specified `width` and `length` at a point (`x1`, `y1`).

Parameter Name	Default Value	Notes
x		X coordinate of the point.
y		Y coordinate of the point.
width		rectangle width (y direction)
length		rectangle length (x direction)
thickness		Width of the line.

Table 11: Keys for Method element_add_rectangle

14 element_set_text_xy

Usage

`Pcb->element_set_text_xy(` ⟨*parameter list*⟩ `)`

Description

Sets the position of the reference designator text.

Parameter Name	Default Value	Notes
x		X coordinate of the point.
y		Y coordinate of the point.
font_size		

Table 12: Keys for Method element_add_mark

15 element_set

Usage

`Pcb->element_set(` ⟨*parameter list*⟩ `)`

Description

Sets values in the element hash table. This should be the only method used to set values in the element hash. ⟨*parameter list*⟩ contains key-value pairs.

16 element_get

Usage

`Pcb->element_get(` ⟨*parameter list*⟩ `)`

Description

Returns a value, from the element hash, for each key specified in ⟨*parameter list*⟩. If the value is undefined in the element hash then a value from the Pcb object hash is returned. A value of `undef` is returned if neither hash contains a defined value for the key.

This should be the only method used to retrieve values from the element hash. ⟨*parameter list*⟩ contains a list of keys.

17 get

Usage

`Pcb->get(` ⟨*parameter list*⟩ `)`

Description

Retrieves values from the PCB object hash. This should be the only method used to retrieve values from the PCB object hash. ⟨*parameter list*⟩ contains a list of keys.

18 element_dump

Usage

`Pcb->element_dump(` ⟨*parameter list*⟩ `)`

Description

A debugging procedure that Prints out the element command drawing commands to STDOUT.

References

Brorson, S. D., & Meier, S. (2005, January). Footprint creation for the open-source layout program PCB. (Retrieved February 6, 2005, from http://www.brorson.com/gEDA/land_patterns_20050129.pdf)

Eaton, H., & Nau, T. (2002). Pcb [Computer software and manual]. (Retrieved February 7, 2007 from http://pcb.sourceforge.net/pcb-cvs/pcb.html)

19 Change Log

Pcb_9	???	jcl	1. Fixed a dimension scaling bug in element_add_lines. The scaling routine now scales an array of points. This bug was reported by Ben Jackson. 2. The scaling routines now accept a dimension suffix which will override the default dimension.
Pcb_8	19-Mar-2007	jcl	1. Removed the export of element_add_arc. Not necessary (OO). 2. Corrected the mask and clearance parameters in the pin, pad and pin_oval procedures. 3. Removed the Mark command since the mark data is now in the Element header. 4. Exported element_str and added a scale_factor parameter (default value of 100) 5. the key to specify dimensional units (input_dim) was changed to dim 6. Fixed the dimension scaling problem in `add_element_lines` 7. Corrected the documentation for `element_add_pin_oval`
Pcb_7	25 March 2005	jcl	1. changed the definition of the mask and clearance. 2. Fixed the mask and clearance parameters in the pin, pad and pin_oval procedures.
Pcb_6	22 March 2005	jcl	1. The `element_add_rectangle` command now uses the `x` and `y` parameters. The center of the rectangle was always placed at (0,0) 2. The `pin_one_square` key-value pair was not getting properly tested in the `element_add_pin` procedure. 3. Added the `clearance` and the `mask` parameters to `element_add_pad_rectangle`.
Pcb_5	6 March 2005	jcl	1. Added the `element_add_lines` command. 2. added the `element_add_pin_oval` command. 3. Modified the debug print messages. 4. Fixed constant for octagonal pads. 5. Fixed errors in the EXPORT_OK and EXPORT_TAGS declarations. 6. Added `element_get_names`.
Pcb_4	27 February 2005	jcl	1. Modified the debug strings to output mm and mils. 2. Fixed the `scale_factor` subroutine. `scale_factor` did not correctly convert from mils to mm. I did not test (or use) the conversion to mm until I modified the debug strings
Pcb_3	7 February 2005	jcl	Initial Release

20 Element Flags

The element flag field determines the state of an element. The bit values are:

Parameter Name	Default Value	Notes
ELEMENT_NAME_HIDDEN	0x10	bit 4: the element name is hidden
ELEMENT_SELECTED	0x40	bit 6: element has been selected
ELEMENT_SOLDER_SIDE	0x80	bit 7: element is located on the solder side

Table 13: Element Flags

21 Text Flags

Parameter Name	Default Value	Notes
TEXT_DIRECTION_0	0	Horizontal
TEXT_DIRECTION_90	1	90 degrees counter-clockwise
TEXT_DIRECTION_180	2	180 degrees counter-clockwise
TEXT_DIRECTION_270	3	270 degrees counter-clockwise

Table 14: Text Direction Flags

Parameter Name	Default Value	Notes
TEXT_SELECTED	0x40	bit 6: the text has been selected
TEXT_ON_SOLDER_SIDE	0x80	bit 7: the text is on the solder (back) side of the board
TEXT_ON_SILKSCREEN	0x400	bit 10: the text is on the silkscreen layer

Table 15: Text Flags

22 Pin Flags

Parameter Name	Default Value	Notes
PIN_MASK	0xFFFD	
PIN_ALWAYS_SET	0x0001	bit 0: always set
		bit 1: always clear
PIN_CONNECTED	0x0004	bit 2: set if pin was found during a connection search
PIN_MOUNTING_HOLE	0x0008	bit 3: set if pin is only a mounting hole (no copper annulus)
PIN_DISPLAY_NAME	0x0020	bit 5: display the pins name
PIN_SELECTED	0x0040	bit 6: pin has been selected
PIN_SQUARE	0x0100	bit 8: pin is drawn as a square
PIN_OCTAGONAL	0x0800	bit 12: set if pin is drawn with an octagonal shape
PIN_ROUND	0x0000	
PIN_SHAPE_MASK	0xEEFF	

Table 16: Pin Flags

23 Pad Flags

Parameter Name	Default Value	Notes
PAD_CONNECTED	0x0004	bit 2: set if pad was found during a connection search
PAD_DISPLAY_NAME	0x0020	bit 5: display the pads name
PAD_SELECTED	0x0040	bit 6: pad has been selected
PAD_SOLDER_SIDE	0x0080	bit 7: pad is located on the solder side
PAD_SQUARE	0x0100	
PAD_ROUNDED	0x0800	bit 11: pad has rounded corners

Table 17: Pad Flags

24 Examples

Listing 2: TO220 Pads

```
#!/usr/bin/perl

# Copyright (C) 2007 John C. Luciani Jr.

# This program may be distributed or modified under the terms of
# version 0.2 of the No-Fee Software License published by
# John C. Luciani Jr.

# Places three rounded pads with holes spaced 100 mils.

use strict;
use warnings;

use Pcb_8;

my $Pcb = Pcb_8 -> new(debug => 1);

$Pcb -> element_begin(description => 'TO220-pads',
                      output_file =>
                         "tmp/" . 'TO220-pads',
                      dim   => 'mils');

my $pin_num = 1;
my @pos = (-100, 0, 0, 0, 100, 0);

while (@pos) {
    my ($x, $y) = splice @pos, 0, 2;
    $Pcb -> element_add_pin_oval(x => $x,
                                 y => $y,
                                 width => 80,
                                 length => 66,
                                 name => '',
                                 pin_number => $pin_num++,
                                 clearance => 10,
                                 drill_hole => 46,
                                 mask => 10);
}


$Pcb -> element_output();
```

Listing 3: SMD Element Creation Example

```
#!/usr/bin/perl

# Copyright (C) 2005 John C. Luciani Jr.

# This program may be distributed or modified under the terms of
# version 0.2 of the No-Fee Software License published by
# John C. Luciani Jr.

# Creates the PCB elements specified in the DATA section.  The
# footprints are for the SMD packages 0402, 0603, 0805, 1206, 1210,
# 2010, 2512, 0402, 0504, 0603, 0805, 1206, 1210, 1812, 1825

use strict;
use warnings;

use Pcb_9;

my $Pcb = Pcb_9 -> new(debug => 1);

my @Fields = qw(land_pattern_length  land_row_distance
                land_width           land_length
                land_row_centers     grid);

while (<DATA>) {
    s/\#.*//; # Remove comments
    s/^\s*//; # Remove leading spaces
    s/\s*$//; # Revove trailing spaces
    next unless length; # Skip empty lines
    my ($type, @values) = split /\s*\\s*/;

    # hash for each footprint

    my %f = map { $_ => shift(@values) } @Fields;

    $Pcb -> element_begin(description => 'SMD',
                          output_file => "tmp/$type",
                          dim   => 'mm');

    my $x = -$f{land_row_centers} / 2;
    foreach my $pin_num (1..2) {
        $Pcb -> element_add_pad_rectangle(width => $f{land_width},
                                          length=> $f{land_length},
                                          x => $x,
                                          y => 0,
                                          name => 'input',
                                          mask => 0.254,
                                          clearance => 0.254,
                                          pin_number => $pin_num);
        $x += $f{land_row_centers};
    }

    # Draw a silkscreen rectangle around the component.  A silkscreen
    # specification that all PCB vendors should be able to meet is
    # 10mil line width and 10mil spacing. The silkscreen line width
    # defaults to 10mils. To achieve the proper spacing we add
    # 30mils (0.762mm) to the maximum extents of the copper pads
    # (10mils on either side of the copper and 2*5 mils for the
    # silkscreen line).

    my $length = $f{land_pattern_length} + 0.762;
    my $width  = $f{land_width} + 0.762;

    $Pcb -> element_add_rectangle(width => $width,
                                  length=> $length,
                                  x => 0,
                                  y => 0);

    # Place the refdes slightly (0.5mm) above the upper left corner of
    # the outline rectangle.

    $Pcb -> element_set_text_xy(x => -$length/2,
                                y => -$width/2 - 0.5);
```

```
73
74      $Pcb -> element_output();
75
76  }
77
78
79  __DATA__
80
81  # type    package name
82  #  Z      overall length of land pattern
83  #  G      distance between land rows
84  #  X      land width
85  #  Y      land length
86  #  C      center-to-center spacing between land rows
87  #  Grid   number of 0.5mm by 0.5mm elements
88
89  # type      Z         G         X          Y         C      Grid
90
91   0402      2.20      0.40      0.70      0.90      1.30    2x6
92   0504      2.40      0.40      1.30      1.00      1.40    4x6
93   0603      2.80      0.60      1.00      1.10      1.70    4x6
94   0805      3.20      0.60      1.50      1.30      1.90    4x8
95   1206      4.40      1.20      1.80      1.60      2.80    4x10
96   1210      4.40      1.20      2.70      1.60      2.80    6x10
97   1812      5.80      2.00      3.40      1.90      3.90    8x12
98   1825      5.80      2.00      6.80      1.90      3.90    14x12
99   2010      6.20      2.60      2.70      1.80      4.40    6x14
100  2512      7.40      3.80      3.20      1.80      5.60    8x16
```

Listing 4: Header Connector Creation Example 1

```
#!/usr/bin/perl

# Copyright (C) 2005 John C. Luciani Jr.

# This program may be distributed or modified under the terms of
# version 0.2 of the No-Fee Software License published by
# John C. Luciani Jr.

# Creates the PCB elements for Molex 8624 header connectors

use strict;
use warnings;

use Pcb_8;

my $Pcb = Pcb_8 -> new(debug => 1);

my @Fields = qw(circuits body_length pin_row_length);

my @Def; # definitions that are common to all components

while (<DATA>) {
    s/\#.*//; # Remove comments
    s/^\s*//; # Remove leading spaces
    s/\s*$//; # Revove trailing spaces
    next unless length; # Skip empty lines

    # Lines that contain an '=' are global definitions.

    push(@Def, $1, $2), next if /(\S+)\s*=\s*(\S.*)/;

    my @values = split /\s*\,\s*/;

    # hash for each footprint

    my %f = ( @Def,
              map { $_ => shift(@values) } @Fields);

    $Pcb -> element_begin(description => 'TH',
                          output_file =>
                             "tmp/" . &package_name($f{circuits}, $f{pin_rows}),
                          dim   => 'mils',
                          pin_one_square => 1);

    my $pin_num = 1;
    my $pins_per_row = $f{circuits} / 2;

    # lower left corner is pin one

    my $x = -$f{pin_spacing} * ($pins_per_row - 1) / 2;
    my $y =  $f{row_spacing} / 2;

    # These header connectors consist of two rows of pins.  With pin
    # one in the lower left corner we will place pins from left to
    # right until half the pins are placed. At the halfway point we
    # will shift to the top row and place pins from right to left.

    while ($pin_num <= $f{circuits}) {
        $Pcb -> element_add_pin(x => $x, y => $y,
                                thickness  => 66,
                                drill_hole => 46,
                                mask       => 10,
                                clearance  => 10,
                                pin_number => $pin_num);

        # If this is the last pin in the row then
        # update the y value otherwise update the x
        # value. If we are past the halfway point move
        # left (-) instead of right (+).

        $y *= -1;
        $x += $f{pin_spacing} if $y > 0;
```

```
73          $pin_num++;
74      }
75
76      $Pcb -> element_add_rectangle(width => $f{body_width},
77                                    length=> $f{body_length},
78                                    x => 0,
79                                    y => 0);
80
81
82      $Pcb -> element_set_text_xy(x => -$f{body_length}/2,
83                                  y => -$f{body_width}/2 - 20);
84
85
86      $Pcb -> element_output();
87  }
88
89  sub package_name ($$) {
90      my ($circuits, $rows) = @_;
91      sprintf("CON_HDR-254P-%iC-%iR-%iN__Molex_8624-Series",
92              $circuits/$rows,
93              $rows,
94              $circuits);
95  }
96
97  __DATA__
98
99  body_width = 200
100 pin_spacing = 100
101 row_spacing = 100
102 pin_diameter = 35
103 pin_rows = 2
104
105 # circuits   body_length   pin_row_length
106
107 4    190   100
108 6    290   200
109 8    390   300
110 10   490   400
111 12   590   500
112 14   690   600
113 16   790   700
114 18   890   800
115 20   990   900
116 22   1090   1000
117 24   1190  1100
118 26   1290  1200
119 28   1390  1300
120
121 30   1490   1400
122 32   1590   1500
123 34   1690   1600
124 36   1790   1700
125 38   1890   1800
126 40   1990   1900
127 42   2090   2000
128 44   2190   2100
129 46   2290   2200
130 48   2390   2300
131 50   2490   2400
132 52   2590   2500
133 54   2690   2600
134
135 56   2790   2700
136 58   2890   2800
137 60   2990   2900
138 62   3090   3000
139 64   3190   3100
140 66   3290   3200
141 68   3390   3300
142 70   3490   3400
143 72   3590   3500
144 74   3690   3600
145 76   3790   3700
```

```
78    3890    3800
80    3990    3900
```

Listing 5: Header Connector Creation Example 2

```
#!/usr/bin/perl

# Copyright (C) 2005 John C. Luciani Jr.

# This program may be distributed or modified under the terms of
# version 0.2 of the No-Fee Software License published by
# John C. Luciani Jr.

# Creates the PCB elements for Molex 8624 header connectors

use strict;
use warnings;

use Pcb_8;

my $Pcb = Pcb_8 -> new(debug => 0);

my @Fields = qw(circuits body_length pin_row_length);

my @Def; # definitions that are common to all components

while (<DATA>) {
    s/\#.*//; # Remove comments
    s/^\s*//; # Remove leading spaces
    s/\s*$//; # Revove trailing spaces
    next unless length; # Skip empty lines

    # Lines that contain an '=' are global definitions.

    push(@Def, $1, $2), next if /(\S+)\s*=\s*(\S.*)/;

    my @values = split /\s*\ \s*/;

    # hash for each footprint

    my %f = ( @Def,
              map { $_ => shift(@values) } @Fields);

    $Pcb -> element_begin(description => 'TH',
                          output_file =>
                            "tmp/" . &package_name($f{circuits}, $f{pin_rows}),
                          dim   => 'mils',
                          pin_one_square => 1);

    my $pin_num = 1;
    my $pins_per_row = $f{circuits} / 2;

    # lower left corner is pin one

    my $x0 = -$f{pin_spacing} * ($pins_per_row - 1) / 2;
    my $y0 =  $f{row_spacing} / 2;

    my $x = $x0;
    my $y = $y0;

    # These header connectors consist of two rows of pins.  With pin
    # one in the lower left corner we will place pins from left to
    # right until half the pins are placed. At the halfway point we
    # will shift to the top row and place pins from right to left.

    while ($pin_num <= $f{circuits}) {
        $Pcb -> element_add_pin(x => $x, y => $y,
                                thickness  => $f{pad_thickness},
                                drill_hole => $f{drill_hole},
                                mask       => 10,
                                clearance  => 10,
                                pin_number => $pin_num);

        # Header connectors usually have pins numbered from left to
        # right with odd numbers on the bottom and even numbers on the
        # top. Since this example program could be used for connectors
        # other than headers three pin-numbering options are provided.
```

```
73
74          # header - two rows of pins. numbers increase from left to right.
75          #          odd numbered pins on the bottom, even on the top.
76
77          # dip    - two rows of pins. starting in the lower left corner
78          #          numbers increase left to right along the bottom row
79          #          and right to left along the top row.
80
81          # power  - two rows of pins. numbers increase from left to right
82          #          starting on the bottom row and then continue left to right
83          #          along the top row.
84
85          if ($f{pin_numbering_scheme} eq 'header') {
86              $y *= -1;
87              $x += $f{pin_spacing} if $y > 0;
88          } elsif ($f{pin_numbering_scheme} eq 'dip') {
89              if ($pin_num == $pins_per_row) {
90                  $y -= $f{row_spacing};
91              } else {
92                  $x += $pin_num > $pins_per_row
93                      ? -$f{pin_spacing}
94                      : $f{pin_spacing};
95              }
96          } elsif ($f{pin_numbering_scheme} eq 'power') {
97              if ($pin_num == $pins_per_row) {
98                  $y -= $f{row_spacing};
99                  $x = $x0;
100             } else {
101                 $x += $f{pin_spacing}
102             }
103         } else {
104             die "unknown pin numbering scheme  $f{pin_numbering_scheme} ";
105         }
106         $pin_num++;
107     }
108
109     $Pcb -> element_add_rectangle(width => $f{body_width},
110                                   length=> $f{body_length},
111                                   x => 0,
112                                   y => 0);
113
114
115     $Pcb -> element_set_text_xy(x => -$f{body_length}/2,
116                                 y => -$f{body_width}/2 - 20);
117
118
119     $Pcb -> element_output();
120 }
121
122 sub package_name ($$) {
123     my ($circuits, $rows) = @_;
124     sprintf("CON_HDR-254P-%iC-%iR-%iN__Molex_8624-Series",
125             $circuits/$rows,
126             $rows,
127             $circuits);
128 }
129
130 __DATA__
131
132 pad_thickness = 66
133 drill_hole = 46
134 pin_numbering_scheme = header
135 body_width = 200
136 pin_spacing = 100
137 row_spacing = 100
138 pin_diameter = 35
139 pin_rows = 2
140
141 # circuits   body_length   pin_row_length
142
143 4    190   100
144 6    290   200
145 8    390   300
```

```
10    490    400
12    590    500
14    690    600
16    790    700
18    890    800
20    990    900
22    1090    1000
24    1190   1100
26    1290   1200
28    1390   1300

30    1490    1400
32    1590    1500
34    1690    1600
36    1790    1700
38    1890    1800
40    1990    1900
42    2090    2000
44    2190    2100
46    2290    2200
48    2390    2300
50    2490    2400
52    2590    2500
54    2690    2600

56    2790    2700
58    2890    2800
60    2990    2900
62    3090    3000
64    3190    3100
66    3290    3200
68    3390    3300
70    3490    3400
72    3590    3500
74    3690    3600
76    3790    3700
78    3890    3800
80    3990    3900
```

Listing 6: TQFN Element Creation Example

```
#!/usr/bin/perl

# Copyright (C) 2005 John C. Luciani Jr.

# This program may be distributed or modified under the terms of
# version 0.2 of the No-Fee Software License published by
# John C. Luciani Jr.

# Creates Maxim TQFN style packages.

# Data is from the Maxim 21-0140 Rev G and Maxim 21-10159 Rev A
# specifications.

# The TQFN (thin quad flat no-lead) packages have solder terminations
# on four sides and a thermal pad in the center. The two denser
# packages (T3255-2 and T4055-1) require smaller pads on the corner
# terminations.

use strict;
use warnings;
use Carp;

use Pcb_8; # routines to create PCB elements (packages)

my $Pcb = Pcb_8 -> new(debug => 0);

# The specifications to generate these symbols is in the __DATA__
# section of this file. Each line can be either blank, contain a global
# definition or contain a package data record.

# Global definitions are saved in @Def.
# The field names for the package data record are in @Fields.

my @Def; # Global definitions saved as key-value pairs.
my @Fields = qw(package_code
                pin_count
                pad_spacing
                pad_width
                pad_length
                corner_pad_length
                thermal_pad_width
                thermal_pad_length);

# Read the __DATA__ section and output a PCB footprint everytime a
# package data record is read.

while (<DATA>) {
    last if /^__END__$/;
    s/\#.*//; # Remove comments
    s/^\s*//; # Remove leading spaces
    s/\s*$//; # Revove trailing spaces
    next unless length; # Skip empty lines

    # Lines that contain an '=' are global definitions.  The key (lhs)
    # and value (rhs) are pushed onto @Def.

    push(@Def, $1, $2), next if /(\S+)\s*=\s*(\S.*)/;

    # A line with non-zero length that is not a global definition is a
    # package data record. We split the package record and create a
    # hash %p that contains key-value pairs for all of the global
    # definitions and the current record.

    my @values = split /\s*\\s*/;
    my %p = ( @Def,
              map { $_ => shift(@values) } @Fields);

    # Create a simple id using the package name, package code and pin
    # count and then start a new element.

    $p{id} = join('-', map { $p{$_} } qw(package_name package_code pin_count));
    $Pcb -> element_begin(description => $p{id},
```

```
                          output_file => "tmp/$p{id}",
                          dim    => 'mm');
    print "$p{id}\n";

    # Create a few convenient specifications from data in the package
    # data record hash. The convenetions for these packages is part
    # centroid at (0,0) and pin one is in the lower left corner.

    # Corner pads on some of the parts are shorter. This condition is
    # handled by creating a new pad length and some pad center
    # offsets.

    $p{num_pads_per_side} = $p{pin_count} / 4; # leads on four sides
    $p{corner_pad_length}  = $p{pad_length} if $p{corner_pad_length} eq '';

    my $row_center = ($p{body_width_max} - $p{pad_length}) / 2;
    my $row_end    = ($p{num_pads_per_side} - 1) * $p{pad_spacing} / 2;
    my $corner_offset = ($p{pad_length} - $p{corner_pad_length}) / 2;

    # @xy contains the staring locations for a row of pads.
    # @inc contains increment values for x and y and offsets for the
    # the corner pads. for each row of pads either x or y is incremented.

    my @xy = (x => -$row_center, y => -$row_end,
              x => -$row_end,    y =>  $row_center,
              x =>  $row_center, y =>  $row_end,
              x =>  $row_end,    y => -$row_center);

    my @inc= (yinc =>  $p{pad_spacing}, xoffset => -$corner_offset,
              xinc =>  $p{pad_spacing}, yoffset =>  $corner_offset,
              yinc => -$p{pad_spacing}, xoffset =>  $corner_offset,
              xinc => -$p{pad_spacing}, yoffset => -$corner_offset);

    # create the rows of pads

    &set_pin_num(1);

    while (@xy) {
        my %xy = (splice(@inc, 0, 4),
                  map { $_ => $p{$_} } qw(pad_spacing pad_width pad_length));

        &draw_row_of_pads(splice(@xy, 0, 4),
                          %xy,
                          pad_length => $p{corner_pad_length},
                          num_pads => 1);

        # no offsets for pads that aren't on the corners

        &draw_row_of_pads(%xy,
                          xoffset => 0,
                          yoffset => 0,
                          num_pads => $p{num_pads_per_side} - 2);

        &draw_row_of_pads(%xy,
                          pad_length => $p{corner_pad_length},
                          num_pads => 1);
    }

    # Add the thermal pad

    $Pcb -> element_add_pad_rectangle(x => 0,
                                      y => 0,
                                      length => $p{thermal_pad_length},
                                      width  => $p{thermal_pad_width},
                                      name => '',
                                      pin_number => $p{pin_count} + 1);

    # add the pin one dot

    my $dot_pos = $row_center + 0.254; #$p{pad_length} / 2;

    $Pcb -> element_add_arc(x => -$dot_pos,
                            y => -$dot_pos,
```

```
                                width => $p{pad_width} / 2,
                                height=> $p{pad_width} / 2,
                                start_angle => 0,
                                delta_angle => 360,
                                thickness => 0.254); # 10 mil lines

    # draw a silksreen rectangle around the package body.

    $Pcb -> element_add_rectangle(x => 0,
                                  y => 0,
                                  width => $p{body_width_max} + 1,
                                  length=> $p{body_length_max} + 1);

    # Set the position of the reference designator to the upper left corner

    $Pcb -> element_set_text_xy(x => -$p{body_length_max} / 2 - 1,
                                y => -$p{body_width_max}  / 2 - 1);

    # Set the centroid mark and output the element

    $Pcb -> element_output;

}

# $v{x}       current x location
# $v{y}       current y location
# $v{pin_num} current pin number

my %v;   # values for draw_row_of_pads

sub set_pin_num ($) {
    $v{pin_num} = shift;
}

sub draw_row_of_pads {
    my %p = (xoffset => 0,
             yoffset => 0,
             @_);

    foreach (qw(pin_num x y)) {
        $v{$_} = $p{$_} if defined $p{$_};
    }

    # swap pad length and width for horizontal rows

    ($p{pad_width}, $p{pad_length}) = ($p{pad_length}, $p{pad_width})
        if defined $p{xinc};

    foreach (1..$p{num_pads}) {
        $Pcb -> element_add_pad_rectangle(x      => $v{x} + $p{xoffset},
                                          y      => $v{y} + $p{yoffset},
                                          width  => $p{pad_width},
                                          length => $p{pad_length},
                                          name   => '',
                                          pin_number => $v{pin_num}++);
        $v{x} += $p{xinc} if defined $p{xinc};
        $v{y} += $p{yinc} if defined $p{yinc};
    }
}




1;



__DATA__
body_width_min  = 4.9  # E
body_width      = 5.0  # E
body_width_max  = 5.1  # E
body_length_min = 4.9  # D
body_length     = 5.0  # D
```

```
body_length_max = 5.1  # D

component_type     = ic
package_name       = TQFN-Maxim-5x5

# Final pad_length = pad_lenth + body_width_max - body_width_min

# T1655-1   e (nom)  b, L, E2, D2 (max)
# T2055-2   e (nom)  b (max) L, E2, D2 (nom)
# T2055-5   e (nom)  b (max) L (min) E2, D2 (nom)

# T2855-1   e (nom)  b (max) L (min) E2, D2 (nom)
# T2855-2   e (nom)  b (max) L (min) E2, D2 (max)

# T3255-2   e (nom)  b (max) L (max) E2, D2 (max)
# T4055-1   e (nom)  b (min) L (nom) E2, D2 (min)


#                 e      b      L     L1      E2     D2
 T1655-1    16   0.8    0.35   0.5           3.2    3.2
 T2055-2    20   0.65   0.35   0.55          3.10   3.10
 T2055-5    20   0.65   0.35   0.45          3.25   3.25

 T2855-1    28   0.50   0.30   0.45          3.25   3.25
 T2855-2    28   0.50   0.30   0.45          2.8    2.8


 T3255-2    32   0.50   0.30   0.5    0.25   3.2    3.2
 T4055-1    40   0.40   0.2    0.5    0.25   3.2    3.2

# Style (adapted from the Perl Cookbook, First Edition, Recipe 12.4)

# 1. Names of functions and local variables are all lowercase.
# 2. The program's persistent variables (either file lexicals
#    or package globals) are capitalized.
# 3. Identifiers with multiple words have each of these
#    separated by an underscore to make it easier to read.
# 4. Constants are all uppercase.
# 5. If the arrow operator (->) is followed by either a
#    method name or a variable containing a method name then
#    there is a space before and after the operator.
```