hello-world circuits:

serial output:
- hello.serial.45.cad
- hello.serial.45.asm
- hello.serial.45.hex

serial I/O:
- hello.serial.io.45.cad
- hello.serial.io.45.asm

light:
- hello.light.45.cad
- hello.light.45.asm
- hello.light.45.py

temperature:
- hello.temp.45.cad
- hello.temp.45.asm
- hello.temp.45.py

step response (resistance, capacitance, inductance, position, proximity):
- hello.step.45.cad
- hello.step.45.asm
- hello.step.45.py

microphone:
- hello.mic.45.cad
- hello.mic.45.asm
- hello.mic.45.py
- hello.mic.44.cad

RGB LED:
- hello.RGB.45.cad
- hello.RGB.45.asm

speaker:
- hello.speaker.45.cad
- hello.speaker.45.pwm.asm
- hello.speaker.45.wave.asm

DC motor:
- hello.H-bridge.44.cad
- hello.H-bridge.44.asm

video:
- hello.video.44.cad
- hello.video.44.asm

LCD:
- hello.LCD.44.cad

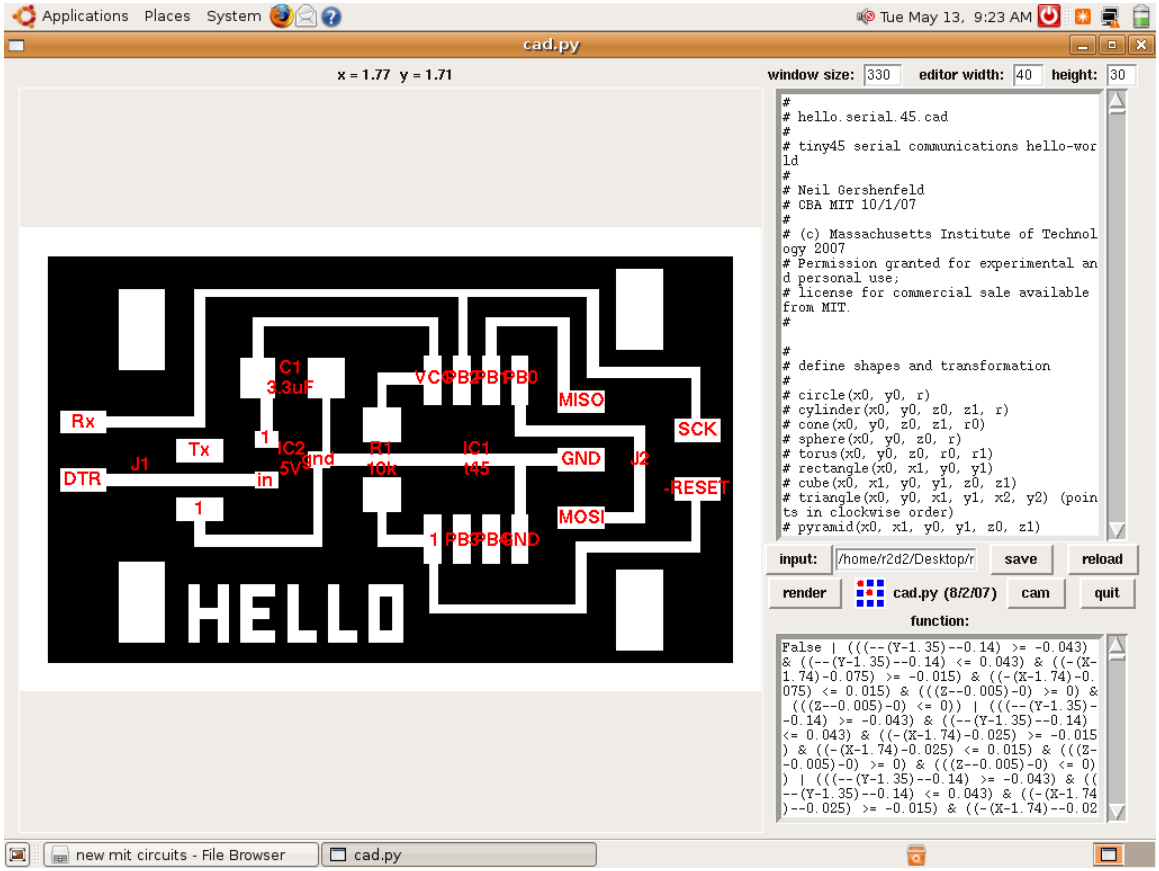
stepper motor:
- hello.stepper.44.cad

LED array:
- hello.array.44.cad

Note: all circuits need programming cable and connectors.

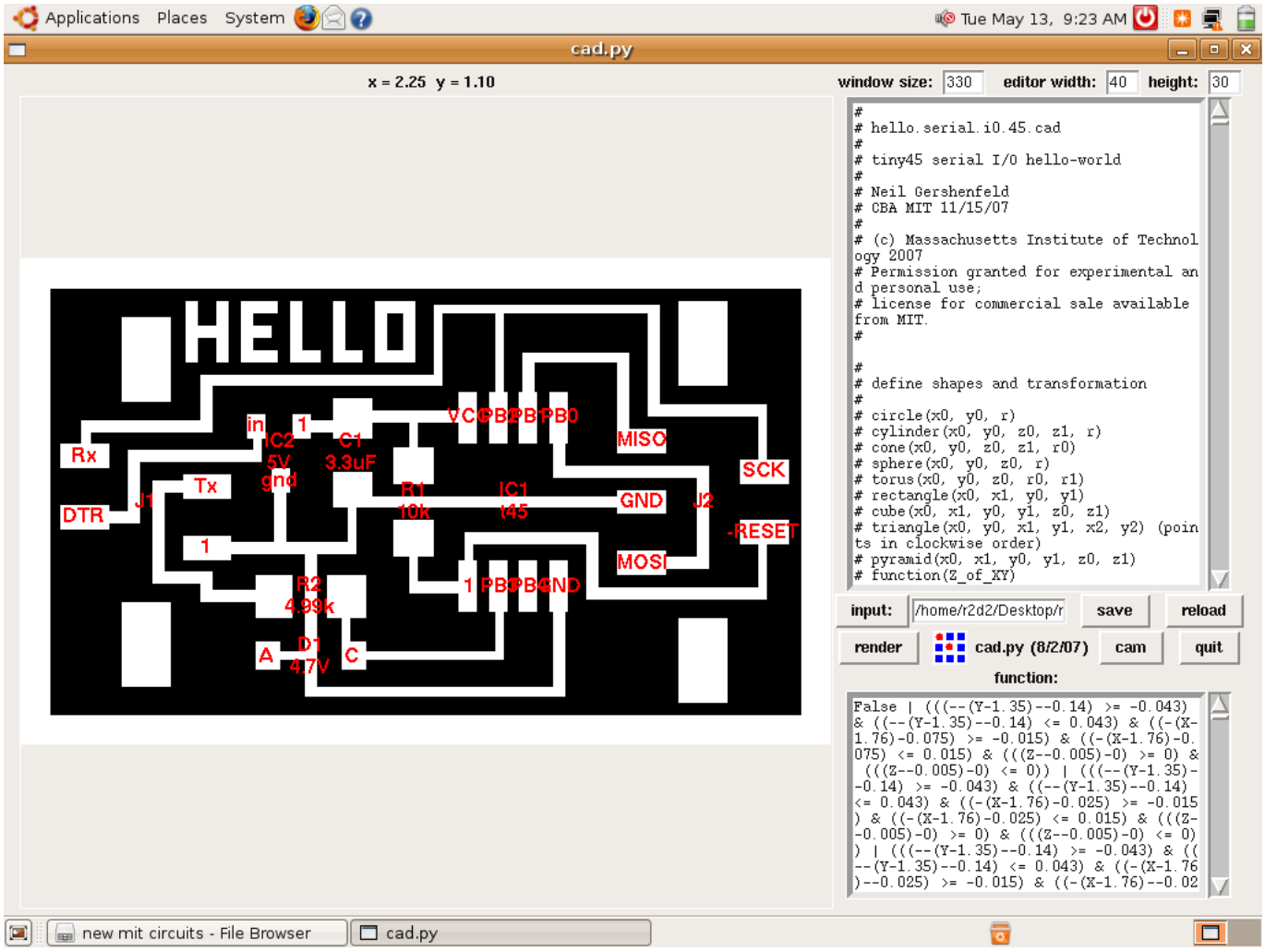
serial output:

- hello.serial.45.cad
- hello.serial.45.asm
- hello.serial.45.hex




```
IC1 = ATtiny45_SOIC('IC1\nt45')
IC2 = regulator_SOT23('IC2\n5V')

R1 = R_1206('R1\n10k')

C1 = C_1206('C1\n3.3uF')

J1 = MTA_serial('J1')
J2 = MTA_ICP('J2')

Serial cable to computer
```

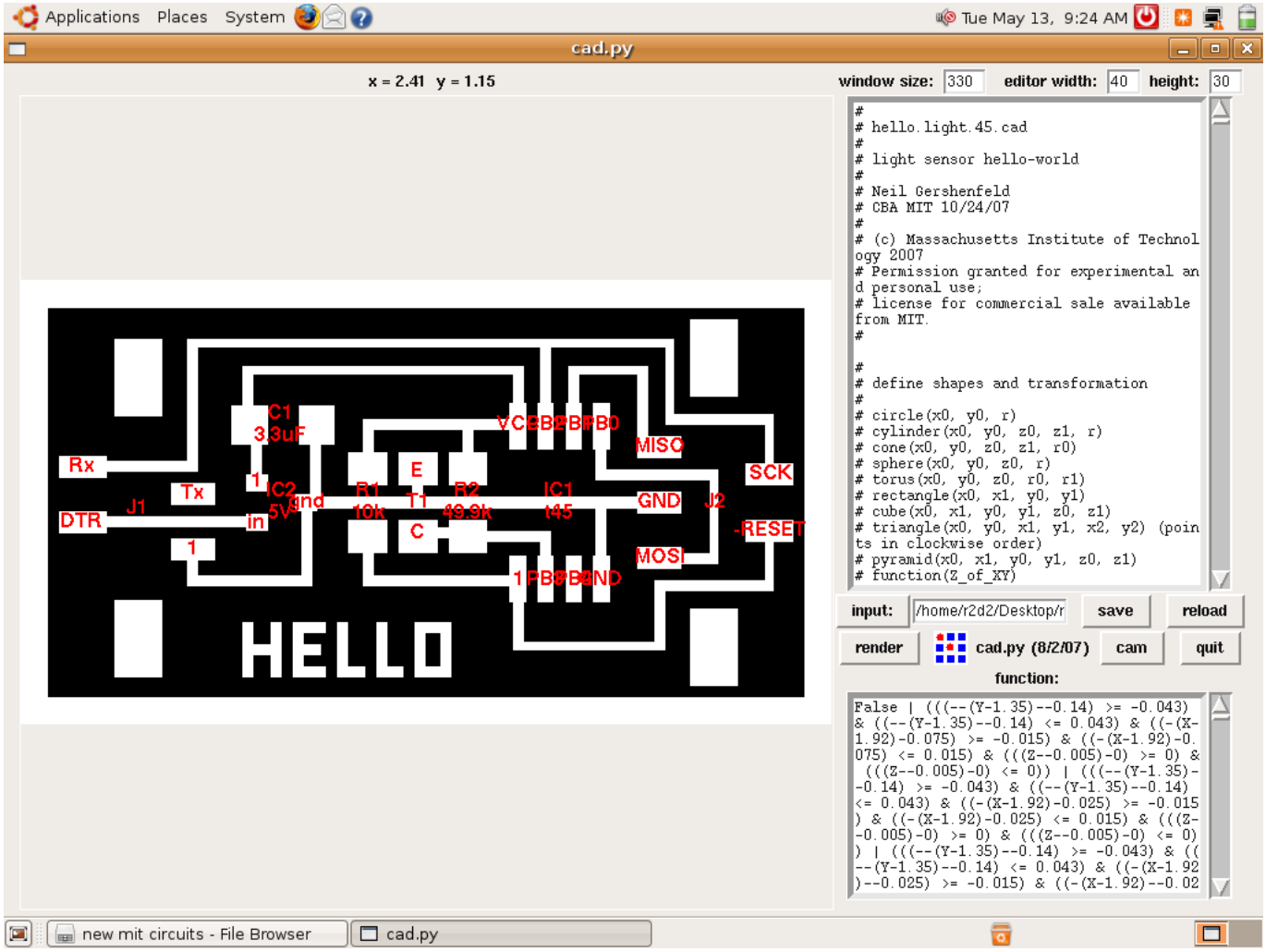
serial I/O:
   hello.serial.io.45.cad
   hello.serial.io.45.asm




```
IC1 = ATtiny45_SOIC('IC1\nt45')
IC2 = regulator_SOT23('IC2\n5V')

R1 = R_1206('R1\n10k')
R2 = R_1206('R2\n4.99k')

C1 = C_1206('C1\n3.3uF')

J1 = MTA_serial('J1')
J2 = MTA_ICP('J2')

D1 = D_SOD_123('D1\n4.7V')
```

Serial cable to computer

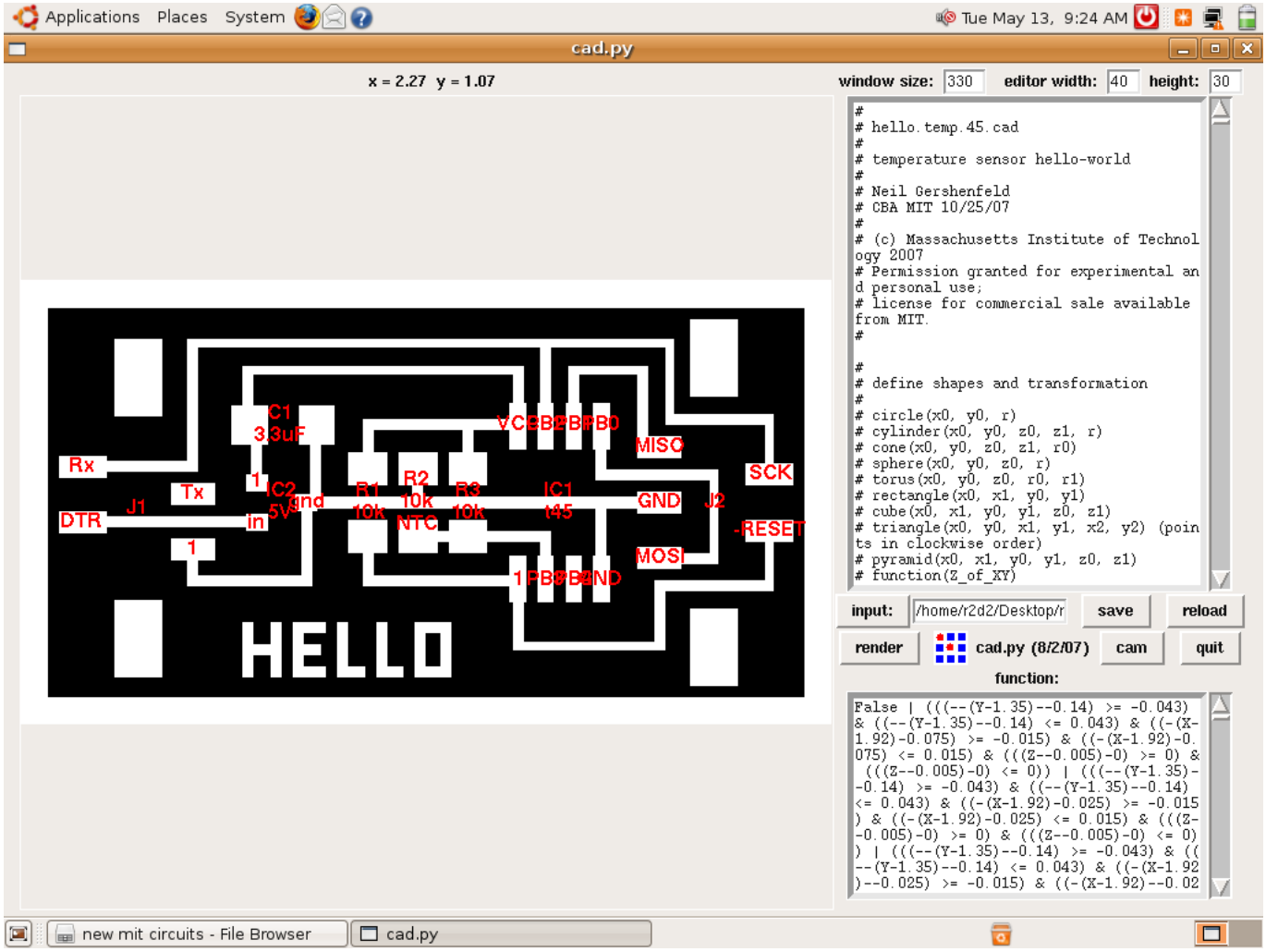
light:
   hello.light.45.cad
   hello.light.45.asm
   hello.light.45.py




```
IC1 = ATtiny45_SOIC('IC1\nt45')

R1 = R_1206('R1\n10k')
R2 = R_1206('R2\n49.9k')

C1 = C_1206('C1\n3.3uF')

J1 = MTA_serial('J1')
J2 = MTA_ICP('J2')

T1 = phototransistor_1206('T1') # OP522
```

Serial cable to computer

temperature:
   hello.temp.45.cad
   hello.temp.45.asm
   hello.temp.45.py




```
IC1 = ATtiny45_SOIC('IC1\nt45')
IC2 = regulator_SOT23('IC2\n5V')

C1 = C_1206('C1\n3.3uF')

J1 = MTA_serial('J1')
J2 = MTA_ICP('J2')

R1 = R_1206('R1\n10k')
R2 = R_1206('R2\n10k\nNTC')
R3 = R_1206('R3\n10k')
```

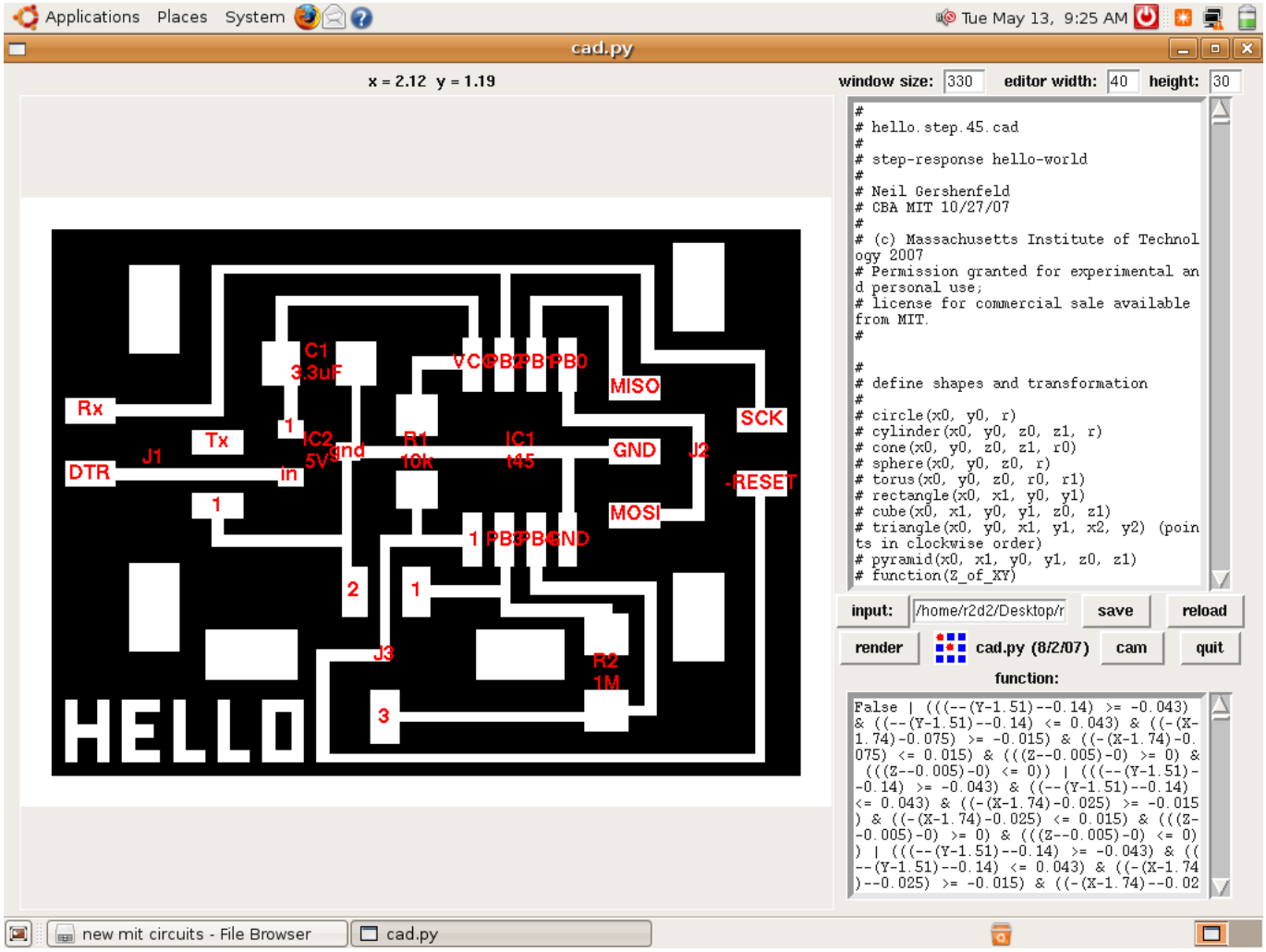
Serial cable to computer
Temperature sensor?

step response (resistance, capacitance, inductance, position, proximity):

hello.step.45.cad

hello.step.45.asm

hello.step.45.py




```
IC1 = ATtiny45_SOIC('IC1\nt45')
IC2 = regulator_SOT23('IC2\n5V')

R1 = R_1206('R1\n10k')
R2 = R_1206('R2\n1M')

C1 = C_1206('C1\n3.3uF')

J1 = MTA_serial('J1')
J2 = MTA_ICP('J2')
J3 = MTA_3('J3')
```

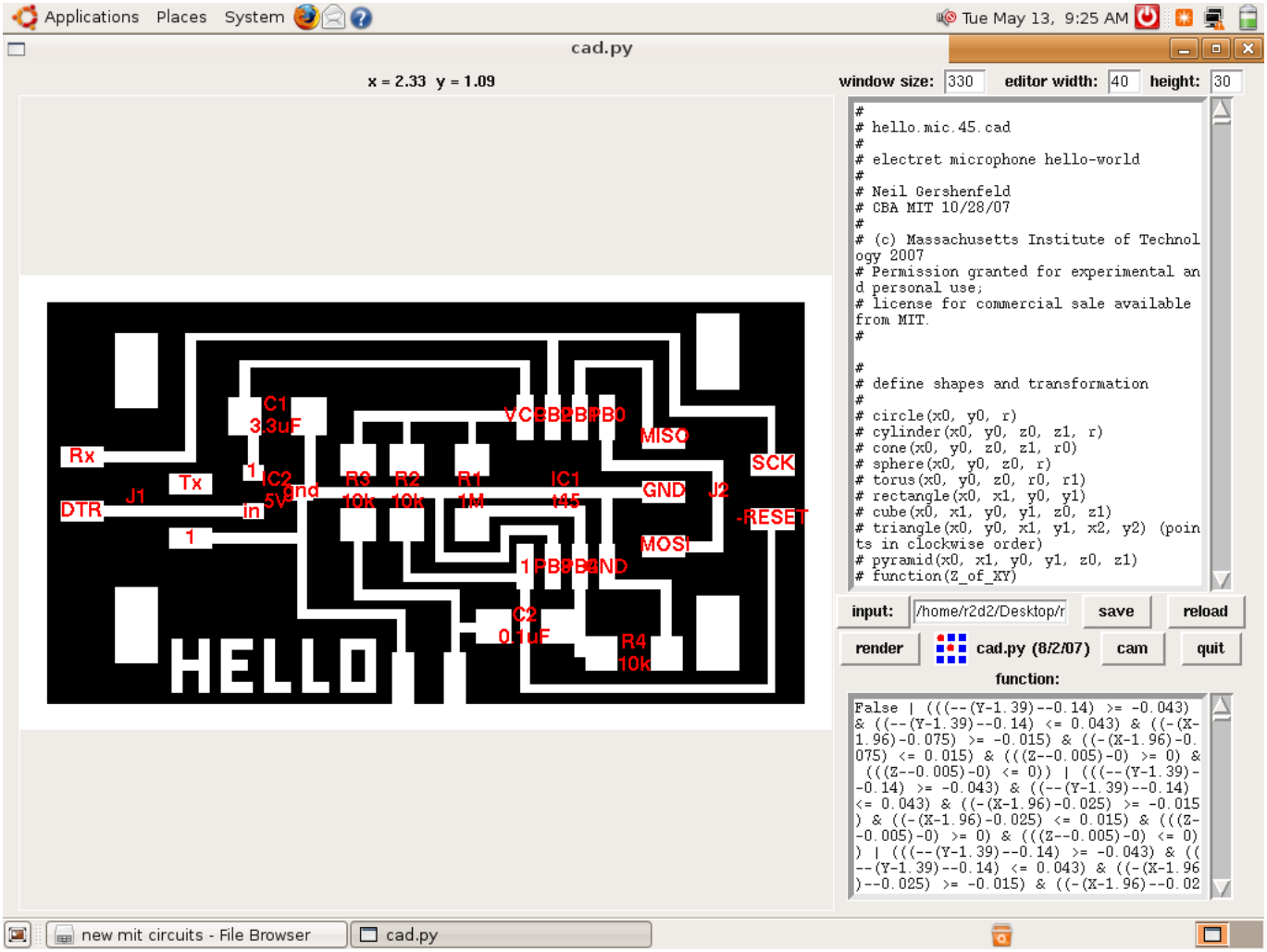
Serial cable to computer

resistance, capacitance, inductance, position, proximity sensor elements

Connector for sensor element

microphone:

- hello.mic.45.cad
- hello.mic.45.asm
- hello.mic.45.py
- hello.mic.44.cad (not shown but CAD file text is included)



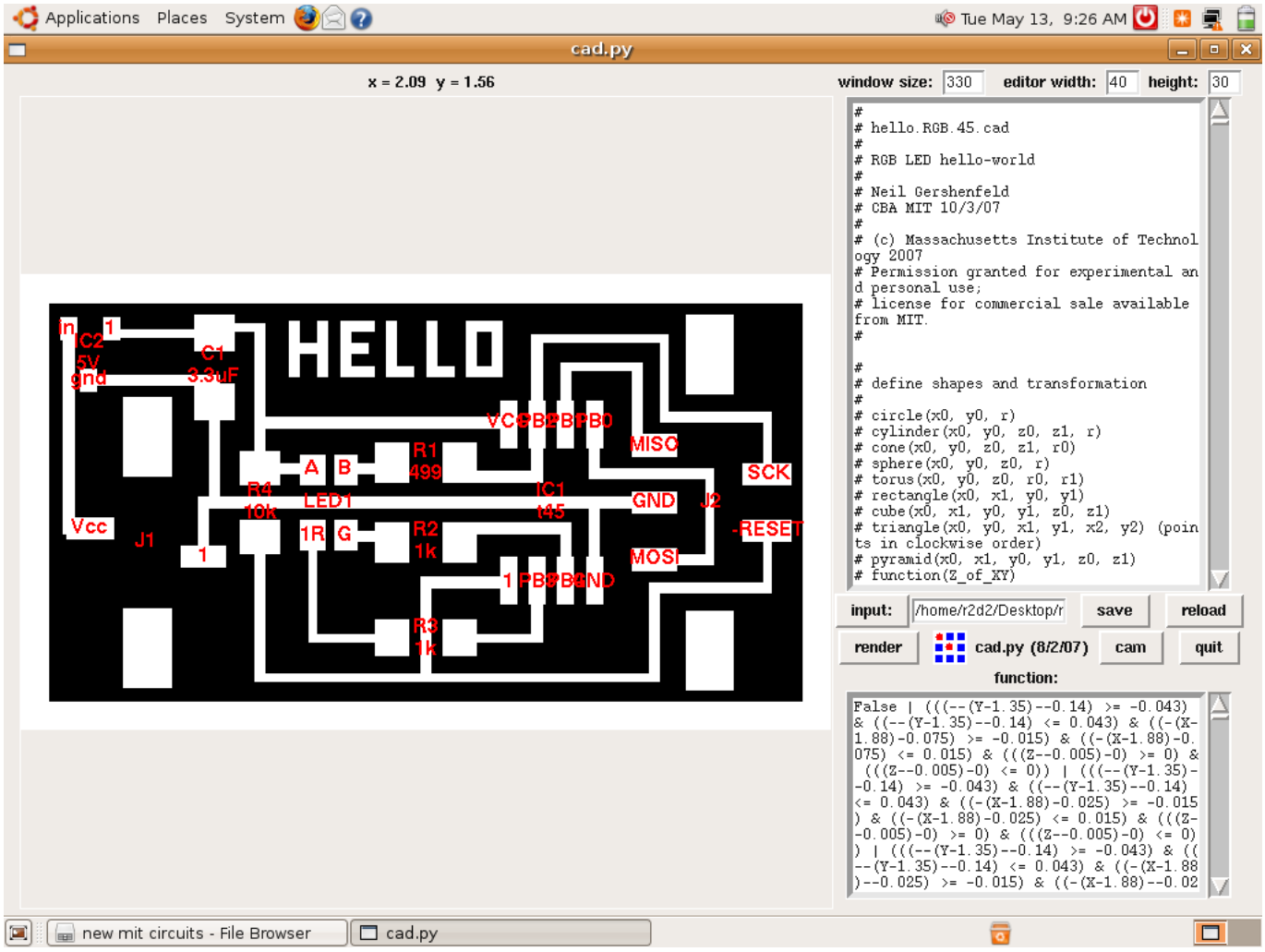

```
IC1 = ATtiny45_SOIC('IC1\nt45')
IC2 = regulator_SOT23('IC2\n5V')

R1 = R_1206('R1\n1M')
R2 = R_1206('R2\n10k')
R3 = R_1206('R3\n10k')
R4 = R_1206('R4\n10k')

C1 = C_1206('C1\n3.3uF')
C2 = C_1206('C2\n0.1uF')

J1 = MTA_serial('J1')
J2 = MTA_ICP('J2')
```

423-1043 electret (microphone)
Serial cable to computer

RGB LED:

hello.RGB.45.cad

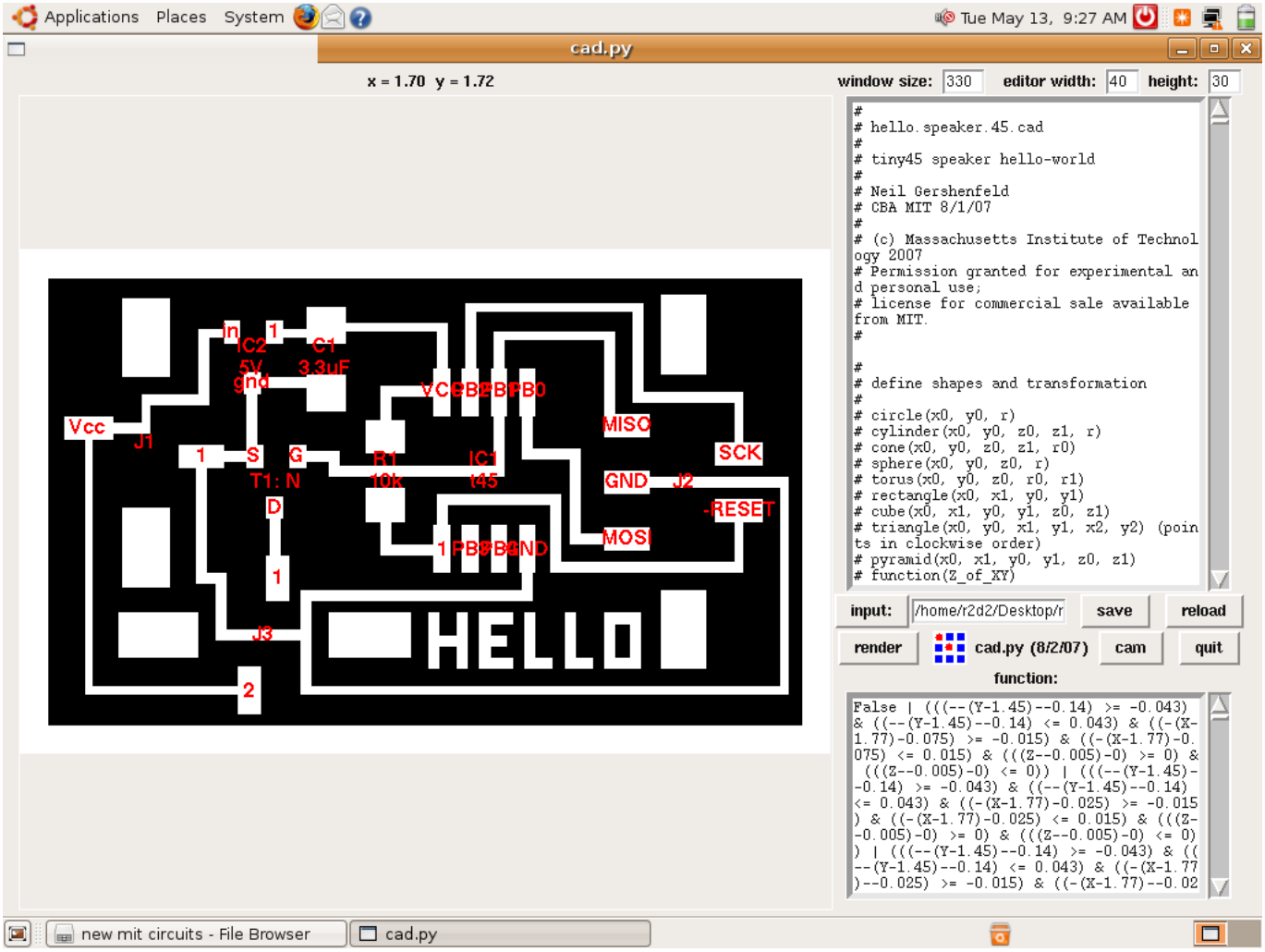
hello.RGB.45.asm




```
IC1 = ATtiny45_SOIC('IC1\nt45')
IC2 = regulator_SOT23('IC2\n5V')

R1 = R_1206('R1\n499')
R2 = R_1206('R2\n1k')
R3 = R_1206('R3\n1k')
R4 = R_1206('R4\n10k')

C1 = C_1206('C1\n3.3uF')

LED1 = LED_RGB('LED1')

J1 = MTA_power('J1')
J2 = MTA_ICP('J2')
```

Battery (power), cable, and connector

speaker:
   hello.speaker.45.cad
   hello.speaker.45.pwm.asm
   hello.speaker.45.wave.asm



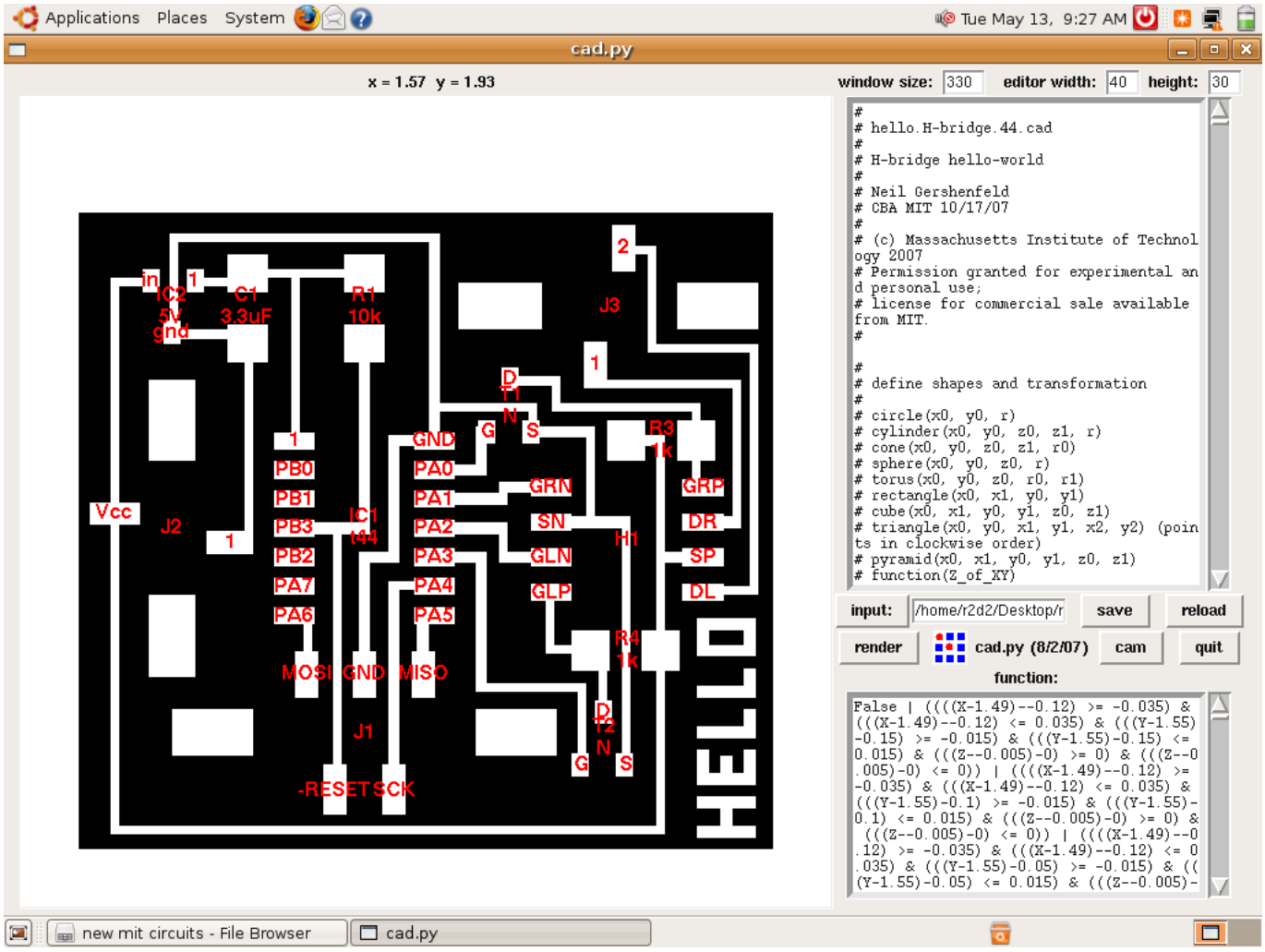

```
IC1 = ATtiny45_SOIC('IC1\nt45')
IC2 = regulator_SOT23('IC2\n5V')

R1 = R_1206('R1\n10k')

C1 = C_1206('C1\n3.3uF')

J1 = MTA_power('J1')
J2 = MTA_ICP('J2')
J3 = MTA_2('J3')

T1 = NMOSFET_SOT23('T1: N')
```

Speaker and connector
Battery (power), cable, and connector

DC motor:

hello.H-bridge.44.cad

hello.H-bridge.44.asm



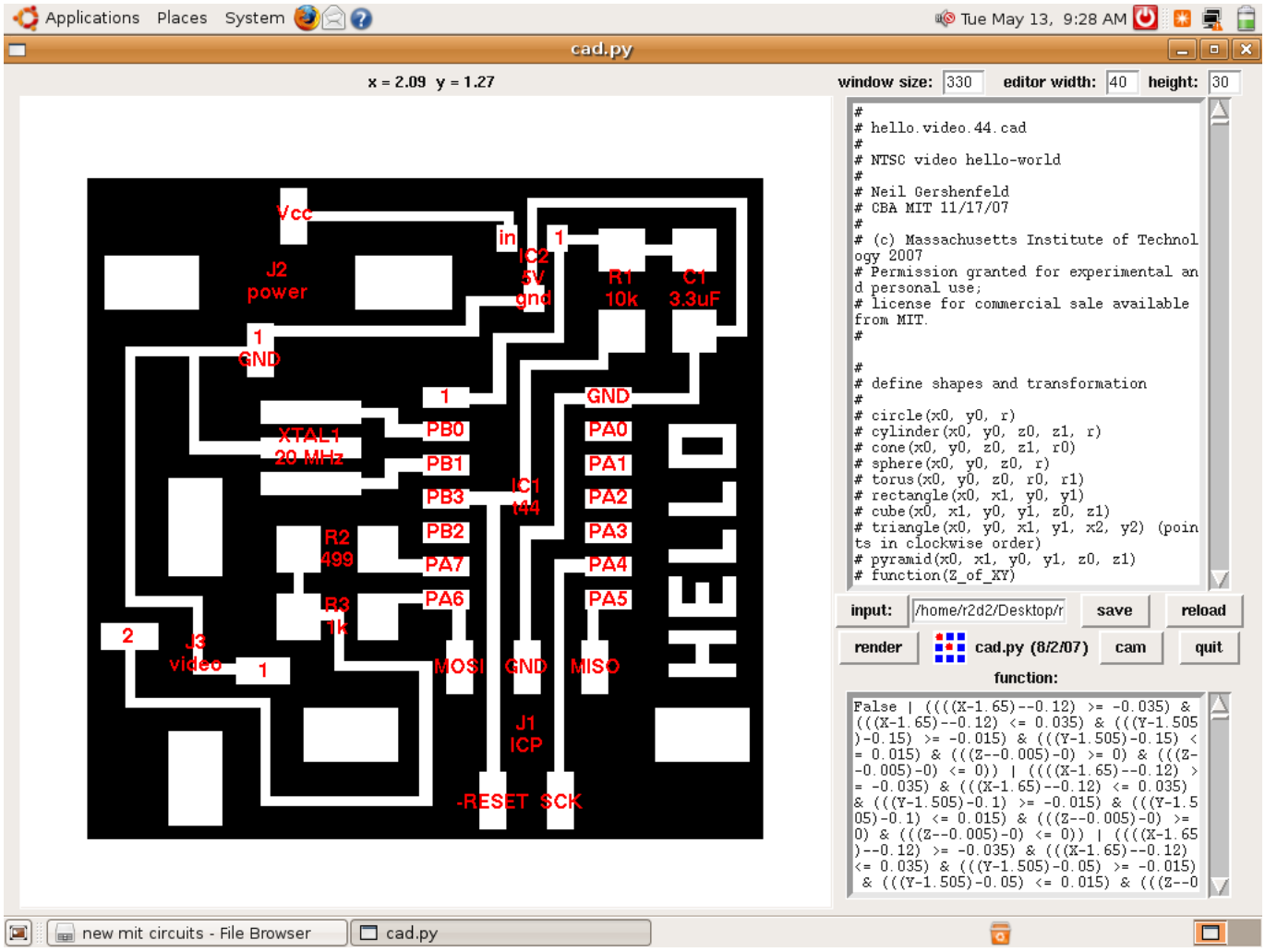

```
IC1 = ATtiny44_SOICN('IC1\nt44')
IC2 = regulator_SOT23('IC2\n5V')

J1 = MTA_ICP('J1')
J2 = MTA_power('J2')
J3 = MTA_2('J3')

C1 = C_1206('C1\n3.3uF')

R1 = R_1206('R1\n10k')
R3 = R_1206('R3\n1k')

R2 = not present

R4 = R_1206('R4\n1k')

H1 = H_bridge_SM8('H1')

T1 = NMOSFET_SOT23('T1\nN')
T2 = NMOSFET_SOT23('T2\nN')
```

Motor to control

Connector for motor

Battery (power), cable, and connector

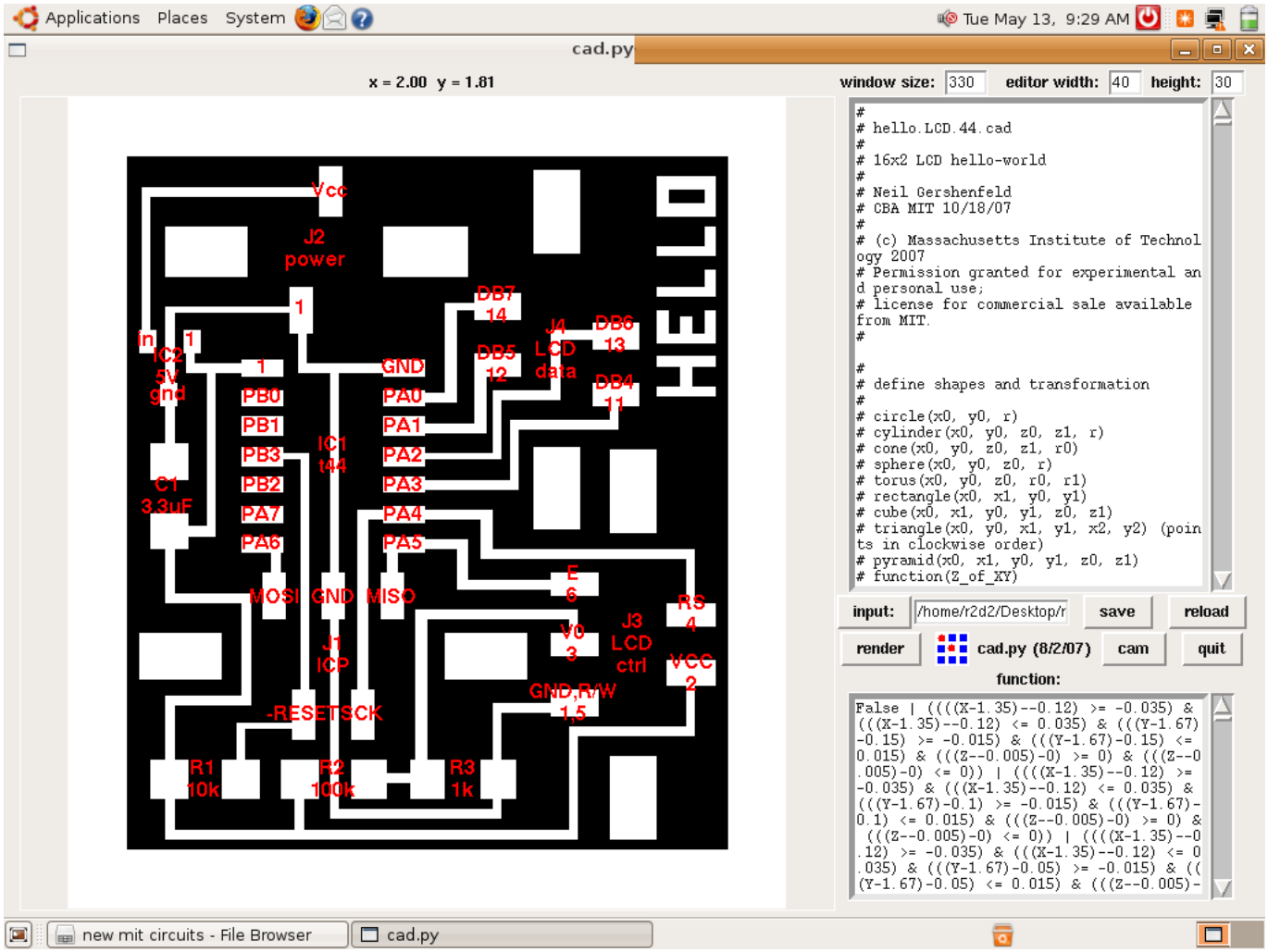
video:
   hello.video.44.cad
   hello.video.44.asm




```
IC1 = ATtiny44_SOICN('IC1\nt44')
IC2 = regulator_SOT23('IC2\n5V')

XTAL1 = XTAL('XTAL1\n20 MHz')

J1 = MTA_ICP('J1\nICP')
J2 = MTA_power('J2\npower')
J3 = MTA_2('J3\nvideo')

R1 = R_1206('R1\n10k')
R2 = R_1206('R2\n499')
R3 = R_1206('R3\n1k')

C1 = C_1206('C1\n3.3uF')

Video display, connector, and cable
Battery (power), cable, and connector
```

LCD:

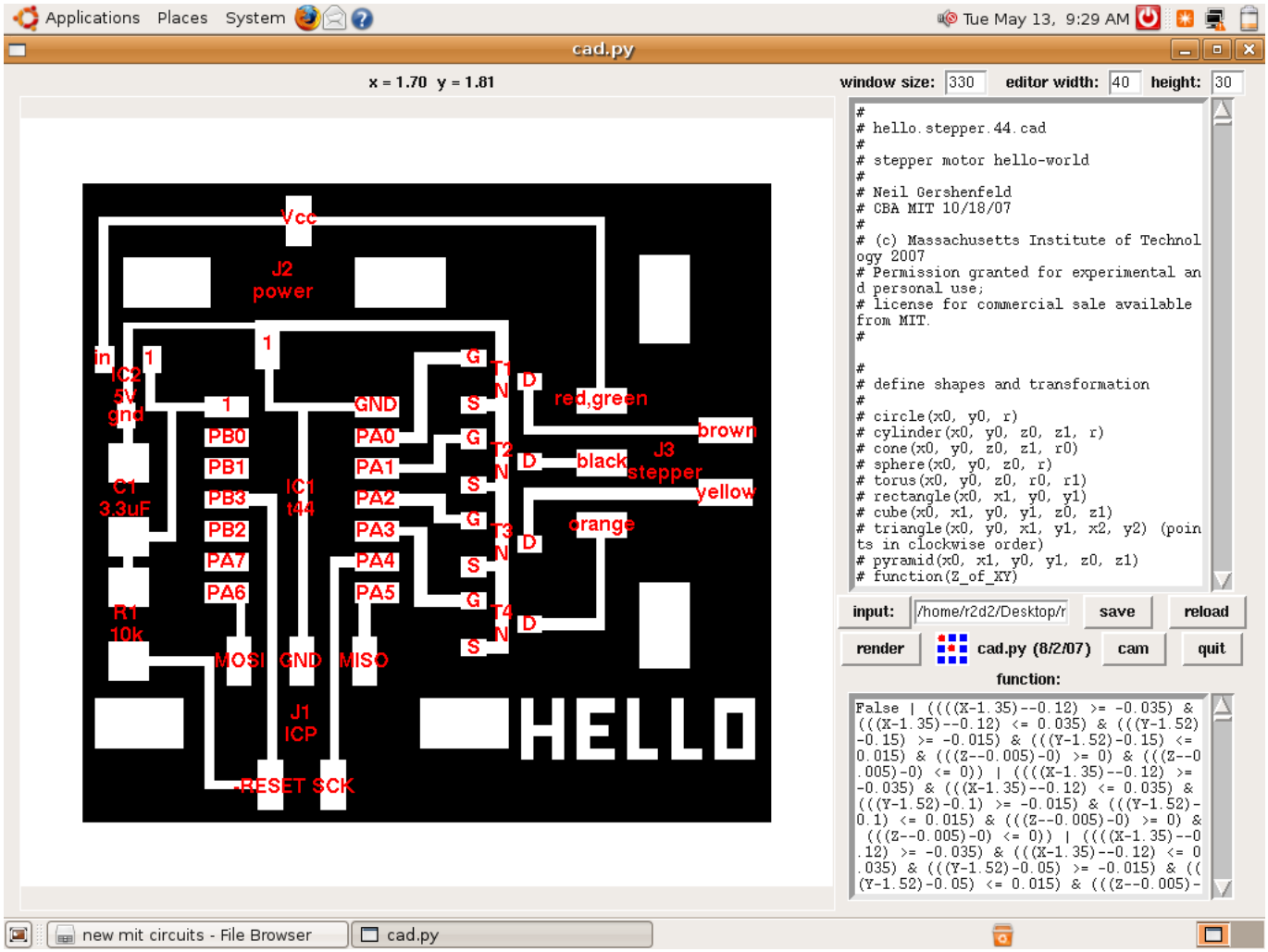
hello.LCD.44.cad




```
IC1 = ATtiny44_SOICN('IC1\nt44')
IC2 = regulator_SOT23('IC2\n5V')

J1 = MTA_ICP('J1\nICP')
J2 = MTA_power('J2\npower')
J3 = MTA_LCD_ctrl('J3\nLCD\nctrl')
J4 = MTA_LCD_data('J4\nLCD\ndata')

C1 = C_1206('C1\n3.3uF')

R1 = R_1206('R1\n10k')
R2 = R_1206('R2\n100k')
R3 = R_1206('R3\n1k')
```

LCD Display, cables and connectors
Battery (power), cable, and connector

stepper motor:
   hello.stepper.44.cad

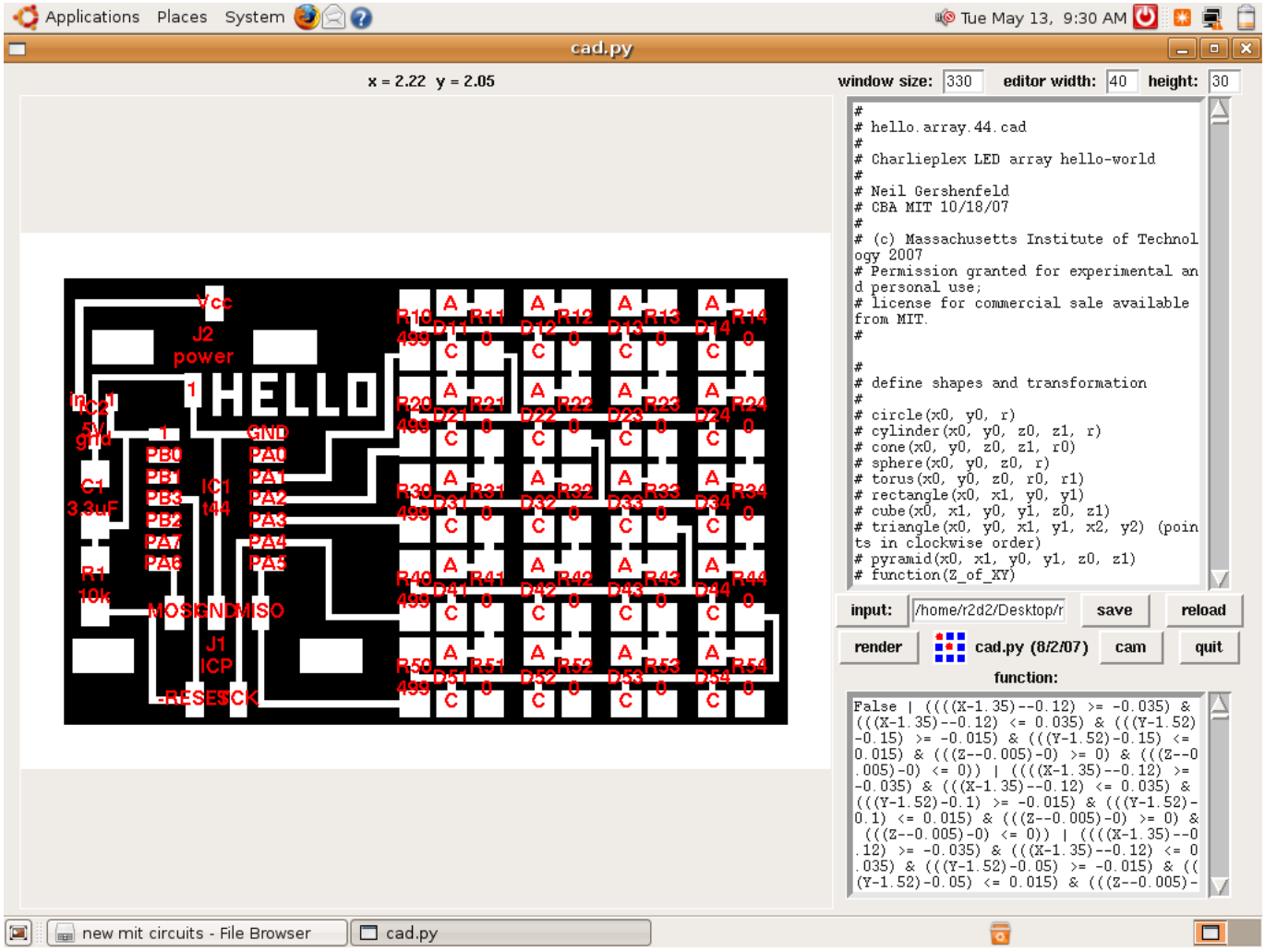



```
IC1 = ATtiny44_SOICN('IC1\nt44')
IC2 = regulator_SOT23('IC2\n5V')

J1 = MTA_ICP('J1\nICP')
J2 = MTA_power('J2\npower')
J3 = MTA_stepper('J3\nstepper')

C1 = C_1206('C1\n3.3uF')

R1 = R_1206('R1\n10k')

T1 = NMOSFET_SOT23('T1\nN')
T2 = NMOSFET_SOT23('T2\nN')
T3 = NMOSFET_SOT23('T3\nN')
T4 = NMOSFET_SOT23('T4\nN')
```

Stepper motor and connector
Battery (power), cable, and connector

LED array:

hello.array.44.cad




```
IC1 = ATtiny44_SOICN('IC1\nt44')
IC2 = regulator_SOT23('IC2\n5V'

J1 = MTA_ICP('J1\nICP')
J2 = MTA_power('J2\npower')

C1 = C_1206('C1\n3.3uF')

R1 = R_1206('R1\n10k')

R10 = R_1206('R10\n499')
R11 = R_1206('R11\n0')
R12 = R_1206('R12\n0')
R13 = R_1206('R13\n0')
R14 = R_1206('R14\n0')

R20 = R_1206('R20\n499')
R21 = R_1206('R21\n0')
R22 = R_1206('R22\n0')
R23 = R_1206('R23\n0')
R24 = R_1206('R24\n0')

R30 = R_1206('R30\n499')
R31 = R_1206('R31\n0')
R32 = R_1206('R32\n0')
R33 = R_1206('R33\n0')
R34 = R_1206('R34\n0')

R40 = R_1206('R40\n499')
R41 = R_1206('R41\n0')
R42 = R_1206('R42\n0')
R43 = R_1206('R43\n0')
R44 = R_1206('R44\n0')
```

```
R50 = R_1206('R50\n499')
R51 = R_1206('R51\n0')
R52 = R_1206('R52\n0')
R53 = R_1206('R53\n0')
R54 = R_1206('R54\n0')


D11 = LED_1206('D11')
D12 = LED_1206('D12')
D13 = LED_1206('D13')
D14 = LED_1206('D14')

D21 = LED_1206('D21')
D22 = LED_1206('D22')
D23 = LED_1206('D23')
D24 = LED_1206('D24')

D31 = LED_1206('D31')
D32 = LED_1206('D32')
D33 = LED_1206('D33')
D34 = LED_1206('D34')

D41 = LED_1206('D41')
D42 = LED_1206('D42')
D43 = LED_1206('D43')
D44 = LED_1206('D44')

D51 = LED_1206('D51')
D52 = LED_1206('D52')
D53 = LED_1206('D53')
D54 = LED_1206('D54')

Battery (power), cable, and connector
```

hello-world circuits:

serial I/O:
   hello.serial.io.45.cad
   hello.serial.io.45.asm
light:
   hello.light.45.cad
   hello.light.45.asm
   hello.light.45.py
temperature:
   hello.temp.45.cad
   hello.temp.45.asm
   hello.temp.45.py
step response (resistance, capacitance, inductance, position, proximity):
   hello.step.45.cad
   hello.step.45.asm
   hello.step.45.py
microphone:
   hello.mic.45.cad
   hello.mic.45.asm
   hello.mic.45.py
   hello.mic.44.cad
RGB LED:
   hello.RGB.45.cad
   hello.RGB.45.asm
speaker:
   hello.speaker.45.cad
   hello.speaker.45.pwm.asm
   hello.speaker.45.wave.asm
DC motor:
   hello.H-bridge.44.cad
   hello.H-bridge.44.asm
video:
   hello.video.44.cad
   hello.video.44.asm
LCD:
   hello.LCD.44.cad
stepper motor:
   hello.stepper.44.cad
LED array:
   hello.array.44.cad

```
#
# hello.serial.45.cad
#
# tiny45 serial communications hello-world
#
# Neil Gershenfeld
# CBA MIT 10/1/07
#
# (c) Massachusetts Institute of Technology 2007
# Permission granted for experimental and personal use;
# license for commercial sale available from MIT.
#

#
# define shapes and transformation
#
# circle(x0, y0, r)
# cylinder(x0, y0, z0, z1, r)
# cone(x0, y0, z0, z1, r0)
# sphere(x0, y0, z0, r)
# torus(x0, y0, z0, r0, r1)
# rectangle(x0, x1, y0, y1)
# cube(x0, x1, y0, y1, z0, z1)
# triangle(x0, y0, x1, y1, x2, y2) (points in clockwise order)
# pyramid(x0, x1, y0, y1, z0, z1)
# function(Z_of_XY)
# functions(upper_Z_of_XY,lower_Z_of_XY)
# add(part1, part2)
# subtract(part1, part2)
# intersect(part1, part2)
# move(part,dx,dy)
# translate(part,dx,dy,dz)
# rotate(part, angle)
# rotate_x(part, angle)
# rotate_y(part, angle)
# rotate_z(part, angle)
# rotate_90(part)
# rotate_180(part)
# rotate_270(part)
# reflect_x(part)
# reflect_y(part)
# reflect_z(part)
# reflect_xy(part)
# reflect_xz(part)
# reflect_yz(part)
# scale_x(part, x0, sx)
# scale_y(part, y0, sy)
# scale_z(part, z0, sz)
# scale_xy(part, x0, y0, sxy)
# scale_xyz(part, x0, y0, z0, sxyz)
# coscale_x_y(part, x0, y0, y1, angle0, angle1, amplitude, offset)
# coscale_x_z(part, x0, z0, z1, angle0, angle1, amplitude, offset)
# coscale_xy_z(part, x0, y0, z0, z1, angle0, angle1, amplitude, offset)
# taper_x_y(part, x0, y0, y1, s0, s1)
# taper_x_z(part, x0, z0, z1, s0, s1)
# taper_xy_z(part, x0, y0, z0, z1, s0, s1)
# shear_x_y(part, y0, y1, dx0, dx1)
# shear_x_z(part, z0, z1, dx0, dx1)
# (more to come)

def circle(x0, y0, r):
   part = "(((X-x0)**2 + (Y-y0)**2) <= r**2)"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'r',str(r))
   return part

def cylinder(x0, y0, z0, z1, r):
   part = "(((X-x0)**2 + (Y-y0)**2 <= r**2) & (Z >= z0) & (Z <= z1))"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'r',str(r))
   return part

def cone(x0, y0, z0, z1, r0):
   part = cylinder(x0, y0, z0, z1, r0)
   part = taper_xy_z(part, x0, y0, z0, z1, 1.0, 0.0)
   return part

def sphere(x0, y0, z0, r):
   part = "(((X-x0)**2 + (Y-y0)**2 + (Z-z0)**2) <= r**2)"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'r',str(r))
   return part

def torus(x0, y0, z0, r0, r1):
   part = "(((r0 - sqrt((X-x0)**2 + (Y-y0)**2))**2 + (Z-z0)**2) <= r1**2)"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'r0',str(r0))
   part = replace(part,'r1',str(r1))
   return part

def rectangle(x0, x1, y0, y1):
   part = "((X >= x0) & (X <= x1) & (Y >= y0) & (Y <= y1))"
   part = replace(part,'x0',str(x0))
```

```
   part = replace(part,'x1',str(x1))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   return part

def cube(x0, x1, y0, y1, z0, z1):
   part = "((X >= x0) & (X <= x1) & (Y >= y0) & (Y <= y1) & (Z >= z0) & (Z <= z1))"
   part = replace(part,'x0',str(x0))
   part = replace(part,'x1',str(x1))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   return part

def triangle(x0, y0, x1, y1, x2, y2): # points in clockwise order
   part = "((((y1-y0)*(X-x0)-(x1-x0)*(Y-y0)) >= 0) & (((y2-y1)*(X-x1)-(x2-x1)*(Y-y1)) >= 0) & (((y0-y2)*(X-x2)-(x0-x2)*(Y-y2)) >= 0))"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'x1',str(x1))
   part = replace(part,'y1',str(y1))
   part = replace(part,'x2',str(x2))
   part = replace(part,'y2',str(y2))
   return part

def pyramid(x0, x1, y0, y1, z0, z1):
   part = cube(x0, x1, y0, y1, z0, z1)
   part = taper_xy_z(part, (x0+x1)/2., (y0+y1)/2., z0, z1, 1.0, 0.0)
   return part

def function(Z_of_XY):
   part = '(Z <= '+Z_of_XY+')'
   return part

def functions(upper_Z_of_XY,lower_Z_of_XY):
   part = '(Z <= '+upper_Z_of_XY+') & (Z >= '+lower_Z_of_XY+')'
   return part

def add(part1, part2):
   part = "part1 | part2"
   part = replace(part,'part1',part1)
   part = replace(part,'part2',part2)
   return part

def subtract(part1, part2):
   part = "(part1) & ~(part2)"
   part = replace(part,'part1',part1)
   part = replace(part,'part2',part2)
   return part

def intersect(part1, part2):
   part = "(part1) & (part2)"
   part = replace(part,'part1',part1)
   part = replace(part,'part2',part2)
   return part

def move(part,dx,dy):
   part = replace(part,'X','(X-'+str(dx)+')')
   part = replace(part,'Y','(Y-'+str(dy)+')')
   return part

def translate(part,dx,dy,dz):
   part = replace(part,'X','(X-'+str(dx)+')')
   part = replace(part,'Y','(Y-'+str(dy)+')')
   part = replace(part,'Z','(Z-'+str(dz)+')')
   return part

def rotate(part, angle):
   angle = angle*pi/180
   part = replace(part,'X','(cos(angle)*X+sin(angle)*y)')
   part = replace(part,'Y','(-sin(angle)*X+cos(angle)*y)')
   part = replace(part,'y','Y')
   part = replace(part,'angle',str(angle))
   return part

def rotate_x(part, angle):
   angle = angle*pi/180
   part = replace(part,'Y','(cos(angle)*Y+sin(angle)*z)')
   part = replace(part,'Z','(-sin(angle)*Y+cos(angle)*z)')
   part = replace(part,'z','Z')
   part = replace(part,'angle',str(angle))
   return part

def rotate_y(part, angle):
   angle = angle*pi/180
   part = replace(part,'X','(cos(angle)*X+sin(angle)*z)')
   part = replace(part,'Z','(-sin(angle)*X+cos(angle)*z)')
   part = replace(part,'z','Z')
   part = replace(part,'angle',str(angle))
   return part

def rotate_z(part, angle):
   angle = angle*pi/180
   part = replace(part,'X','(cos(angle)*X+sin(angle)*y)')
   part = replace(part,'Y','(-sin(angle)*X+cos(angle)*y)')
   part = replace(part,'y','Y')
   part = replace(part,'angle',str(angle))
   return part

def rotate_90(part):
   part = reflect_xy(part)
```

```
   part = reflect_y(part)
   return part

def rotate_180(part):
   part = reflect_xy(part)
   part = reflect_y(part)
   part = reflect_xy(part)
   part = reflect_y(part)
   return part

def rotate_270(part):
   part = reflect_xy(part)
   part = reflect_y(part)
   part = reflect_xy(part)
   part = reflect_y(part)
   part = reflect_xy(part)
   part = reflect_y(part)
   return part

def reflect_x(part):
   part = replace(part,'X','-X')
   return part

def reflect_y(part):
   part = replace(part,'Y','-Y')
   return part

def reflect_z(part):
   part = replace(part,'Z','-Z')
   return part

def reflect_xy(part):
   part = replace(part,'X','temp')
   part = replace(part,'Y','X')
   part = replace(part,'temp','Y')
   return part

def reflect_xz(part):
   part = replace(part,'X','temp')
   part = replace(part,'Z','X')
   part = replace(part,'temp','Z')
   return part

def reflect_yz(part):
   part = replace(part,'Y','temp')
   part = replace(part,'Z','Y')
   part = replace(part,'temp','Z')
   return part

def scale_x(part, x0, sx):
   part = replace(part,'X','(x0 + (X-x0)/sx)')
   part = replace(part,'x0',str(x0))
   part = replace(part,'sx',str(sx))
   return part

def scale_y(part, y0, sy):
   part = replace(part,'Y','(y0 + (Y-y0)/sy)')
   part = replace(part,'y0',str(y0))
   part = replace(part,'sy',str(sy))
   return part

def scale_z(part, z0, sz):
   part = replace(part,'Z','(z0 + (Z-z0)/sz)')
   part = replace(part,'z0',str(z0))
   part = replace(part,'sz',str(sz))
   return part

def scale_xy(part, x0, y0, sxy):
   part = replace(part,'X','(x0 + (X-x0)/sxy)')
   part = replace(part,'Y','(y0 + (Y-y0)/sxy)')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'sxy',str(sxy))
   return part

def scale_xyz(part, x0, y0, z0, sxyz):
   part = replace(part,'X','(x0 + (X-x0)/sxyz)')
   part = replace(part,'Y','(y0 + (Y-y0)/sxyz)')
   part = replace(part,'Z','(z0 + (Z-z0)/sxyz)')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'sxyz',str(sxyz))
   return part

def coscale_x_y(part, x0, y0, y1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
   part = replace(part,'X','(x0 + (X-x0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Y-y0)/(y1-y0))))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

def coscale_x_z(part, x0, z0, z1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
```

```
   part = replace(part,'X','(x0 + (X-x0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0))))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

def coscale_xy_z(part, x0, y0, z0, z1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
   part = replace(part,'X','(x0 + (X-x0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0))))')
   part = replace(part,'Y','(y0 + (Y-y0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0))))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

def taper_x_y(part, x0, y0, y1, s0, s1):
   part = replace(part,'X','(x0 + (X-x0)*(y1-y0)/(s1*(Y-y0) + s0*(y1-Y)))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'s0',str(s0))
   part = replace(part,'s1',str(s1))
   return part

def taper_x_z(part, x0, z0, z1, s0, s1):
   part = replace(part,'X','(x0 + (X-x0)*(z1-z0)/(s1*(Z-z0) + s0*(z1-Z)))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'s0',str(s0))
   part = replace(part,'s1',str(s1))
   return part

def taper_xy_z(part, x0, y0, z0, z1, s0, s1):
   part = replace(part,'X','(x0 + (X-x0)*(z1-z0)/(s1*(Z-z0) + s0*(z1-Z)))')
   part = replace(part,'Y','(y0 + (Y-y0)*(z1-z0)/(s1*(Z-z0) + s0*(z1-Z)))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'s0',str(s0))
   part = replace(part,'s1',str(s1))
   return part

def shear_x_y(part, y0, y1, dx0, dx1):
   part = replace(part,'X','(X - dx0 - (dx1-dx0)*(Y-y0)/(y1-y0))')
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'dx0',str(dx0))
   part = replace(part,'dx1',str(dx1))
   return part

def shear_x_z(part, z0, z1, dx0, dx1):
   part = replace(part,'X','(X - dx0 - (dx1-dx0)*(Z-z0)/(z1-z0))')
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'dx0',str(dx0))
   part = replace(part,'dx1',str(dx1))
   return part

def coshear_x_z(part, z0, z1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
   part = replace(part,'X','(X - offset - amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0)))')
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

#
# PCB classes and definitions
#

cad.labels = []

class PCB:
   def __init__(self,x0,y0,width,height,mask):
      self.board = "False"
      self.interior = rectangle(x0,x0+width,y0,y0+height)
      self.exterior = subtract("True",rectangle(x0,x0+width,y0,y0+height))
      self.mask = "False"
   def add(self,part):
      self.board = add(self.board,part)
      self.mask = add(self.mask,move(part,-mask,mask))
      self.mask = add(self.mask,move(part,-mask,-mask))
      self.mask = add(self.mask,move(part,mask,mask))
      self.mask = add(self.mask,move(part,mask,-mask))
      return self
```

```
class part:
   def add(self,pcb,x,y,z=0,angle=0):
      self.x = x
      self.y = y
      self.z = z
      self.angle = angle
      if (angle == 90):
         self.shape = rotate_90(self.shape)
      elif (angle == 180):
         self.shape = rotate_180(self.shape)
      elif ((angle == 270) | (angle == -90)):
         self.shape = rotate_270(self.shape)
      angle = pi*angle/180
      self.shape = translate(self.shape,x,y,z)
      for i in range(len(self.pad)):
         xnew = cos(angle)*self.pad[i].x + sin(angle)*self.pad[i].y
         ynew = -sin(angle)*self.pad[i].x + cos(angle)*self.pad[i].y
           self.pad[i].x = x + xnew
           self.pad[i].y = y + ynew
         self.pad[i].z += z
      cad.labels.append(cad_text(x,y,z,self.value,size=14))
      for i in range(len(self.labels)):
         xnew = cos(angle)*self.labels[i].x + sin(angle)*self.labels[i].y
         ynew = -sin(angle)*self.labels[i].x + cos(angle)*self.labels[i].y
           self.labels[i].x = x + xnew
           self.labels[i].y = y + ynew
         self.labels[i].z += z
           cad.labels.append(self.labels[i])
      pcb = pcb.add(self.shape)
      return pcb

def wire(pcb,width,*points):
   for i in range(1,len(points)):
      x0 = points[i-1].x
      y0 = points[i-1].y
      z0 = points[i-1].z
      x1 = points[i].x
      y1 = points[i].y
      z1 = points[i].z
      if (x0 < x1):
         pcb.board = add(pcb.board,cube(x0-width/2,x1+width/2,y0-width/2,y0+width/2,z0,z0))
      elif (x1 < x0):
         pcb.board = add(pcb.board,cube(x1-width/2,x0+width/2,y0-width/2,y0+width/2,z0,z0))
      if (y0 < y1):
         pcb.board = add(pcb.board,cube(x1-width/2,x1+width/2,y0-width/2,y1+width/2,z0,z0))
      elif (y1 < y0):
         pcb.board = add(pcb.board,cube(x1-width/2,x1+width/2,y1-width/2,y0+width/2,z0,z0))
   return pcb

#
# PCB library
#

#
# discretes
#

pad_0402 = cube(-.0175,.0175,-.014,.014,0,0)

class R_0402(part):
   #
   # 0402 resistor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_0402,-.0265,0,0)
      self.pad.append(point(-.0265,0,0))
      self.shape = add(self.shape,translate(pad_0402,.0265,0,0))
      self.pad.append(point(.0265,0,0))

pad_1206 = cube(-.032,.032,-.034,.034,0,0)

class R_1206(part):
   #
   # 1206 resistor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.06,0,0))
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.06,0,0))

class C_1206(part):
   #
   # 1206 capacitor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.06,0,0))
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.06,0,0))
```

```
pad_1210 = cube(-.032,.032,-.048,.048,0,0)

class L_1210(part):
   #
   # 1210 inductor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_1210,-.06,0,0)
      self.pad.append(point(-.06,0,0))
      self.shape = add(self.shape,translate(pad_1210,.06,0,0))
      self.pad.append(point(.06,0,0))

pad_choke = cube(-.06,.06,-.06,.06,0,0)

class choke(part):
   #
   # Panasonic ELLCTV
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_choke,-.177,-.177,0)
      self.pad.append(point(-.177,-.177,0))
      self.shape = add(self.shape,translate(pad_choke,.177,.177,0))
      self.pad.append(point(.177,.177,0))

#
# connectors
#

pad_MTA = cube(-.021,.021,-.041,.041,0,0)
pad_MTA_solder = cube(-.071,.071,-.041,.041,0,0)

class MTA_2(part):
   #
   # AMP 1445121-2
   # MTA .050 SMT 2-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1
      #
      self.shape = translate(pad_MTA,-.025,-.1,0)
      self.pad.append(point(-.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,.1,0))
      self.pad.append(point(.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'2',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.187,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.187,0,0))

class MTA_power(part):
   #
   # AMP 1445121-2
   # MTA .050 SMT 2-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: Gnd
      #
      self.shape = translate(pad_MTA,-.025,-.1,0)
      self.pad.append(point(-.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: Vcc
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,.1,0))
      self.pad.append(point(.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Vcc',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.187,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.187,0,0))

class MTA_i0(part):
   #
   # AMP 1445121-3
   # MTA .050 SMT 3-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
```

```
      # pin 1: GND
      #
      self.shape = translate(pad_MTA,.05,.1,0)
      self.pad.append(point(.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1\nGND',14))
      #
      # pin 2: power
      #
      self.shape = add(self.shape,translate(pad_MTA,-.05,.1,0))
      self.pad.append(point(-.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V',14))
      #
      # pin 3: data
      #
      self.shape = add(self.shape,translate(pad_MTA,0,-.1,0))
      self.pad.append(point(0,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'data',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.212,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.212,0,0))

class MTA_serial(part):
   #
   # AMP 1445121-4
   # MTA .050 SMT 4-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: Gnd
      #
      self.shape = translate(pad_MTA,-.075,-.1,0)
      self.pad.append(point(-.075,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: Tx
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,-.1,0))
      self.pad.append(point(.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Tx',14))
      #
      # pin 3: Rx
      #
      self.shape = add(self.shape,translate(pad_MTA,.075,.1,0))
      self.pad.append(point(.075,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Rx',14))
      #
      # pin 4: DTR
      #
      self.shape = add(self.shape,translate(pad_MTA,-.025,.1,0))
      self.pad.append(point(-.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DTR',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.237,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.237,0,0))

class MTA_ICP(part):
   #
   # AMP 1445121-5
   # MTA .050 SMT 4-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: MISO
      #
      self.shape = translate(pad_MTA,-.1,-.1,0)
      self.pad.append(point(-.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'MISO',14))
      #
      # pin 2: GND
      #
      self.shape = add(self.shape,translate(pad_MTA,0,-.1,0))
      self.pad.append(point(0,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))
      #
      # pin 3: MOSI
      #
      self.shape = add(self.shape,translate(pad_MTA,.1,-.1,0))
      self.pad.append(point(.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'MOSI',14))
      #
      # pin 4: -RESET
      #
      self.shape = add(self.shape,translate(pad_MTA,.05,.1,0))
      self.pad.append(point(.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'-RESET',14))
      #
      # pin 5: SCK
      #
      self.shape = add(self.shape,translate(pad_MTA,-.05,.1,0))
      self.pad.append(point(-.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'SCK',14))
```

```
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.262,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.262,0,0))

class power_65mm(part):
   #
   # CUI PJ1-023-SMT
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: power
      #
      self.shape = cube(.433,.512,-.047,.047,0,0)
      self.pad.append(point(.467,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'P',14))
      #
      # pin 2: ground
      #
      self.shape = add(self.shape,cube(.285,.423,-.189,-.098,0,0))
      self.pad.append(point(.354,-.144,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 3: contact
      #
      self.shape = add(self.shape,cube(.325,.463,.098,.189,0,0))
      self.pad.append(point(.394,.144,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,cube(.108,.246,-.169,-.110,0,0))
      self.shape = add(self.shape,cube(.069,.207,.110,.169,0,0))

pad_stereo_2_5mm = cube(-.03,.03,-.05,.05,0,0)

class stereo_2_5mm(part):
   #
   # CUI SJ1-2533-SMT
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: base
      #
      self.shape = translate(pad_stereo_2_5mm,-.130,-.16,0)
      self.pad.append(point(-.130,-.149,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'base',14))
      #
      # pin 2: tip
      #
      self.shape = add(self.shape,translate(pad_stereo_2_5mm,.197,.15,0))
      self.pad.append(point(.197,.141,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'tip',14))
      #
      # pin 3: middle
      #
      self.shape = add(self.shape,translate(pad_stereo_2_5mm,-.012,-.16,0))
      self.pad.append(point(-.012,-.149,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'middle',14))

pad_Molex = cube(-.0155,.0155,-.0265,.0265,0,0)
pad_Molex_solder = cube(-.055,.055,-.065,.065,0,0)

class Molex_serial(part):
   #
   # Molex 53261-0471
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: Rx
      #
      self.shape = translate(pad_Molex,-.075,.064,0)
      self.pad.append(point(-.075,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Rx',14))
      #
      # pin 2: Tx
      #
      self.shape = add(self.shape,translate(pad_Molex,-.025,.064,0))
      self.pad.append(point(-.025,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Tx',14))
      #
      # pin 3: DTR
      #
      self.shape = add(self.shape,translate(pad_Molex,.025,.064,0))
      self.pad.append(point(.025,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DTR',14))
      #
      # pin 4: GND
      #
      self.shape = add(self.shape,translate(pad_Molex,.075,.064,0))
      self.pad.append(point(.075,.064,0))
```

```
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_Molex_solder,-.16,-.065,0))
      self.shape = add(self.shape,translate(pad_Molex_solder,.16,-.065,0))

#
# switches
#

pad_button_6mm = cube(-.04,.04,-.03,.03,0,0)

class button_6mm(part):
   #
   # Omron 6mm pushbutton
   # B3SN-3112P
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # left 1
      #
      self.shape = translate(pad_button_6mm,-.125,.08,0)
      self.pad.append(point(-.125,.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'L1',14))
      #
      # right 1
      #
      self.shape = add(self.shape,translate(pad_button_6mm,-.125,-.08,0))
      self.pad.append(point(-.125,-.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'R1',14))
      #
      # right 2
      #
      self.shape = add(self.shape,translate(pad_button_6mm,.125,-.08,0))
      self.pad.append(point(.125,-.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'R2',14))
      #
      # left 2
      #
      self.shape = add(self.shape,translate(pad_button_6mm,.125,.08,0))
      self.pad.append(point(.125,.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'L2',14))

#
# crystals and resonators
#

pad_XTAL = cube(-.016,.016,-.077,.077,0,0)

class XTAL(part):
   #
   # Panasonic EFOBM series
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # left
      #
      self.shape = translate(pad_XTAL,-.053,0,0)
      self.pad.append(point(-.053,0,0))
      #
      # ground
      #
      self.shape = add(self.shape,translate(pad_XTAL,0,0,0))
      self.pad.append(point(0,0,0))
      #
      # right
      #
      self.shape = add(self.shape,translate(pad_XTAL,.053,0,0))
      self.pad.append(point(.053,0,0))

#
# diodes, transistors, regulators
#

class D_1206(part):
   #
   # 1206 diode
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # anode
      #
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A',14))
      #
      # cathode
      #
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))
```

```
class LED_1206(part):
   #
   # 1206 LED
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # anode
      #
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A',14))
      #
      # cathode
      #
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))

class phototransistor_1206(part):
   #
   # 1206 phototransistor
   # OPTEK 520,521
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # collector
      #
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))
      #
      # emitter
      #
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'E',14))

pad_SOD_123 = cube(-.02,.02,-.024,.024,0,0)

class D_SOD_123(part):
   #
   # SOD-123 diode
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # anode
      #
      self.shape = translate(pad_SOD_123,-.07,0,0)
      self.pad.append(point(-.07,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A',14))
      #
      # cathode
      #
      self.shape = add(self.shape,translate(pad_SOD_123,.07,0,0))
      self.pad.append(point(.07,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))

pad_SOT23 = cube(-.015,.015,-.02,.02,0,0)

class NMOSFET_SOT23(part):
   #
   # Fairchild NDS355AN
   #
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: gate
      #
      self.shape = translate(pad_SOT23,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 2: source
      #
      self.shape = add(self.shape,translate(pad_SOT23,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'S',14))
      #
      # pin 3: drain
      #
      self.shape = add(self.shape,translate(pad_SOT23,0,.045,0))
      self.pad.append(point(0,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'D',14))

class PMOSFET_SOT23(part):
   #
```

```
   # Fairchild NDS356AP
   #
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: gate
      #
      self.shape = translate(pad_SOT23,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 2: source
      #
      self.shape = add(self.shape,translate(pad_SOT23,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'S',14))
      #
      # pin 3: drain
      #
      self.shape = add(self.shape,translate(pad_SOT23,0,.045,0))
      self.pad.append(point(0,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'D',14))

class regulator_SOT23(part):
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: output
      #
      self.shape = translate(pad_SOT23,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: input
      #
      self.shape = add(self.shape,translate(pad_SOT23,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'in',14))
      #
      # pin 3: ground
      #
      self.shape = add(self.shape,translate(pad_SOT23,0,.045,0))
      self.pad.append(point(0,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'gnd',14))

pad_SOT223 = cube(-.02,.02,-.03,.03,0,0)
pad_SOT223_ground = cube(-.065,.065,-.03,.03,0,0)

class regulator_SOT223(part):
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: input
      #
      self.shape = translate(pad_SOT223,-.09,-.12,0)
      self.pad.append(point(-.09,-.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1I',14))
      #
      # pin 2: ground
      #
      self.shape = add(self.shape,translate(pad_SOT223,0,-.12,0))
      self.pad.append(point(0,-.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 3: output
      #
      self.shape = add(self.shape,translate(pad_SOT223,.09,-.12,0))
      self.pad.append(point(.09,-.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'O',14))
      #
      # pin 4: ground
      #
      self.shape = add(self.shape,translate(pad_SOT223_ground,0,.12,0))
      self.pad.append(point(0,.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))

pad_SM8 = cube(-.035,.035,-.016,.016,0,0)

class H_bridge_SM8(part):
   #
   # Zetex ZXMHC3A01T8
   #
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
```

```
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      d = .13
      #
      # pin 1: G3 (right N gate)
      #
      self.shape = translate(pad_SM8,-d,.09,0)
      self.pad.append(point(-d,.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GRN',14))
      #
      # pin 2: S2 S3 (N source)
      #
      self.shape = add(self.shape,translate(pad_SM8,-d,.03,0))
      self.pad.append(point(-d,.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'SN',14))
      #
      # pin 3: G2 (left N gate)
      #
      self.shape = add(self.shape,translate(pad_SM8,-d,-.03,0))
      self.pad.append(point(-d,-.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GLN',14))
      #
      # pin 4: G1 (left P gate)
      #
      self.shape = add(self.shape,translate(pad_SM8,-d,-.09,0))
      self.pad.append(point(-d,-.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GLP',14))
      #
      # pin 5: D1 D2 (left drain)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,-.09,0))
      self.pad.append(point(d,-.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DL',14))
      #
      # pin 6: S1 S4 (P source)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,-.03,0))
      self.pad.append(point(d,-.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'SP',14))
      #
      # pin 7: D3 D4 (right drain)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,.03,0))
      self.pad.append(point(d,.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DR',14))
      #
      # pin 8: G4 (right N gate)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,.09,0))
      self.pad.append(point(d,.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GRP',14))

#
# ICs
#

pad_SOT23_5 = cube(-.01,.01,-.02,.02,0,0)

class op_amp_SOT23_5(part):
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: output
      #
      self.shape = translate(pad_SOT23_5,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'O',14))
      #
      # pin 2: V-
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,0,-.045,0))
      self.pad.append(point(0,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V-',14))
      #
      # pin 3: I+
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'I+',14))
      #
      # pin 4: I-
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,.0375,.045,0))
      self.pad.append(point(.0375,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'I-',14))
      #
      # pin 5: V+
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,-.0375,.045,0))
      self.pad.append(point(-.0375,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V+',14))

pad_SOICN = cube(-.035,.035,-.015,.015,0,0)

class op_amp_SOICN(part):
```

```
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: A out
      #
      self.shape = translate(pad_SOICN,-.12,.075,0)
      self.pad.append(point(-.12,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1 Ao',14))
      #
      # pin 2: A-
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.025,0))
      self.pad.append(point(-.12,.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A-',14))
      #
      # pin 3: A+
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.025,0))
      self.pad.append(point(-.12,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A+',14))
      #
      # pin 4: V-
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.075,0))
      self.pad.append(point(-.12,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V-',14))
      #
      # pin 5: B+
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.075,0))
      self.pad.append(point(.12,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'B+',14))
      #
      # pin 6: B-
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.025,0))
      self.pad.append(point(.12,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'B-',14))
      #
      # pin 7: B out
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.025,0))
      self.pad.append(point(.12,.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Bo',14))
      #
      # pin 8: V+
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.075,0))
      self.pad.append(point(.12,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V+',14))

pad_SOIC = cube(-.043,.043,-.015,.015,0,0)

class ATtiny45_SOIC(part):
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: PB5/dW/ADC0/-RESET/PCINT5
      #
      self.shape = translate(pad_SOIC,-.14,.075,0)
      self.pad.append(point(-.14,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: PB3/ADC3/-OC1B/CLKI/XTAL1/PCINT3
      #
      self.shape = add(self.shape,translate(pad_SOIC,-.14,.025,0))
      self.pad.append(point(-.14,.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB3',14))
      #
      # pin 3: PB4/ADC2/OC1B/CLKO/XTAL2/PCINT4
      #
      self.shape = add(self.shape,translate(pad_SOIC,-.14,-.025,0))
      self.pad.append(point(-.14,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB4',14))
      #
      # pin 4: GND
      #
      self.shape = add(self.shape,translate(pad_SOIC,-.14,-.075,0))
      self.pad.append(point(-.14,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))
      #
      # pin 5: PB0/MOSI/DI/SDA/AIN0/OC0A/-OC1A/AREF/PCINT0
      #
      self.shape = add(self.shape,translate(pad_SOIC,.14,-.075,0))
      self.pad.append(point(.14,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 6: PB1/MISO/DO/AIN1/OC0B/OC1A/PCINT1
      #
      self.shape = add(self.shape,translate(pad_SOIC,.14,-.025,0))
      self.pad.append(point(.14,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB1',14))
      #
      # pin 7: PB2/SCK/USCK/SCL/ADC1/T0/INT0/PCINT2
      #
      self.shape = add(self.shape,translate(pad_SOIC,.14,.025,0))
      self.pad.append(point(.14,.025,0))
```

```
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB2',14))
      #
      # pin 8: VCC
      #
      self.shape = add(self.shape,translate(pad_SOIC,.14,.075,0))
      self.pad.append(point(.14,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'VCC',14))

class ATtiny44_SOICN(part):
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: VCC
      #
      self.shape = translate(pad_SOICN,-.12,.15,0)
      self.pad.append(point(-.12,.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: PB0/XTAL1/PCINT8
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 3: PB1/XTAL2/PCINT9
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.050,0))
      self.pad.append(point(-.12,.05,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB1',14))
      #
      # pin 4: PB3/dW/-RESET/PCINT11
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,0,0))
      self.pad.append(point(-.12,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB3',14))
      #
      # pin 5: PB2/CKOUT/OC0A/INT0/PCINT10
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.05,0))
      self.pad.append(point(-.12,-.05,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB2',14))
      #
      # pin 6: PA7/ADC7/OC0B/ICP/PCINT7
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.1,0))
      self.pad.append(point(-.12,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA7',14))
      #
      # pin 7: PA6/ADC6/MOSI/SDA/OC1A/PCINT6
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.15,0))
      self.pad.append(point(-.12,-.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA6',14))
      #
      # pin 8: PA5/ADC5/DO/MISO/OC1B/PCINT5
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.15,0))
      self.pad.append(point(.12,-.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA5',14))
      #
      # pin 9: PA4/ADC4/USCK/SCL/T1/PCINT4
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.1,0))
      self.pad.append(point(.12,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA4',14))
      #
      # pin 10: PA3/ADC3/T0/PCINT3
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.05,0))
      self.pad.append(point(.12,-.05,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA3',14))
      #
      # pin 11: PA2/ADC2/AIN1/PCINT2
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,0,0))
      self.pad.append(point(.12,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA2',14))
      #
      # pin 12: PA1/ADC1/AIN0/PCINT1
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.050,0))
      self.pad.append(point(.12,.05,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA1',14))
      #
      # pin 13: PA0/ADC0/AREF/PCINT0
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.1,0))
      self.pad.append(point(.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA0',14))
      #
      # pin 14: GND
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.15,0))
      self.pad.append(point(.12,.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))

pad_TQFP = cube(-.043,.043,-.015,.015,0,0)

class ATmega88_TQFP(part):
```

```
def __init__(self,value=''):
   self.value = value
   self.pad = [point(0,0,0)]
   self.labels = []

   #
   # pin 1: PD3/PCINT19/OC2B/INT1
   #
   self.shape = translate(pad_SOICN,-.12,.15,0)
   self.pad.append(point(-.12,.15,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
   #
   # pin 2: PD4/PCINT20/XCK/T0
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 3: GND
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 4: VCC
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 5: GND
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 6: VCC
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 7: PB6/PCINT6/XTAL1/TOSC1
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 8: PB7/PCINT7/XTAL2/TOSC2
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

   #
   # pin 9: PD5/PCINT21/OC0B/T1
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 10: PD6/PCINT22/OC0A/AIN0
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 11: PD7/PCINT23/AIN1
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 12: PB0/PCINT0/CLKO/ICP1
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 13: PB1/PCINT1/OC1A
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 14: PB2/PCINT2/-SS/OC1B
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 15: PB3/PCINT3/OC2A/MOSI
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 16: PB4/PCINT4/MISO
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
```

```
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

        #
        # pin 17: PB5/SCK/PCINT5
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 18: AVCC
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 19: ADC6
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 20: AREF
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 21: GND
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 22: ADC7
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 23: PC0/ADC0/PCINT8
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 24: PC1/ADC1/PCINT9
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

        #
        # pin 25: PC2/ADC2/PCINT10
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 26: PC2/ADC3/PCINT11
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 27: PC4/ADC4/SDA/PCINT12
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 28: PC5/ADC5/SCL/PCINT13
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 29: PC6/-RESET/PCINT14
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 30: PD0/RXD/PCINT16
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 31: PD1/TXD/PCINT17
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 32: PD2/INT0/PCINT18
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

#
```

```
# graphics
#

class CBA(part):
   def __init__(self,r=.02):
      self.value = ''
      self.pad = [point(0,0,0)]
      self.labels = []
      d = 3*r
      self.shape = circle(0,0,r)
      self.shape = add(self.shape,translate(circle(0,0,r),-d,d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),-d,0,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),-d,-d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),0,-d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),d,-d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),d,0,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),d,d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),0,d,0))

class hello(part):
   #
   # HELLO
   #
   def __init__(self,w,z):
      self.value = ''
      self.pad = [point(0,0,0)]
      self.labels = []
      # H
      self.shape = rectangle(-.01,.01,0,.1)
      self.shape = add(self.shape,rectangle(0,.05,.04,.06))
      self.shape = add(self.shape,rectangle(.04,.06,0,.1))
      # E
      self.shape = add(self.shape,rectangle(.08,.1,0,.1))
      self.shape = add(self.shape,rectangle(.1,.14,0,.02))
      self.shape = add(self.shape,rectangle(.1,.13,.04,.06))
      self.shape = add(self.shape,rectangle(.1,.14,.08,.1))
      # L
      self.shape = add(self.shape,rectangle(.16,.18,0,.1))
      self.shape = add(self.shape,rectangle(.18,.21,0,.02))
      # L
      self.shape = add(self.shape,rectangle(.23,.25,0,.1))
      self.shape = add(self.shape,rectangle(.25,.28,0,.02))
      # 0
      self.shape = add(self.shape,rectangle(.3,.32,0,.1))
      self.shape = add(self.shape,rectangle(.32,.345,0,.02))
      self.shape = add(self.shape,rectangle(.32,.345,.08,.1))
      self.shape = add(self.shape,rectangle(.345,.365,0,.1))

#
# define board
#

x0 = 1
y0 = 1
width = 1.18
height = .7
z = -.005
w = .018
mask = .004

pcb = PCB(x0,y0,width,height,mask)

IC1 = ATtiny45_SOIC('IC1\nt45')
pcb = IC1.add(pcb,x0+.74,y0+.35,z,angle=-90)

R1 = R_1206('R1\n10k')
pcb = R1.add(pcb,IC1.pad[1].x-.09,IC1.y,z,angle=90)

pcb = wire(pcb,w,
   IC1.pad[1],
   R1.pad[2])

pcb = wire(pcb,w,
   IC1.pad[8],
   R1.pad[1])

IC2 = regulator_SOT23('IC2\n5V')
pcb = IC2.add(pcb,R1.x-.155,R1.y,z,angle=90)

pcb = wire(pcb,w,
   IC2.pad[3],
   IC1.pad[4])

C1 = C_1206('C1\n3.3uF')
pcb = C1.add(pcb,IC2.x,IC1.pad[8].y,z)

pcb = wire(pcb,w,
   C1.pad[1],
   IC2.pad[1])

pcb = wire(pcb,w,
   IC2.pad[3],
   C1.pad[2])

pcb = wire(pcb,w,
   C1.pad[1],
   point(C1.pad[1].x,IC1.pad[8].y+.1,z),
   IC1.pad[8])

J1 = MTA_serial('J1')
pcb = J1.add(pcb,IC2.x-.26,IC1.y-.01,z,angle=-90)
```

```
pcb = wire(pcb,w,
    IC2.pad[3],
    point(IC2.pad[3].x,IC1.pad[1].y,z),
    J1.pad[1])

pcb = wire(pcb,w,
    J1.pad[4],
    IC2.pad[2])

pcb = wire(pcb,w,
    J1.pad[3],
    point(J1.pad[2].x,IC1.pad[7].y+.15,z),
    IC1.pad[7])

J2 = MTA_ICP('J2')
pcb = J2.add(pcb,IC1.x+.28,IC1.y,z,angle=90)

pcb = wire(pcb,w,
    IC1.pad[6],
    point(IC1.pad[6].x,IC1.pad[6].y+.1,z),
    point(J2.pad[1].x-.03,J2.pad[1].y,z),
    J2.pad[1])

pcb = wire(pcb,w,
    J2.pad[2],
    IC1.pad[4])

pcb = wire(pcb,w,
    J2.pad[3],
    point(J2.x,J2.pad[5].y,z),
    IC1.pad[5])

pcb = wire(pcb,w,
    J2.pad[4],
    point(J2.pad[4].x,J2.pad[3].y-.05,z),
    point(J2.pad[3].x,IC1.pad[1].y-.12,z),
    IC1.pad[1])

pcb = wire(pcb,w,
    IC1.pad[7],
    point(IC1.pad[7].x,IC1.pad[7].y+.15,z),
    point(J2.pad[1].x+.02,J2.pad[1].y+.05,z),
    J2.pad[5])

H1 = hello(w,z)
pcb = H1.add(pcb,x0+.25,y0+.035,z)

#
# assign board and exterior to cad.function for milling
#
# use single-sided FR2
# use 1/64 end-mill
# set tool diameter = .0156
# set xy, z speed = 4
# set # contours = -1
#

cad.function = add(pcb.board,pcb.exterior)

#
# uncomment to export traces
#

#cad.function = pcb.board

#
# uncomment to export exterior
#

#cad.function = pcb.exterior

#
# uncomment to export solder mask
#

#cad.function = pcb.mask

#
# uncomment to mill out board
#
# use 1/32 end-mill
# set tool diameter = .0312
# set xy, z speed = .5
# set # contours = 1
#

#cad.function = pcb.interior
#z = -.065

#
# define limits and parameters
#

d = 0.05
cad.xmin = x0-d # min x to render
cad.xmax = x0+width+d # max x to render
cad.ymin = y0-d # min y to render
cad.ymax = y0+height+d # max y to render
cad.zmin = z
cad.zmax = .050
```

```
dpi = 200 # low resolution for previewing
#dpi = 500 # high resolution for machining
nxy = int(dpi*(cad.xmax-cad.xmin))
cad.nx = int(dpi*(cad.xmax-cad.xmin)) # x points to render
cad.ny = int(dpi*(cad.ymax-cad.ymin)) # y points to render
cad.nz = 1
cad.inches_per_unit = 1.0 # use inch units
cad.view('xy') # 2D view
```

```
#
# hello.serial.i0.45.cad
#
# tiny45 serial I/O hello-world
#
# Neil Gershenfeld
# CBA MIT 11/15/07
#
# (c) Massachusetts Institute of Technology 2007
# Permission granted for experimental and personal use;
# license for commercial sale available from MIT.
#

#
# define shapes and transformation
#
# circle(x0, y0, r)
# cylinder(x0, y0, z0, z1, r)
# cone(x0, y0, z0, z1, r0)
# sphere(x0, y0, z0, r)
# torus(x0, y0, z0, r0, r1)
# rectangle(x0, x1, y0, y1)
# cube(x0, x1, y0, y1, z0, z1)
# triangle(x0, y0, x1, y1, x2, y2) (points in clockwise order)
# pyramid(x0, x1, y0, y1, z0, z1)
# function(Z_of_XY)
# functions(upper_Z_of_XY,lower_Z_of_XY)
# add(part1, part2)
# subtract(part1, part2)
# intersect(part1, part2)
# move(part,dx,dy)
# translate(part,dx,dy,dz)
# rotate(part, angle)
# rotate_x(part, angle)
# rotate_y(part, angle)
# rotate_z(part, angle)
# rotate_90(part)
# rotate_180(part)
# rotate_270(part)
# reflect_x(part)
# reflect_y(part)
# reflect_z(part)
# reflect_xy(part)
# reflect_xz(part)
# reflect_yz(part)
# scale_x(part, x0, sx)
# scale_y(part, y0, sy)
# scale_z(part, z0, sz)
# scale_xy(part, x0, y0, sxy)
# scale_xyz(part, x0, y0, z0, sxyz)
# coscale_x_y(part, x0, y0, y1, angle0, angle1, amplitude, offset)
# coscale_x_z(part, x0, z0, z1, angle0, angle1, amplitude, offset)
# coscale_xy_z(part, x0, y0, z0, z1, angle0, angle1, amplitude, offset)
# taper_x_y(part, x0, y0, y1, s0, s1)
# taper_x_z(part, x0, z0, z1, s0, s1)
# taper_xy_z(part, x0, y0, z0, z1, s0, s1)
# shear_x_y(part, y0, y1, dx0, dx1)
# shear_x_z(part, z0, z1, dx0, dx1)
# (more to come)

def circle(x0, y0, r):
   part = "(((X-x0)**2 + (Y-y0)**2) <= r**2)"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'r',str(r))
   return part

def cylinder(x0, y0, z0, z1, r):
   part = "(((X-x0)**2 + (Y-y0)**2 <= r**2) & (Z >= z0) & (Z <= z1))"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'r',str(r))
   return part

def cone(x0, y0, z0, z1, r0):
   part = cylinder(x0, y0, z0, z1, r0)
   part = taper_xy_z(part, x0, y0, z0, z1, 1.0, 0.0)
   return part

def sphere(x0, y0, z0, r):
   part = "(((X-x0)**2 + (Y-y0)**2 + (Z-z0)**2) <= r**2)"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'r',str(r))
   return part

def torus(x0, y0, z0, r0, r1):
   part = "(((r0 - sqrt((X-x0)**2 + (Y-y0)**2))**2 + (Z-z0)**2) <= r1**2)"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'r0',str(r0))
   part = replace(part,'r1',str(r1))
   return part

def rectangle(x0, x1, y0, y1):
   part = "((X >= x0) & (X <= x1) & (Y >= y0) & (Y <= y1))"
   part = replace(part,'x0',str(x0))
```

```
   part = replace(part,'x1',str(x1))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   return part

def cube(x0, x1, y0, y1, z0, z1):
   part = "((X >= x0) & (X <= x1) & (Y >= y0) & (Y <= y1) & (Z >= z0) & (Z <= z1))"
   part = replace(part,'x0',str(x0))
   part = replace(part,'x1',str(x1))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   return part

def triangle(x0, y0, x1, y1, x2, y2): # points in clockwise order
   part = "((((y1-y0)*(X-x0)-(x1-x0)*(Y-y0)) >= 0) & (((y2-y1)*(X-x1)-(x2-x1)*(Y-y1)) >= 0) & (((y0-y2)*(X-x2)-(x0-x2)*(Y-y2)) >= 0))"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'x1',str(x1))
   part = replace(part,'y1',str(y1))
   part = replace(part,'x2',str(x2))
   part = replace(part,'y2',str(y2))
   return part

def pyramid(x0, x1, y0, y1, z0, z1):
   part = cube(x0, x1, y0, y1, z0, z1)
   part = taper_xy_z(part, (x0+x1)/2., (y0+y1)/2., z0, z1, 1.0, 0.0)
   return part

def function(Z_of_XY):
   part = '(Z <= '+Z_of_XY+')'
   return part

def functions(upper_Z_of_XY,lower_Z_of_XY):
   part = '(Z <= '+upper_Z_of_XY+') & (Z >= '+lower_Z_of_XY+')'
   return part

def add(part1, part2):
   part = "part1 | part2"
   part = replace(part,'part1',part1)
   part = replace(part,'part2',part2)
   return part

def subtract(part1, part2):
   part = "(part1) & ~(part2)"
   part = replace(part,'part1',part1)
   part = replace(part,'part2',part2)
   return part

def intersect(part1, part2):
   part = "(part1) & (part2)"
   part = replace(part,'part1',part1)
   part = replace(part,'part2',part2)
   return part

def move(part,dx,dy):
   part = replace(part,'X','(X-'+str(dx)+')')
   part = replace(part,'Y','(Y-'+str(dy)+')')
   return part

def translate(part,dx,dy,dz):
   part = replace(part,'X','(X-'+str(dx)+')')
   part = replace(part,'Y','(Y-'+str(dy)+')')
   part = replace(part,'Z','(Z-'+str(dz)+')')
   return part

def rotate(part, angle):
   angle = angle*pi/180
   part = replace(part,'X','(cos(angle)*X+sin(angle)*y)')
   part = replace(part,'Y','(-sin(angle)*X+cos(angle)*y)')
   part = replace(part,'y','Y')
   part = replace(part,'angle',str(angle))
   return part

def rotate_x(part, angle):
   angle = angle*pi/180
   part = replace(part,'Y','(cos(angle)*Y+sin(angle)*z)')
   part = replace(part,'Z','(-sin(angle)*Y+cos(angle)*z)')
   part = replace(part,'z','Z')
   part = replace(part,'angle',str(angle))
   return part

def rotate_y(part, angle):
   angle = angle*pi/180
   part = replace(part,'X','(cos(angle)*X+sin(angle)*z)')
   part = replace(part,'Z','(-sin(angle)*X+cos(angle)*z)')
   part = replace(part,'z','Z')
   part = replace(part,'angle',str(angle))
   return part

def rotate_z(part, angle):
   angle = angle*pi/180
   part = replace(part,'X','(cos(angle)*X+sin(angle)*y)')
   part = replace(part,'Y','(-sin(angle)*X+cos(angle)*y)')
   part = replace(part,'y','Y')
   part = replace(part,'angle',str(angle))
   return part

def rotate_90(part):
   part = reflect_xy(part)
```

```
   part = reflect_y(part)
   return part

def rotate_180(part):
   part = reflect_xy(part)
   part = reflect_y(part)
   part = reflect_xy(part)
   part = reflect_y(part)
   return part

def rotate_270(part):
   part = reflect_xy(part)
   part = reflect_y(part)
   part = reflect_xy(part)
   part = reflect_y(part)
   part = reflect_xy(part)
   part = reflect_y(part)
   return part

def reflect_x(part):
   part = replace(part,'X','-X')
   return part

def reflect_y(part):
   part = replace(part,'Y','-Y')
   return part

def reflect_z(part):
   part = replace(part,'Z','-Z')
   return part

def reflect_xy(part):
   part = replace(part,'X','temp')
   part = replace(part,'Y','X')
   part = replace(part,'temp','Y')
   return part

def reflect_xz(part):
   part = replace(part,'X','temp')
   part = replace(part,'Z','X')
   part = replace(part,'temp','Z')
   return part

def reflect_yz(part):
   part = replace(part,'Y','temp')
   part = replace(part,'Z','Y')
   part = replace(part,'temp','Z')
   return part

def scale_x(part, x0, sx):
   part = replace(part,'X','(x0 + (X-x0)/sx)')
   part = replace(part,'x0',str(x0))
   part = replace(part,'sx',str(sx))
   return part

def scale_y(part, y0, sy):
   part = replace(part,'Y','(y0 + (Y-y0)/sy)')
   part = replace(part,'y0',str(y0))
   part = replace(part,'sy',str(sy))
   return part

def scale_z(part, z0, sz):
   part = replace(part,'Z','(z0 + (Z-z0)/sz)')
   part = replace(part,'z0',str(z0))
   part = replace(part,'sz',str(sz))
   return part

def scale_xy(part, x0, y0, sxy):
   part = replace(part,'X','(x0 + (X-x0)/sxy)')
   part = replace(part,'Y','(y0 + (Y-y0)/sxy)')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'sxy',str(sxy))
   return part

def scale_xyz(part, x0, y0, z0, sxyz):
   part = replace(part,'X','(x0 + (X-x0)/sxyz)')
   part = replace(part,'Y','(y0 + (Y-y0)/sxyz)')
   part = replace(part,'Z','(z0 + (Z-z0)/sxyz)')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'sxyz',str(sxyz))
   return part

def coscale_x_y(part, x0, y0, y1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
   part = replace(part,'X','(x0 + (X-x0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Y-y0)/(y1-y0))))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

def coscale_x_z(part, x0, z0, z1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
```

```
   part = replace(part,'X','(x0 + (X-x0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0))))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

def coscale_xy_z(part, x0, y0, z0, z1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
   part = replace(part,'X','(x0 + (X-x0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0))))')
   part = replace(part,'Y','(y0 + (Y-y0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0))))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

def taper_x_y(part, x0, y0, y1, s0, s1):
   part = replace(part,'X','(x0 + (X-x0)*(y1-y0)/(s1*(Y-y0) + s0*(y1-Y)))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'s0',str(s0))
   part = replace(part,'s1',str(s1))
   return part

def taper_x_z(part, x0, z0, z1, s0, s1):
   part = replace(part,'X','(x0 + (X-x0)*(z1-z0)/(s1*(Z-z0) + s0*(z1-Z)))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'s0',str(s0))
   part = replace(part,'s1',str(s1))
   return part

def taper_xy_z(part, x0, y0, z0, z1, s0, s1):
   part = replace(part,'X','(x0 + (X-x0)*(z1-z0)/(s1*(Z-z0) + s0*(z1-Z)))')
   part = replace(part,'Y','(y0 + (Y-y0)*(z1-z0)/(s1*(Z-z0) + s0*(z1-Z)))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'s0',str(s0))
   part = replace(part,'s1',str(s1))
   return part

def shear_x_y(part, y0, y1, dx0, dx1):
   part = replace(part,'X','(X - dx0 - (dx1-dx0)*(Y-y0)/(y1-y0))')
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'dx0',str(dx0))
   part = replace(part,'dx1',str(dx1))
   return part

def shear_x_z(part, z0, z1, dx0, dx1):
   part = replace(part,'X','(X - dx0 - (dx1-dx0)*(Z-z0)/(z1-z0))')
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'dx0',str(dx0))
   part = replace(part,'dx1',str(dx1))
   return part

def coshear_x_z(part, z0, z1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
   part = replace(part,'X','(X - offset - amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0)))')
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

#
# PCB classes and definitions
#

cad.labels = []

class PCB:
   def __init__(self,x0,y0,width,height,mask):
      self.board = "False"
      self.interior = rectangle(x0,x0+width,y0,y0+height)
      self.exterior = subtract("True",rectangle(x0,x0+width,y0,y0+height))
      self.mask = "False"
   def add(self,part):
      self.board = add(self.board,part)
      self.mask = add(self.mask,move(part,-mask,mask))
      self.mask = add(self.mask,move(part,-mask,-mask))
      self.mask = add(self.mask,move(part,mask,mask))
      self.mask = add(self.mask,move(part,mask,-mask))
      return self
```

```
class part:
   def add(self,pcb,x,y,z=0,angle=0):
      self.x = x
      self.y = y
      self.z = z
      self.angle = angle
      if (angle == 90):
         self.shape = rotate_90(self.shape)
      elif (angle == 180):
         self.shape = rotate_180(self.shape)
      elif ((angle == 270) | (angle == -90)):
         self.shape = rotate_270(self.shape)
      angle = pi*angle/180
      self.shape = translate(self.shape,x,y,z)
      for i in range(len(self.pad)):
         xnew = cos(angle)*self.pad[i].x + sin(angle)*self.pad[i].y
         ynew = -sin(angle)*self.pad[i].x + cos(angle)*self.pad[i].y
           self.pad[i].x = x + xnew
           self.pad[i].y = y + ynew
         self.pad[i].z += z
      cad.labels.append(cad_text(x,y,z,self.value,size=14))
      for i in range(len(self.labels)):
         xnew = cos(angle)*self.labels[i].x + sin(angle)*self.labels[i].y
         ynew = -sin(angle)*self.labels[i].x + cos(angle)*self.labels[i].y
           self.labels[i].x = x + xnew
           self.labels[i].y = y + ynew
         self.labels[i].z += z
           cad.labels.append(self.labels[i])
      pcb = pcb.add(self.shape)
      return pcb

def wire(pcb,width,*points):
   for i in range(1,len(points)):
      x0 = points[i-1].x
      y0 = points[i-1].y
      z0 = points[i-1].z
      x1 = points[i].x
      y1 = points[i].y
      z1 = points[i].z
      if (x0 < x1):
         pcb.board = add(pcb.board,cube(x0-width/2,x1+width/2,y0-width/2,y0+width/2,z0,z0))
      elif (x1 < x0):
         pcb.board = add(pcb.board,cube(x1-width/2,x0+width/2,y0-width/2,y0+width/2,z0,z0))
      if (y0 < y1):
         pcb.board = add(pcb.board,cube(x1-width/2,x1+width/2,y0-width/2,y1+width/2,z0,z0))
      elif (y1 < y0):
         pcb.board = add(pcb.board,cube(x1-width/2,x1+width/2,y1-width/2,y0+width/2,z0,z0))
   return pcb

#
# PCB library
#

#
# discretes
#

pad_0402 = cube(-.0175,.0175,-.014,.014,0,0)

class R_0402(part):
   #
   # 0402 resistor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_0402,-.0265,0,0)
      self.pad.append(point(-.0265,0,0))
      self.shape = add(self.shape,translate(pad_0402,.0265,0,0))
      self.pad.append(point(.0265,0,0))

pad_1206 = cube(-.032,.032,-.034,.034,0,0)

class R_1206(part):
   #
   # 1206 resistor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.06,0,0))
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.06,0,0))

class C_1206(part):
   #
   # 1206 capacitor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.06,0,0))
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.06,0,0))
```

```
pad_1210 = cube(-.032,.032,-.048,.048,0,0)

class L_1210(part):
   #
   # 1210 inductor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_1210,-.06,0,0)
      self.pad.append(point(-.06,0,0))
      self.shape = add(self.shape,translate(pad_1210,.06,0,0))
      self.pad.append(point(.06,0,0))

pad_choke = cube(-.06,.06,-.06,.06,0,0)

class choke(part):
   #
   # Panasonic ELLCTV
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_choke,-.177,-.177,0)
      self.pad.append(point(-.177,-.177,0))
      self.shape = add(self.shape,translate(pad_choke,.177,.177,0))
      self.pad.append(point(.177,.177,0))

#
# connectors
#

pad_MTA = cube(-.021,.021,-.041,.041,0,0)
pad_MTA_solder = cube(-.071,.071,-.041,.041,0,0)

class MTA_2(part):
   #
   # AMP 1445121-2
   # MTA .050 SMT 2-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1
      #
      self.shape = translate(pad_MTA,-.025,-.1,0)
      self.pad.append(point(-.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,.1,0))
      self.pad.append(point(.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'2',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.187,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.187,0,0))

class MTA_power(part):
   #
   # AMP 1445121-2
   # MTA .050 SMT 2-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: Gnd
      #
      self.shape = translate(pad_MTA,-.025,-.1,0)
      self.pad.append(point(-.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: Vcc
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,.1,0))
      self.pad.append(point(.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Vcc',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.187,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.187,0,0))

class MTA_i0(part):
   #
   # AMP 1445121-3
   # MTA .050 SMT 3-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
```

```
      # pin 1: GND
      #
      self.shape = translate(pad_MTA,.05,.1,0)
      self.pad.append(point(.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1\nGND',14))
      #
      # pin 2: power
      #
      self.shape = add(self.shape,translate(pad_MTA,-.05,.1,0))
      self.pad.append(point(-.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V',14))
      #
      # pin 3: data
      #
      self.shape = add(self.shape,translate(pad_MTA,0,-.1,0))
      self.pad.append(point(0,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'data',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.212,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.212,0,0))

class MTA_serial(part):
   #
   # AMP 1445121-4
   # MTA .050 SMT 4-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: Gnd
      #
      self.shape = translate(pad_MTA,-.075,-.1,0)
      self.pad.append(point(-.075,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: Tx
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,-.1,0))
      self.pad.append(point(.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Tx',14))
      #
      # pin 3: Rx
      #
      self.shape = add(self.shape,translate(pad_MTA,.075,.1,0))
      self.pad.append(point(.075,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Rx',14))
      #
      # pin 4: DTR
      #
      self.shape = add(self.shape,translate(pad_MTA,-.025,.1,0))
      self.pad.append(point(-.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DTR',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.237,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.237,0,0))

class MTA_ICP(part):
   #
   # AMP 1445121-5
   # MTA .050 SMT 4-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: MISO
      #
      self.shape = translate(pad_MTA,-.1,-.1,0)
      self.pad.append(point(-.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'MISO',14))
      #
      # pin 2: GND
      #
      self.shape = add(self.shape,translate(pad_MTA,0,-.1,0))
      self.pad.append(point(0,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))
      #
      # pin 3: MOSI
      #
      self.shape = add(self.shape,translate(pad_MTA,.1,-.1,0))
      self.pad.append(point(.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'MOSI',14))
      #
      # pin 4: -RESET
      #
      self.shape = add(self.shape,translate(pad_MTA,.05,.1,0))
      self.pad.append(point(.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'-RESET',14))
      #
      # pin 5: SCK
      #
      self.shape = add(self.shape,translate(pad_MTA,-.05,.1,0))
      self.pad.append(point(-.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'SCK',14))
```

```
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.262,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.262,0,0))

class power_65mm(part):
   #
   # CUI PJ1-023-SMT
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: power
      #
      self.shape = cube(.433,.512,-.047,.047,0,0)
      self.pad.append(point(.467,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'P',14))
      #
      # pin 2: ground
      #
      self.shape = add(self.shape,cube(.285,.423,-.189,-.098,0,0))
      self.pad.append(point(.354,-.144,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 3: contact
      #
      self.shape = add(self.shape,cube(.325,.463,.098,.189,0,0))
      self.pad.append(point(.394,.144,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,cube(.108,.246,-.169,-.110,0,0))
      self.shape = add(self.shape,cube(.069,.207,.110,.169,0,0))

pad_stereo_2_5mm = cube(-.03,.03,-.05,.05,0,0)

class stereo_2_5mm(part):
   #
   # CUI SJ1-2533-SMT
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: base
      #
      self.shape = translate(pad_stereo_2_5mm,-.130,-.16,0)
      self.pad.append(point(-.130,-.149,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'base',14))
      #
      # pin 2: tip
      #
      self.shape = add(self.shape,translate(pad_stereo_2_5mm,.197,.15,0))
      self.pad.append(point(.197,.141,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'tip',14))
      #
      # pin 3: middle
      #
      self.shape = add(self.shape,translate(pad_stereo_2_5mm,-.012,-.16,0))
      self.pad.append(point(-.012,-.149,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'middle',14))

pad_Molex = cube(-.0155,.0155,-.0265,.0265,0,0)
pad_Molex_solder = cube(-.055,.055,-.065,.065,0,0)

class Molex_serial(part):
   #
   # Molex 53261-0471
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: Rx
      #
      self.shape = translate(pad_Molex,-.075,.064,0)
      self.pad.append(point(-.075,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Rx',14))
      #
      # pin 2: Tx
      #
      self.shape = add(self.shape,translate(pad_Molex,-.025,.064,0))
      self.pad.append(point(-.025,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Tx',14))
      #
      # pin 3: DTR
      #
      self.shape = add(self.shape,translate(pad_Molex,.025,.064,0))
      self.pad.append(point(.025,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DTR',14))
      #
      # pin 4: GND
      #
      self.shape = add(self.shape,translate(pad_Molex,.075,.064,0))
      self.pad.append(point(.075,.064,0))
```

```
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_Molex_solder,-.16,-.065,0))
      self.shape = add(self.shape,translate(pad_Molex_solder,.16,-.065,0))

#
# switches
#

pad_button_6mm = cube(-.04,.04,-.03,.03,0,0)

class button_6mm(part):
   #
   # Omron 6mm pushbutton
   # B3SN-3112P
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # left 1
      #
      self.shape = translate(pad_button_6mm,-.125,.08,0)
      self.pad.append(point(-.125,.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'L1',14))
      #
      # right 1
      #
      self.shape = add(self.shape,translate(pad_button_6mm,-.125,-.08,0))
      self.pad.append(point(-.125,-.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'R1',14))
      #
      # right 2
      #
      self.shape = add(self.shape,translate(pad_button_6mm,.125,-.08,0))
      self.pad.append(point(.125,-.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'R2',14))
      #
      # left 2
      #
      self.shape = add(self.shape,translate(pad_button_6mm,.125,.08,0))
      self.pad.append(point(.125,.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'L2',14))

#
# crystals and resonators
#

pad_XTAL = cube(-.016,.016,-.077,.077,0,0)

class XTAL(part):
   #
   # Panasonic EFOBM series
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # left
      #
      self.shape = translate(pad_XTAL,-.053,0,0)
      self.pad.append(point(-.053,0,0))
      #
      # ground
      #
      self.shape = add(self.shape,translate(pad_XTAL,0,0,0))
      self.pad.append(point(0,0,0))
      #
      # right
      #
      self.shape = add(self.shape,translate(pad_XTAL,.053,0,0))
      self.pad.append(point(.053,0,0))

#
# diodes, transistors, regulators
#

class D_1206(part):
   #
   # 1206 diode
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # anode
      #
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A',14))
      #
      # cathode
      #
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))
```

```
class LED_1206(part):
   #
   # 1206 LED
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # anode
      #
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A',14))
      #
      # cathode
      #
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))

class phototransistor_1206(part):
   #
   # 1206 phototransistor
   # OPTEK 520,521
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # collector
      #
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))
      #
      # emitter
      #
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'E',14))

pad_SOD_123 = cube(-.02,.02,-.024,.024,0,0)

class D_SOD_123(part):
   #
   # SOD-123 diode
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # anode
      #
      self.shape = translate(pad_SOD_123,-.07,0,0)
      self.pad.append(point(-.07,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A',14))
      #
      # cathode
      #
      self.shape = add(self.shape,translate(pad_SOD_123,.07,0,0))
      self.pad.append(point(.07,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))

pad_SOT23 = cube(-.015,.015,-.02,.02,0,0)

class NMOSFET_SOT23(part):
   #
   # Fairchild NDS355AN
   #
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: gate
      #
      self.shape = translate(pad_SOT23,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 2: source
      #
      self.shape = add(self.shape,translate(pad_SOT23,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'S',14))
      #
      # pin 3: drain
      #
      self.shape = add(self.shape,translate(pad_SOT23,0,.045,0))
      self.pad.append(point(0,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'D',14))

class PMOSFET_SOT23(part):
   #
```

```
   # Fairchild NDS356AP
   #
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: gate
      #
      self.shape = translate(pad_SOT23,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 2: source
      #
      self.shape = add(self.shape,translate(pad_SOT23,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'S',14))
      #
      # pin 3: drain
      #
      self.shape = add(self.shape,translate(pad_SOT23,0,.045,0))
      self.pad.append(point(0,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'D',14))

class regulator_SOT23(part):
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: output
      #
      self.shape = translate(pad_SOT23,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: input
      #
      self.shape = add(self.shape,translate(pad_SOT23,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'in',14))
      #
      # pin 3: ground
      #
      self.shape = add(self.shape,translate(pad_SOT23,0,.045,0))
      self.pad.append(point(0,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'gnd',14))

pad_SOT223 = cube(-.02,.02,-.03,.03,0,0)
pad_SOT223_ground = cube(-.065,.065,-.03,.03,0,0)

class regulator_SOT223(part):
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: input
      #
      self.shape = translate(pad_SOT223,-.09,-.12,0)
      self.pad.append(point(-.09,-.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1I',14))
      #
      # pin 2: ground
      #
      self.shape = add(self.shape,translate(pad_SOT223,0,-.12,0))
      self.pad.append(point(0,-.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 3: output
      #
      self.shape = add(self.shape,translate(pad_SOT223,.09,-.12,0))
      self.pad.append(point(.09,-.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'O',14))
      #
      # pin 4: ground
      #
      self.shape = add(self.shape,translate(pad_SOT223_ground,0,.12,0))
      self.pad.append(point(0,.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))

pad_SM8 = cube(-.035,.035,-.016,.016,0,0)

class H_bridge_SM8(part):
   #
   # Zetex ZXMHC3A01T8
   #
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
```

```
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      d = .13
      #
      # pin 1: G3 (right N gate)
      #
      self.shape = translate(pad_SM8,-d,.09,0)
      self.pad.append(point(-d,.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GRN',14))
      #
      # pin 2: S2 S3 (N source)
      #
      self.shape = add(self.shape,translate(pad_SM8,-d,.03,0))
      self.pad.append(point(-d,.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'SN',14))
      #
      # pin 3: G2 (left N gate)
      #
      self.shape = add(self.shape,translate(pad_SM8,-d,-.03,0))
      self.pad.append(point(-d,-.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GLN',14))
      #
      # pin 4: G1 (left P gate)
      #
      self.shape = add(self.shape,translate(pad_SM8,-d,-.09,0))
      self.pad.append(point(-d,-.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GLP',14))
      #
      # pin 5: D1 D2 (left drain)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,-.09,0))
      self.pad.append(point(d,-.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DL',14))
      #
      # pin 6: S1 S4 (P source)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,-.03,0))
      self.pad.append(point(d,-.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'SP',14))
      #
      # pin 7: D3 D4 (right drain)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,.03,0))
      self.pad.append(point(d,.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DR',14))
      #
      # pin 8: G4 (right N gate)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,.09,0))
      self.pad.append(point(d,.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GRP',14))

#
# ICs
#

pad_SOT23_5 = cube(-.01,.01,-.02,.02,0,0)

class op_amp_SOT23_5(part):
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: output
      #
      self.shape = translate(pad_SOT23_5,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'O',14))
      #
      # pin 2: V-
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,0,-.045,0))
      self.pad.append(point(0,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V-',14))
      #
      # pin 3: I+
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'I+',14))
      #
      # pin 4: I-
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,.0375,.045,0))
      self.pad.append(point(.0375,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'I-',14))
      #
      # pin 5: V+
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,-.0375,.045,0))
      self.pad.append(point(-.0375,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V+',14))

pad_SOICN = cube(-.035,.035,-.015,.015,0,0)

class op_amp_SOICN(part):
```

```
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: A out
      #
      self.shape = translate(pad_SOICN,-.12,.075,0)
      self.pad.append(point(-.12,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1 Ao',14))
      #
      # pin 2: A-
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.025,0))
      self.pad.append(point(-.12,.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A-',14))
      #
      # pin 3: A+
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.025,0))
      self.pad.append(point(-.12,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A+',14))
      #
      # pin 4: V-
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.075,0))
      self.pad.append(point(-.12,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V-',14))
      #
      # pin 5: B+
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.075,0))
      self.pad.append(point(.12,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'B+',14))
      #
      # pin 6: B-
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.025,0))
      self.pad.append(point(.12,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'B-',14))
      #
      # pin 7: B out
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.025,0))
      self.pad.append(point(.12,.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Bo',14))
      #
      # pin 8: V+
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.075,0))
      self.pad.append(point(.12,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V+',14))

pad_SOIC = cube(-.043,.043,-.015,.015,0,0)

class ATtiny45_SOIC(part):
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: PB5/dW/ADC0/-RESET/PCINT5
      #
      self.shape = translate(pad_SOIC,-.14,.075,0)
      self.pad.append(point(-.14,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: PB3/ADC3/-OC1B/CLKI/XTAL1/PCINT3
      #
      self.shape = add(self.shape,translate(pad_SOIC,-.14,.025,0))
      self.pad.append(point(-.14,.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB3',14))
      #
      # pin 3: PB4/ADC2/OC1B/CLKO/XTAL2/PCINT4
      #
      self.shape = add(self.shape,translate(pad_SOIC,-.14,-.025,0))
      self.pad.append(point(-.14,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB4',14))
      #
      # pin 4: GND
      #
      self.shape = add(self.shape,translate(pad_SOIC,-.14,-.075,0))
      self.pad.append(point(-.14,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))
      #
      # pin 5: PB0/MOSI/DI/SDA/AIN0/OC0A/-OC1A/AREF/PCINT0
      #
      self.shape = add(self.shape,translate(pad_SOIC,.14,-.075,0))
      self.pad.append(point(.14,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 6: PB1/MISO/DO/AIN1/OC0B/OC1A/PCINT1
      #
      self.shape = add(self.shape,translate(pad_SOIC,.14,-.025,0))
      self.pad.append(point(.14,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB1',14))
      #
      # pin 7: PB2/SCK/USCK/SCL/ADC1/T0/INT0/PCINT2
      #
      self.shape = add(self.shape,translate(pad_SOIC,.14,.025,0))
      self.pad.append(point(.14,.025,0))
```

```
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB2',14))
      #
      # pin 8: VCC
      #
      self.shape = add(self.shape,translate(pad_SOIC,.14,.075,0))
      self.pad.append(point(.14,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'VCC',14))

class ATtiny44_SOICN(part):
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: VCC
      #
      self.shape = translate(pad_SOICN,-.12,.15,0)
      self.pad.append(point(-.12,.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: PB0/XTAL1/PCINT8
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 3: PB1/XTAL2/PCINT9
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.050,0))
      self.pad.append(point(-.12,.05,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB1',14))
      #
      # pin 4: PB3/dW/-RESET/PCINT11
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,0,0))
      self.pad.append(point(-.12,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB3',14))
      #
      # pin 5: PB2/CKOUT/OC0A/INT0/PCINT10
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.05,0))
      self.pad.append(point(-.12,-.05,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB2',14))
      #
      # pin 6: PA7/ADC7/OC0B/ICP/PCINT7
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.1,0))
      self.pad.append(point(-.12,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA7',14))
      #
      # pin 7: PA6/ADC6/MOSI/SDA/OC1A/PCINT6
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.15,0))
      self.pad.append(point(-.12,-.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA6',14))
      #
      # pin 8: PA5/ADC5/DO/MISO/OC1B/PCINT5
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.15,0))
      self.pad.append(point(.12,-.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA5',14))
      #
      # pin 9: PA4/ADC4/USCK/SCL/T1/PCINT4
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.1,0))
      self.pad.append(point(.12,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA4',14))
      #
      # pin 10: PA3/ADC3/T0/PCINT3
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.05,0))
      self.pad.append(point(.12,-.05,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA3',14))
      #
      # pin 11: PA2/ADC2/AIN1/PCINT2
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,0,0))
      self.pad.append(point(.12,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA2',14))
      #
      # pin 12: PA1/ADC1/AIN0/PCINT1
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.050,0))
      self.pad.append(point(.12,.05,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA1',14))
      #
      # pin 13: PA0/ADC0/AREF/PCINT0
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.1,0))
      self.pad.append(point(.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA0',14))
      #
      # pin 14: GND
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.15,0))
      self.pad.append(point(.12,.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))

pad_TQFP = cube(-.043,.043,-.015,.015,0,0)

class ATmega88_TQFP(part):
```

```
def __init__(self,value=''):
   self.value = value
   self.pad = [point(0,0,0)]
   self.labels = []

   #
   # pin 1: PD3/PCINT19/OC2B/INT1
   #
   self.shape = translate(pad_SOICN,-.12,.15,0)
   self.pad.append(point(-.12,.15,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
   #
   # pin 2: PD4/PCINT20/XCK/T0
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 3: GND
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 4: VCC
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 5: GND
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 6: VCC
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 7: PB6/PCINT6/XTAL1/TOSC1
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 8: PB7/PCINT7/XTAL2/TOSC2
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

   #
   # pin 9: PD5/PCINT21/OC0B/T1
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 10: PD6/PCINT22/OC0A/AIN0
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 11: PD7/PCINT23/AIN1
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 12: PB0/PCINT0/CLKO/ICP1
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 13: PB1/PCINT1/OC1A
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 14: PB2/PCINT2/-SS/OC1B
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 15: PB3/PCINT3/OC2A/MOSI
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 16: PB4/PCINT4/MISO
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
```

```
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

        #
        # pin 17: PB5/SCK/PCINT5
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 18: AVCC
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 19: ADC6
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 20: AREF
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 21: GND
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 22: ADC7
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 23: PC0/ADC0/PCINT8
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 24: PC1/ADC1/PCINT9
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

        #
        # pin 25: PC2/ADC2/PCINT10
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 26: PC2/ADC3/PCINT11
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 27: PC4/ADC4/SDA/PCINT12
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 28: PC5/ADC5/SCL/PCINT13
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 29: PC6/-RESET/PCINT14
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 30: PD0/RXD/PCINT16
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 31: PD1/TXD/PCINT17
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 32: PD2/INT0/PCINT18
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

#
```

```
# graphics
#

class CBA(part):
   def __init__(self,r=.02):
      self.value = ''
      self.pad = [point(0,0,0)]
      self.labels = []
      d = 3*r
      self.shape = circle(0,0,r)
      self.shape = add(self.shape,translate(circle(0,0,r),-d,d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),-d,0,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),-d,-d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),0,-d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),d,-d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),d,0,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),d,d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),0,d,0))

class hello(part):
   #
   # HELLO
   #
   def __init__(self,w,z):
      self.value = ''
      self.pad = [point(0,0,0)]
      self.labels = []
      # H
      self.shape = rectangle(-.01,.01,0,.1)
      self.shape = add(self.shape,rectangle(0,.05,.04,.06))
      self.shape = add(self.shape,rectangle(.04,.06,0,.1))
      # E
      self.shape = add(self.shape,rectangle(.08,.1,0,.1))
      self.shape = add(self.shape,rectangle(.1,.14,0,.02))
      self.shape = add(self.shape,rectangle(.1,.13,.04,.06))
      self.shape = add(self.shape,rectangle(.1,.14,.08,.1))
      # L
      self.shape = add(self.shape,rectangle(.16,.18,0,.1))
      self.shape = add(self.shape,rectangle(.18,.21,0,.02))
      # L
      self.shape = add(self.shape,rectangle(.23,.25,0,.1))
      self.shape = add(self.shape,rectangle(.25,.28,0,.02))
      # 0
      self.shape = add(self.shape,rectangle(.3,.32,0,.1))
      self.shape = add(self.shape,rectangle(.32,.345,0,.02))
      self.shape = add(self.shape,rectangle(.32,.345,.08,.1))
      self.shape = add(self.shape,rectangle(.345,.365,0,.1))

#
# define board
#

x0 = 1
y0 = 1
width = 1.23
height = .7
z = -.005
w = .018
mask = .004

pcb = PCB(x0,y0,width,height,mask)

IC1 = ATtiny45_SOIC('IC1\nt45')
pcb = IC1.add(pcb,x0+.76,y0+.35,z,angle=-90)

R1 = R_1206('R1\n10k')
pcb = R1.add(pcb,IC1.pad[1].x-.09,IC1.y,z,angle=90)

pcb = wire(pcb,w,
   IC1.pad[1],
   R1.pad[2])

pcb = wire(pcb,w,
   IC1.pad[8],
   R1.pad[1])

IC2 = regulator_SOT23('IC2\n5V')
pcb = IC2.add(pcb,R1.x-.22,IC1.y+.08,z,angle=180)

C1 = C_1206('C1\n3.3uF')
pcb = C1.add(pcb,IC2.x+.12,IC2.y,z,angle=90)

pcb = wire(pcb,w,
   IC2.pad[1],
   C1.pad[1])

pcb = wire(pcb,w,
   C1.pad[1],
   R1.pad[1])

J1 = MTA_serial('J1')
pcb = J1.add(pcb,IC2.x-.22,IC1.y,z,angle=-90)

pcb = wire(pcb,w,
   J1.pad[1],
   IC2.pad[3])

pcb = wire(pcb,w,
   J1.pad[1],
   C1.pad[2])
```

```
pcb = wire(pcb,w,
    J1.pad[3],
    point(J1.pad[3].x,J1.pad[3].y+.05,z),
    point(J1.pad[2].x,IC1.pad[7].y+.06,z),
    point(IC1.pad[8].x-.05,IC1.pad[7].y+.18,z),
    IC1.pad[7])

pcb = wire(pcb,w,
    IC2.pad[2],
    point(IC2.pad[2].x,J1.pad[3].y,z),
    point(J1.x-.02,J1.pad[4].y,z),
    J1.pad[4])

J2 = MTA_ICP('J2')
pcb = J2.add(pcb,IC1.x+.31,IC1.y,z,angle=90)

pcb = wire(pcb,w,
    IC1.pad[6],
    point(IC1.pad[6].x,IC1.pad[6].y+.1,z),
    point(J2.pad[1].x-.03,J2.pad[1].y,z),
    J2.pad[1])

pcb = wire(pcb,w,
    J2.pad[2],
    C1.pad[2])

pcb = wire(pcb,w,
    J2.pad[3],
    point(J2.x,J2.pad[5].y,z),
    IC1.pad[5])

pcb = wire(pcb,w,
    J2.pad[4],
    point(J2.pad[4].x,J2.pad[3].y-.05,z),
    point(J2.pad[3].x-.08,R1.pad[2].y,z),
    IC1.pad[1])

pcb = wire(pcb,w,
    IC1.pad[7],
    point(IC1.pad[7].x,IC1.pad[7].y+.18,z),
    point(J2.pad[1].x+.02,J2.pad[1].y+.05,z),
    J2.pad[5])

R2 = R_1206('R2\n4.99k')
pcb = R2.add(pcb,IC2.pad[3].x+.05,J1.pad[1].y-.08,z)

pcb = wire(pcb,w,
    J1.pad[2],
    point(J1.x+.02,J1.pad[1].y-.05,z),
    point(J1.pad[1].x,R2.y,z),
    R2.pad[1])

D1 = D_SOD_123('D1\n4.7V')
pcb = D1.add(pcb,R2.x,R2.y-.1,z)

pcb = wire(pcb,w,
    D1.pad[1],
    point(D1.x,D1.y,z))

pcb = wire(pcb,w,
    D1.pad[2],
    IC1.pad[2])

pcb = wire(pcb,w,
    D1.pad[2],
    R2.pad[2])

pcb = wire(pcb,w,
    J1.pad[1],
    point(D1.x,D1.y-.06,z),
    IC1.pad[4])

H1 = hello(w,z)
pcb = H1.add(pcb,x0+.23,y0+height-.12,z)

#
# assign board and exterior to cad.function for milling
#
# use single-sided FR2
# use 1/64 end-mill
# set tool diameter = .0156
# set xy, z speed = 4
# set # contours = -1
#

cad.function = add(pcb.board,pcb.exterior)

#
# uncomment to export traces
#

#cad.function = pcb.board

#
# uncomment to export exterior
#

#cad.function = pcb.exterior

#
# uncomment to export solder mask
```

```
#

#cad.function = pcb.mask

#
# uncomment to mill out board
#
# use 1/32 end-mill
# set tool diameter = .0312
# set xy, z speed = .5
# set # contours = 1
#

#cad.function = pcb.interior
#z = -.065

#
# define limits and parameters
#

d = 0.05
cad.xmin = x0-d # min x to render
cad.xmax = x0+width+d # max x to render
cad.ymin = y0-d # min y to render
cad.ymax = y0+height+d # max y to render
cad.zmin = z
cad.zmax = .050
dpi = 200 # low resolution for previewing
#dpi = 500 # high resolution for machining
nxy = int(dpi*(cad.xmax-cad.xmin))
cad.nx = int(dpi*(cad.xmax-cad.xmin)) # x points to render
cad.ny = int(dpi*(cad.ymax-cad.ymin)) # y points to render
cad.nz = 1
cad.inches_per_unit = 1.0 # use inch units
cad.view('xy') # 2D view
```

```
#
# hello.light.45.cad
#
# light sensor hello-world
#
# Neil Gershenfeld
# CBA MIT 10/24/07
#
# (c) Massachusetts Institute of Technology 2007
# Permission granted for experimental and personal use;
# license for commercial sale available from MIT.
#

#
# define shapes and transformation
#
# circle(x0, y0, r)
# cylinder(x0, y0, z0, z1, r)
# cone(x0, y0, z0, z1, r0)
# sphere(x0, y0, z0, r)
# torus(x0, y0, z0, r0, r1)
# rectangle(x0, x1, y0, y1)
# cube(x0, x1, y0, y1, z0, z1)
# triangle(x0, y0, x1, y1, x2, y2) (points in clockwise order)
# pyramid(x0, x1, y0, y1, z0, z1)
# function(Z_of_XY)
# functions(upper_Z_of_XY,lower_Z_of_XY)
# add(part1, part2)
# subtract(part1, part2)
# intersect(part1, part2)
# move(part,dx,dy)
# translate(part,dx,dy,dz)
# rotate(part, angle)
# rotate_x(part, angle)
# rotate_y(part, angle)
# rotate_z(part, angle)
# rotate_90(part)
# rotate_180(part)
# rotate_270(part)
# reflect_x(part)
# reflect_y(part)
# reflect_z(part)
# reflect_xy(part)
# reflect_xz(part)
# reflect_yz(part)
# scale_x(part, x0, sx)
# scale_y(part, y0, sy)
# scale_z(part, z0, sz)
# scale_xy(part, x0, y0, sxy)
# scale_xyz(part, x0, y0, z0, sxyz)
# coscale_x_y(part, x0, y0, y1, angle0, angle1, amplitude, offset)
# coscale_x_z(part, x0, z0, z1, angle0, angle1, amplitude, offset)
# coscale_xy_z(part, x0, y0, z0, z1, angle0, angle1, amplitude, offset)
# taper_x_y(part, x0, y0, y1, s0, s1)
# taper_x_z(part, x0, z0, z1, s0, s1)
# taper_xy_z(part, x0, y0, z0, z1, s0, s1)
# shear_x_y(part, y0, y1, dx0, dx1)
# shear_x_z(part, z0, z1, dx0, dx1)
# (more to come)

def circle(x0, y0, r):
   part = "(((X-x0)**2 + (Y-y0)**2) <= r**2)"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'r',str(r))
   return part

def cylinder(x0, y0, z0, z1, r):
   part = "(((X-x0)**2 + (Y-y0)**2 <= r**2) & (Z >= z0) & (Z <= z1))"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'r',str(r))
   return part

def cone(x0, y0, z0, z1, r0):
   part = cylinder(x0, y0, z0, z1, r0)
   part = taper_xy_z(part, x0, y0, z0, z1, 1.0, 0.0)
   return part

def sphere(x0, y0, z0, r):
   part = "(((X-x0)**2 + (Y-y0)**2 + (Z-z0)**2) <= r**2)"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'r',str(r))
   return part

def torus(x0, y0, z0, r0, r1):
   part = "(((r0 - sqrt((X-x0)**2 + (Y-y0)**2))**2 + (Z-z0)**2) <= r1**2)"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'r0',str(r0))
   part = replace(part,'r1',str(r1))
   return part

def rectangle(x0, x1, y0, y1):
   part = "((X >= x0) & (X <= x1) & (Y >= y0) & (Y <= y1))"
   part = replace(part,'x0',str(x0))
```

```
   part = replace(part,'x1',str(x1))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   return part

def cube(x0, x1, y0, y1, z0, z1):
   part = "((X >= x0) & (X <= x1) & (Y >= y0) & (Y <= y1) & (Z >= z0) & (Z <= z1))"
   part = replace(part,'x0',str(x0))
   part = replace(part,'x1',str(x1))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   return part

def triangle(x0, y0, x1, y1, x2, y2): # points in clockwise order
   part = "((((y1-y0)*(X-x0)-(x1-x0)*(Y-y0)) >= 0) & (((y2-y1)*(X-x1)-(x2-x1)*(Y-y1)) >= 0) & (((y0-y2)*(X-x2)-(x0-x2)*(Y-y2)) >= 0))"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'x1',str(x1))
   part = replace(part,'y1',str(y1))
   part = replace(part,'x2',str(x2))
   part = replace(part,'y2',str(y2))
   return part

def pyramid(x0, x1, y0, y1, z0, z1):
   part = cube(x0, x1, y0, y1, z0, z1)
   part = taper_xy_z(part, (x0+x1)/2., (y0+y1)/2., z0, z1, 1.0, 0.0)
   return part

def function(Z_of_XY):
   part = '(Z <= '+Z_of_XY+')'
   return part

def functions(upper_Z_of_XY,lower_Z_of_XY):
   part = '(Z <= '+upper_Z_of_XY+') & (Z >= '+lower_Z_of_XY+')'
   return part

def add(part1, part2):
   part = "part1 | part2"
   part = replace(part,'part1',part1)
   part = replace(part,'part2',part2)
   return part

def subtract(part1, part2):
   part = "(part1) & ~(part2)"
   part = replace(part,'part1',part1)
   part = replace(part,'part2',part2)
   return part

def intersect(part1, part2):
   part = "(part1) & (part2)"
   part = replace(part,'part1',part1)
   part = replace(part,'part2',part2)
   return part

def move(part,dx,dy):
   part = replace(part,'X','(X-'+str(dx)+')')
   part = replace(part,'Y','(Y-'+str(dy)+')')
   return part

def translate(part,dx,dy,dz):
   part = replace(part,'X','(X-'+str(dx)+')')
   part = replace(part,'Y','(Y-'+str(dy)+')')
   part = replace(part,'Z','(Z-'+str(dz)+')')
   return part

def rotate(part, angle):
   angle = angle*pi/180
   part = replace(part,'X','(cos(angle)*X+sin(angle)*y)')
   part = replace(part,'Y','(-sin(angle)*X+cos(angle)*y)')
   part = replace(part,'y','Y')
   part = replace(part,'angle',str(angle))
   return part

def rotate_x(part, angle):
   angle = angle*pi/180
   part = replace(part,'Y','(cos(angle)*Y+sin(angle)*z)')
   part = replace(part,'Z','(-sin(angle)*Y+cos(angle)*z)')
   part = replace(part,'z','Z')
   part = replace(part,'angle',str(angle))
   return part

def rotate_y(part, angle):
   angle = angle*pi/180
   part = replace(part,'X','(cos(angle)*X+sin(angle)*z)')
   part = replace(part,'Z','(-sin(angle)*X+cos(angle)*z)')
   part = replace(part,'z','Z')
   part = replace(part,'angle',str(angle))
   return part

def rotate_z(part, angle):
   angle = angle*pi/180
   part = replace(part,'X','(cos(angle)*X+sin(angle)*y)')
   part = replace(part,'Y','(-sin(angle)*X+cos(angle)*y)')
   part = replace(part,'y','Y')
   part = replace(part,'angle',str(angle))
   return part

def rotate_90(part):
   part = reflect_xy(part)
```

```
   part = reflect_y(part)
   return part

def rotate_180(part):
   part = reflect_xy(part)
   part = reflect_y(part)
   part = reflect_xy(part)
   part = reflect_y(part)
   return part

def rotate_270(part):
   part = reflect_xy(part)
   part = reflect_y(part)
   part = reflect_xy(part)
   part = reflect_y(part)
   part = reflect_xy(part)
   part = reflect_y(part)
   return part

def reflect_x(part):
   part = replace(part,'X','-X')
   return part

def reflect_y(part):
   part = replace(part,'Y','-Y')
   return part

def reflect_z(part):
   part = replace(part,'Z','-Z')
   return part

def reflect_xy(part):
   part = replace(part,'X','temp')
   part = replace(part,'Y','X')
   part = replace(part,'temp','Y')
   return part

def reflect_xz(part):
   part = replace(part,'X','temp')
   part = replace(part,'Z','X')
   part = replace(part,'temp','Z')
   return part

def reflect_yz(part):
   part = replace(part,'Y','temp')
   part = replace(part,'Z','Y')
   part = replace(part,'temp','Z')
   return part

def scale_x(part, x0, sx):
   part = replace(part,'X','(x0 + (X-x0)/sx)')
   part = replace(part,'x0',str(x0))
   part = replace(part,'sx',str(sx))
   return part

def scale_y(part, y0, sy):
   part = replace(part,'Y','(y0 + (Y-y0)/sy)')
   part = replace(part,'y0',str(y0))
   part = replace(part,'sy',str(sy))
   return part

def scale_z(part, z0, sz):
   part = replace(part,'Z','(z0 + (Z-z0)/sz)')
   part = replace(part,'z0',str(z0))
   part = replace(part,'sz',str(sz))
   return part

def scale_xy(part, x0, y0, sxy):
   part = replace(part,'X','(x0 + (X-x0)/sxy)')
   part = replace(part,'Y','(y0 + (Y-y0)/sxy)')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'sxy',str(sxy))
   return part

def scale_xyz(part, x0, y0, z0, sxyz):
   part = replace(part,'X','(x0 + (X-x0)/sxyz)')
   part = replace(part,'Y','(y0 + (Y-y0)/sxyz)')
   part = replace(part,'Z','(z0 + (Z-z0)/sxyz)')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'sxyz',str(sxyz))
   return part

def coscale_x_y(part, x0, y0, y1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
   part = replace(part,'X','(x0 + (X-x0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Y-y0)/(y1-y0))))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

def coscale_x_z(part, x0, z0, z1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
```

```
   part = replace(part,'X','(x0 + (X-x0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0))))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

def coscale_xy_z(part, x0, y0, z0, z1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
   part = replace(part,'X','(x0 + (X-x0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0))))')
   part = replace(part,'Y','(y0 + (Y-y0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0))))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

def taper_x_y(part, x0, y0, y1, s0, s1):
   part = replace(part,'X','(x0 + (X-x0)*(y1-y0)/(s1*(Y-y0) + s0*(y1-Y)))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'s0',str(s0))
   part = replace(part,'s1',str(s1))
   return part

def taper_x_z(part, x0, z0, z1, s0, s1):
   part = replace(part,'X','(x0 + (X-x0)*(z1-z0)/(s1*(Z-z0) + s0*(z1-Z)))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'s0',str(s0))
   part = replace(part,'s1',str(s1))
   return part

def taper_xy_z(part, x0, y0, z0, z1, s0, s1):
   part = replace(part,'X','(x0 + (X-x0)*(z1-z0)/(s1*(Z-z0) + s0*(z1-Z)))')
   part = replace(part,'Y','(y0 + (Y-y0)*(z1-z0)/(s1*(Z-z0) + s0*(z1-Z)))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'s0',str(s0))
   part = replace(part,'s1',str(s1))
   return part

def shear_x_y(part, y0, y1, dx0, dx1):
   part = replace(part,'X','(X - dx0 - (dx1-dx0)*(Y-y0)/(y1-y0))')
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'dx0',str(dx0))
   part = replace(part,'dx1',str(dx1))
   return part

def shear_x_z(part, z0, z1, dx0, dx1):
   part = replace(part,'X','(X - dx0 - (dx1-dx0)*(Z-z0)/(z1-z0))')
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'dx0',str(dx0))
   part = replace(part,'dx1',str(dx1))
   return part

def coshear_x_z(part, z0, z1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
   part = replace(part,'X','(X - offset - amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0)))')
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

#
# PCB classes and definitions
#

cad.labels = []

class PCB:
   def __init__(self,x0,y0,width,height,mask):
      self.board = "False"
      self.interior = rectangle(x0,x0+width,y0,y0+height)
      self.exterior = subtract("True",rectangle(x0,x0+width,y0,y0+height))
      self.mask = "False"
   def add(self,part):
      self.board = add(self.board,part)
      self.mask = add(self.mask,move(part,-mask,mask))
      self.mask = add(self.mask,move(part,-mask,-mask))
      self.mask = add(self.mask,move(part,mask,mask))
      self.mask = add(self.mask,move(part,mask,-mask))
      return self
```

```
class part:
   def add(self,pcb,x,y,z=0,angle=0):
      self.x = x
      self.y = y
      self.z = z
      self.angle = angle
      if (angle == 90):
         self.shape = rotate_90(self.shape)
      elif (angle == 180):
         self.shape = rotate_180(self.shape)
      elif ((angle == 270) | (angle == -90)):
         self.shape = rotate_270(self.shape)
      angle = pi*angle/180
      self.shape = translate(self.shape,x,y,z)
      for i in range(len(self.pad)):
         xnew = cos(angle)*self.pad[i].x + sin(angle)*self.pad[i].y
         ynew = -sin(angle)*self.pad[i].x + cos(angle)*self.pad[i].y
           self.pad[i].x = x + xnew
           self.pad[i].y = y + ynew
         self.pad[i].z += z
      cad.labels.append(cad_text(x,y,z,self.value,size=14))
      for i in range(len(self.labels)):
         xnew = cos(angle)*self.labels[i].x + sin(angle)*self.labels[i].y
         ynew = -sin(angle)*self.labels[i].x + cos(angle)*self.labels[i].y
           self.labels[i].x = x + xnew
           self.labels[i].y = y + ynew
         self.labels[i].z += z
           cad.labels.append(self.labels[i])
      pcb = pcb.add(self.shape)
      return pcb

def wire(pcb,width,*points):
   for i in range(1,len(points)):
      x0 = points[i-1].x
      y0 = points[i-1].y
      z0 = points[i-1].z
      x1 = points[i].x
      y1 = points[i].y
      z1 = points[i].z
      if (x0 < x1):
         pcb.board = add(pcb.board,cube(x0-width/2,x1+width/2,y0-width/2,y0+width/2,z0,z0))
      elif (x1 < x0):
         pcb.board = add(pcb.board,cube(x1-width/2,x0+width/2,y0-width/2,y0+width/2,z0,z0))
      if (y0 < y1):
         pcb.board = add(pcb.board,cube(x1-width/2,x1+width/2,y0-width/2,y1+width/2,z0,z0))
      elif (y1 < y0):
         pcb.board = add(pcb.board,cube(x1-width/2,x1+width/2,y1-width/2,y0+width/2,z0,z0))
   return pcb

#
# PCB library
#

#
# discretes
#

pad_0402 = cube(-.0175,.0175,-.014,.014,0,0)

class R_0402(part):
   #
   # 0402 resistor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_0402,-.0265,0,0)
      self.pad.append(point(-.0265,0,0))
      self.shape = add(self.shape,translate(pad_0402,.0265,0,0))
      self.pad.append(point(.0265,0,0))

pad_1206 = cube(-.032,.032,-.034,.034,0,0)

class R_1206(part):
   #
   # 1206 resistor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.06,0,0))
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.06,0,0))

class C_1206(part):
   #
   # 1206 capacitor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.06,0,0))
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.06,0,0))
```

```
pad_1210 = cube(-.032,.032,-.048,.048,0,0)

class L_1210(part):
   #
   # 1210 inductor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_1210,-.06,0,0)
      self.pad.append(point(-.06,0,0))
      self.shape = add(self.shape,translate(pad_1210,.06,0,0))
      self.pad.append(point(.06,0,0))

pad_choke = cube(-.06,.06,-.06,.06,0,0)

class choke(part):
   #
   # Panasonic ELLCTV
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_choke,-.177,-.177,0)
      self.pad.append(point(-.177,-.177,0))
      self.shape = add(self.shape,translate(pad_choke,.177,.177,0))
      self.pad.append(point(.177,.177,0))

#
# connectors
#

pad_MTA = cube(-.021,.021,-.041,.041,0,0)
pad_MTA_solder = cube(-.071,.071,-.041,.041,0,0)

class MTA_2(part):
   #
   # AMP 1445121-2
   # MTA .050 SMT 2-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1
      #
      self.shape = translate(pad_MTA,-.025,-.1,0)
      self.pad.append(point(-.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,.1,0))
      self.pad.append(point(.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'2',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.187,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.187,0,0))

class MTA_power(part):
   #
   # AMP 1445121-2
   # MTA .050 SMT 2-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: Gnd
      #
      self.shape = translate(pad_MTA,-.025,-.1,0)
      self.pad.append(point(-.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: Vcc
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,.1,0))
      self.pad.append(point(.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Vcc',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.187,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.187,0,0))

class MTA_i0(part):
   #
   # AMP 1445121-3
   # MTA .050 SMT 3-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
```

```
      # pin 1: GND
      #
      self.shape = translate(pad_MTA,.05,.1,0)
      self.pad.append(point(.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1\nGND',14))
      #
      # pin 2: power
      #
      self.shape = add(self.shape,translate(pad_MTA,-.05,.1,0))
      self.pad.append(point(-.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V',14))
      #
      # pin 3: data
      #
      self.shape = add(self.shape,translate(pad_MTA,0,-.1,0))
      self.pad.append(point(0,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'data',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.212,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.212,0,0))

class MTA_serial(part):
   #
   # AMP 1445121-4
   # MTA .050 SMT 4-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: Gnd
      #
      self.shape = translate(pad_MTA,-.075,-.1,0)
      self.pad.append(point(-.075,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: Tx
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,-.1,0))
      self.pad.append(point(.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Tx',14))
      #
      # pin 3: Rx
      #
      self.shape = add(self.shape,translate(pad_MTA,.075,.1,0))
      self.pad.append(point(.075,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Rx',14))
      #
      # pin 4: DTR
      #
      self.shape = add(self.shape,translate(pad_MTA,-.025,.1,0))
      self.pad.append(point(-.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DTR',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.237,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.237,0,0))

class MTA_ICP(part):
   #
   # AMP 1445121-5
   # MTA .050 SMT 4-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: MISO
      #
      self.shape = translate(pad_MTA,-.1,-.1,0)
      self.pad.append(point(-.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'MISO',14))
      #
      # pin 2: GND
      #
      self.shape = add(self.shape,translate(pad_MTA,0,-.1,0))
      self.pad.append(point(0,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))
      #
      # pin 3: MOSI
      #
      self.shape = add(self.shape,translate(pad_MTA,.1,-.1,0))
      self.pad.append(point(.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'MOSI',14))
      #
      # pin 4: -RESET
      #
      self.shape = add(self.shape,translate(pad_MTA,.05,.1,0))
      self.pad.append(point(.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'-RESET',14))
      #
      # pin 5: SCK
      #
      self.shape = add(self.shape,translate(pad_MTA,-.05,.1,0))
      self.pad.append(point(-.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'SCK',14))
```

```
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.262,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.262,0,0))

class power_65mm(part):
   #
   # CUI PJ1-023-SMT
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: power
      #
      self.shape = cube(.433,.512,-.047,.047,0,0)
      self.pad.append(point(.467,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'P',14))
      #
      # pin 2: ground
      #
      self.shape = add(self.shape,cube(.285,.423,-.189,-.098,0,0))
      self.pad.append(point(.354,-.144,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 3: contact
      #
      self.shape = add(self.shape,cube(.325,.463,.098,.189,0,0))
      self.pad.append(point(.394,.144,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,cube(.108,.246,-.169,-.110,0,0))
      self.shape = add(self.shape,cube(.069,.207,.110,.169,0,0))

pad_stereo_2_5mm = cube(-.03,.03,-.05,.05,0,0)

class stereo_2_5mm(part):
   #
   # CUI SJ1-2533-SMT
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: base
      #
      self.shape = translate(pad_stereo_2_5mm,-.130,-.16,0)
      self.pad.append(point(-.130,-.149,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'base',14))
      #
      # pin 2: tip
      #
      self.shape = add(self.shape,translate(pad_stereo_2_5mm,.197,.15,0))
      self.pad.append(point(.197,.141,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'tip',14))
      #
      # pin 3: middle
      #
      self.shape = add(self.shape,translate(pad_stereo_2_5mm,-.012,-.16,0))
      self.pad.append(point(-.012,-.149,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'middle',14))

pad_Molex = cube(-.0155,.0155,-.0265,.0265,0,0)
pad_Molex_solder = cube(-.055,.055,-.065,.065,0,0)

class Molex_serial(part):
   #
   # Molex 53261-0471
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: Rx
      #
      self.shape = translate(pad_Molex,-.075,.064,0)
      self.pad.append(point(-.075,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Rx',14))
      #
      # pin 2: Tx
      #
      self.shape = add(self.shape,translate(pad_Molex,-.025,.064,0))
      self.pad.append(point(-.025,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Tx',14))
      #
      # pin 3: DTR
      #
      self.shape = add(self.shape,translate(pad_Molex,.025,.064,0))
      self.pad.append(point(.025,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DTR',14))
      #
      # pin 4: GND
      #
      self.shape = add(self.shape,translate(pad_Molex,.075,.064,0))
      self.pad.append(point(.075,.064,0))
```

```
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_Molex_solder,-.16,-.065,0))
      self.shape = add(self.shape,translate(pad_Molex_solder,.16,-.065,0))

#
# switches
#

pad_button_6mm = cube(-.04,.04,-.03,.03,0,0)

class button_6mm(part):
   #
   # Omron 6mm pushbutton
   # B3SN-3112P
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # left 1
      #
      self.shape = translate(pad_button_6mm,-.125,.08,0)
      self.pad.append(point(-.125,.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'L1',14))
      #
      # right 1
      #
      self.shape = add(self.shape,translate(pad_button_6mm,-.125,-.08,0))
      self.pad.append(point(-.125,-.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'R1',14))
      #
      # right 2
      #
      self.shape = add(self.shape,translate(pad_button_6mm,.125,-.08,0))
      self.pad.append(point(.125,-.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'R2',14))
      #
      # left 2
      #
      self.shape = add(self.shape,translate(pad_button_6mm,.125,.08,0))
      self.pad.append(point(.125,.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'L2',14))

#
# crystals and resonators
#

pad_XTAL = cube(-.016,.016,-.077,.077,0,0)

class XTAL(part):
   #
   # Panasonic EFOBM series
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # left
      #
      self.shape = translate(pad_XTAL,-.053,0,0)
      self.pad.append(point(-.053,0,0))
      #
      # ground
      #
      self.shape = add(self.shape,translate(pad_XTAL,0,0,0))
      self.pad.append(point(0,0,0))
      #
      # right
      #
      self.shape = add(self.shape,translate(pad_XTAL,.053,0,0))
      self.pad.append(point(.053,0,0))

#
# diodes, transistors, regulators
#

class D_1206(part):
   #
   # 1206 diode
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # anode
      #
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A',14))
      #
      # cathode
      #
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))
```

```
class LED_1206(part):
   #
   # 1206 LED
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # anode
      #
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A',14))
      #
      # cathode
      #
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))

class phototransistor_1206(part):
   #
   # 1206 phototransistor
   # OPTEK 520,521,522
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # collector
      #
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))
      #
      # emitter
      #
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'E',14))

pad_SOD_123 = cube(-.02,.02,-.024,.024,0,0)

class D_SOD_123(part):
   #
   # SOD-123 diode
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # anode
      #
      self.shape = translate(pad_SOD_123,-.07,0,0)
      self.pad.append(point(-.07,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A',14))
      #
      # cathode
      #
      self.shape = add(self.shape,translate(pad_SOD_123,.07,0,0))
      self.pad.append(point(.07,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))

pad_SOT23 = cube(-.015,.015,-.02,.02,0,0)

class NMOSFET_SOT23(part):
   #
   # Fairchild NDS355AN
   #
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: gate
      #
      self.shape = translate(pad_SOT23,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 2: source
      #
      self.shape = add(self.shape,translate(pad_SOT23,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'S',14))
      #
      # pin 3: drain
      #
      self.shape = add(self.shape,translate(pad_SOT23,0,.045,0))
      self.pad.append(point(0,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'D',14))

class PMOSFET_SOT23(part):
   #
```

```
   # Fairchild NDS356AP
   #
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: gate
      #
      self.shape = translate(pad_SOT23,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 2: source
      #
      self.shape = add(self.shape,translate(pad_SOT23,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'S',14))
      #
      # pin 3: drain
      #
      self.shape = add(self.shape,translate(pad_SOT23,0,.045,0))
      self.pad.append(point(0,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'D',14))

class regulator_SOT23(part):
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: output
      #
      self.shape = translate(pad_SOT23,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: input
      #
      self.shape = add(self.shape,translate(pad_SOT23,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'in',14))
      #
      # pin 3: ground
      #
      self.shape = add(self.shape,translate(pad_SOT23,0,.045,0))
      self.pad.append(point(0,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'gnd',14))

pad_SOT223 = cube(-.02,.02,-.03,.03,0,0)
pad_SOT223_ground = cube(-.065,.065,-.03,.03,0,0)

class regulator_SOT223(part):
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: input
      #
      self.shape = translate(pad_SOT223,-.09,-.12,0)
      self.pad.append(point(-.09,-.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1I',14))
      #
      # pin 2: ground
      #
      self.shape = add(self.shape,translate(pad_SOT223,0,-.12,0))
      self.pad.append(point(0,-.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 3: output
      #
      self.shape = add(self.shape,translate(pad_SOT223,.09,-.12,0))
      self.pad.append(point(.09,-.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'O',14))
      #
      # pin 4: ground
      #
      self.shape = add(self.shape,translate(pad_SOT223_ground,0,.12,0))
      self.pad.append(point(0,.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))

pad_SM8 = cube(-.035,.035,-.016,.016,0,0)

class H_bridge_SM8(part):
   #
   # Zetex ZXMHC3A01T8
   #
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
```

```
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      d = .13
      #
      # pin 1: G3 (right N gate)
      #
      self.shape = translate(pad_SM8,-d,.09,0)
      self.pad.append(point(-d,.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GRN',14))
      #
      # pin 2: S2 S3 (N source)
      #
      self.shape = add(self.shape,translate(pad_SM8,-d,.03,0))
      self.pad.append(point(-d,.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'SN',14))
      #
      # pin 3: G2 (left N gate)
      #
      self.shape = add(self.shape,translate(pad_SM8,-d,-.03,0))
      self.pad.append(point(-d,-.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GLN',14))
      #
      # pin 4: G1 (left P gate)
      #
      self.shape = add(self.shape,translate(pad_SM8,-d,-.09,0))
      self.pad.append(point(-d,-.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GLP',14))
      #
      # pin 5: D1 D2 (left drain)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,-.09,0))
      self.pad.append(point(d,-.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DL',14))
      #
      # pin 6: S1 S4 (P source)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,-.03,0))
      self.pad.append(point(d,-.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'SP',14))
      #
      # pin 7: D3 D4 (right drain)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,.03,0))
      self.pad.append(point(d,.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DR',14))
      #
      # pin 8: G4 (right N gate)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,.09,0))
      self.pad.append(point(d,.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GRP',14))

#
# ICs
#

pad_SOT23_5 = cube(-.01,.01,-.02,.02,0,0)

class op_amp_SOT23_5(part):
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: output
      #
      self.shape = translate(pad_SOT23_5,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'O',14))
      #
      # pin 2: V-
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,0,-.045,0))
      self.pad.append(point(0,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V-',14))
      #
      # pin 3: I+
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'I+',14))
      #
      # pin 4: I-
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,.0375,.045,0))
      self.pad.append(point(.0375,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'I-',14))
      #
      # pin 5: V+
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,-.0375,.045,0))
      self.pad.append(point(-.0375,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V+',14))

pad_SOICN = cube(-.035,.035,-.015,.015,0,0)

class op_amp_SOICN(part):
```

```
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: A out
      #
      self.shape = translate(pad_SOICN,-.12,.075,0)
      self.pad.append(point(-.12,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1 Ao',14))
      #
      # pin 2: A-
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.025,0))
      self.pad.append(point(-.12,.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A-',14))
      #
      # pin 3: A+
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.025,0))
      self.pad.append(point(-.12,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A+',14))
      #
      # pin 4: V-
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.075,0))
      self.pad.append(point(-.12,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V-',14))
      #
      # pin 5: B+
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.075,0))
      self.pad.append(point(.12,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'B+',14))
      #
      # pin 6: B-
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.025,0))
      self.pad.append(point(.12,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'B-',14))
      #
      # pin 7: B out
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.025,0))
      self.pad.append(point(.12,.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Bo',14))
      #
      # pin 8: V+
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.075,0))
      self.pad.append(point(.12,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V+',14))

pad_SOIC = cube(-.043,.043,-.015,.015,0,0)

class ATtiny45_SOIC(part):
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: PB5/dW/ADC0/-RESET/PCINT5
      #
      self.shape = translate(pad_SOIC,-.14,.075,0)
      self.pad.append(point(-.14,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: PB3/ADC3/-OC1B/CLKI/XTAL1/PCINT3
      #
      self.shape = add(self.shape,translate(pad_SOIC,-.14,.025,0))
      self.pad.append(point(-.14,.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB3',14))
      #
      # pin 3: PB4/ADC2/OC1B/CLKO/XTAL2/PCINT4
      #
      self.shape = add(self.shape,translate(pad_SOIC,-.14,-.025,0))
      self.pad.append(point(-.14,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB4',14))
      #
      # pin 4: GND
      #
      self.shape = add(self.shape,translate(pad_SOIC,-.14,-.075,0))
      self.pad.append(point(-.14,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))
      #
      # pin 5: PB0/MOSI/DI/SDA/AIN0/OC0A/-OC1A/AREF/PCINT0
      #
      self.shape = add(self.shape,translate(pad_SOIC,.14,-.075,0))
      self.pad.append(point(.14,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 6: PB1/MISO/DO/AIN1/OC0B/OC1A/PCINT1
      #
      self.shape = add(self.shape,translate(pad_SOIC,.14,-.025,0))
      self.pad.append(point(.14,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB1',14))
      #
      # pin 7: PB2/SCK/USCK/SCL/ADC1/T0/INT0/PCINT2
      #
      self.shape = add(self.shape,translate(pad_SOIC,.14,.025,0))
      self.pad.append(point(.14,.025,0))
```

```
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB2',14))
      #
      # pin 8: VCC
      #
      self.shape = add(self.shape,translate(pad_SOIC,.14,.075,0))
      self.pad.append(point(.14,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'VCC',14))

class ATtiny44_SOICN(part):
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: VCC
      #
      self.shape = translate(pad_SOICN,-.12,.15,0)
      self.pad.append(point(-.12,.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: PB0/XTAL1/PCINT8
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 3: PB1/XTAL2/PCINT9
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.050,0))
      self.pad.append(point(-.12,.05,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB1',14))
      #
      # pin 4: PB3/dW/-RESET/PCINT11
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,0,0))
      self.pad.append(point(-.12,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB3',14))
      #
      # pin 5: PB2/CKOUT/OC0A/INT0/PCINT10
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.05,0))
      self.pad.append(point(-.12,-.05,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB2',14))
      #
      # pin 6: PA7/ADC7/OC0B/ICP/PCINT7
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.1,0))
      self.pad.append(point(-.12,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA7',14))
      #
      # pin 7: PA6/ADC6/MOSI/SDA/OC1A/PCINT6
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.15,0))
      self.pad.append(point(-.12,-.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA6',14))
      #
      # pin 8: PA5/ADC5/DO/MISO/OC1B/PCINT5
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.15,0))
      self.pad.append(point(.12,-.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA5',14))
      #
      # pin 9: PA4/ADC4/USCK/SCL/T1/PCINT4
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.1,0))
      self.pad.append(point(.12,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA4',14))
      #
      # pin 10: PA3/ADC3/T0/PCINT3
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.05,0))
      self.pad.append(point(.12,-.05,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA3',14))
      #
      # pin 11: PA2/ADC2/AIN1/PCINT2
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,0,0))
      self.pad.append(point(.12,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA2',14))
      #
      # pin 12: PA1/ADC1/AIN0/PCINT1
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.050,0))
      self.pad.append(point(.12,.05,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA1',14))
      #
      # pin 13: PA0/ADC0/AREF/PCINT0
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.1,0))
      self.pad.append(point(.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA0',14))
      #
      # pin 14: GND
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.15,0))
      self.pad.append(point(.12,.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))

pad_TQFP = cube(-.043,.043,-.015,.015,0,0)

class ATmega88_TQFP(part):
```

```
def __init__(self,value=''):
   self.value = value
   self.pad = [point(0,0,0)]
   self.labels = []

   #
   # pin 1: PD3/PCINT19/OC2B/INT1
   #
   self.shape = translate(pad_SOICN,-.12,.15,0)
   self.pad.append(point(-.12,.15,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
   #
   # pin 2: PD4/PCINT20/XCK/T0
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 3: GND
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 4: VCC
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 5: GND
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 6: VCC
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 7: PB6/PCINT6/XTAL1/TOSC1
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 8: PB7/PCINT7/XTAL2/TOSC2
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

   #
   # pin 9: PD5/PCINT21/OC0B/T1
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 10: PD6/PCINT22/OC0A/AIN0
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 11: PD7/PCINT23/AIN1
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 12: PB0/PCINT0/CLKO/ICP1
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 13: PB1/PCINT1/OC1A
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 14: PB2/PCINT2/-SS/OC1B
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 15: PB3/PCINT3/OC2A/MOSI
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 16: PB4/PCINT4/MISO
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
```

```
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

        #
        # pin 17: PB5/SCK/PCINT5
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 18: AVCC
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 19: ADC6
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 20: AREF
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 21: GND
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 22: ADC7
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 23: PC0/ADC0/PCINT8
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 24: PC1/ADC1/PCINT9
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

        #
        # pin 25: PC2/ADC2/PCINT10
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 26: PC2/ADC3/PCINT11
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 27: PC4/ADC4/SDA/PCINT12
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 28: PC5/ADC5/SCL/PCINT13
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 29: PC6/-RESET/PCINT14
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 30: PD0/RXD/PCINT16
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 31: PD1/TXD/PCINT17
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 32: PD2/INT0/PCINT18
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

#
```

```
# graphics
#

class CBA(part):
   def __init__(self,r=.02):
      self.value = ''
      self.pad = [point(0,0,0)]
      self.labels = []
      d = 3*r
      self.shape = circle(0,0,r)
      self.shape = add(self.shape,translate(circle(0,0,r),-d,d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),-d,0,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),-d,-d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),0,-d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),d,-d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),d,0,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),d,d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),0,d,0))

class hello(part):
   #
   # HELLO
   #
   def __init__(self,w,z):
      self.value = ''
      self.pad = [point(0,0,0)]
      self.labels = []
      # H
      self.shape = rectangle(-.01,.01,0,.1)
      self.shape = add(self.shape,rectangle(0,.05,.04,.06))
      self.shape = add(self.shape,rectangle(.04,.06,0,.1))
      # E
      self.shape = add(self.shape,rectangle(.08,.1,0,.1))
      self.shape = add(self.shape,rectangle(.1,.14,0,.02))
      self.shape = add(self.shape,rectangle(.1,.13,.04,.06))
      self.shape = add(self.shape,rectangle(.1,.14,.08,.1))
      # L
      self.shape = add(self.shape,rectangle(.16,.18,0,.1))
      self.shape = add(self.shape,rectangle(.18,.21,0,.02))
      # L
      self.shape = add(self.shape,rectangle(.23,.25,0,.1))
      self.shape = add(self.shape,rectangle(.25,.28,0,.02))
      # 0
      self.shape = add(self.shape,rectangle(.3,.32,0,.1))
      self.shape = add(self.shape,rectangle(.32,.345,0,.02))
      self.shape = add(self.shape,rectangle(.32,.345,.08,.1))
      self.shape = add(self.shape,rectangle(.345,.365,0,.1))

#
# define board
#

x0 = 1
y0 = 1
width = 1.36
height = .7
z = -.005
w = .018
mask = .004

pcb = PCB(x0,y0,width,height,mask)

IC1 = ATtiny45_SOIC('IC1\nt45')
pcb = IC1.add(pcb,x0+.92,y0+.35,z,angle=-90)

R1 = R_1206('R1\n10k')
pcb = R1.add(pcb,IC1.pad[1].x-.27,IC1.y,z,angle=90)

pcb = wire(pcb,w,
   IC1.pad[1],
   R1.pad[2])

pcb = wire(pcb,w,
   IC1.pad[8],
   R1.pad[1])

IC2 = regulator_SOT23('IC2\n5V')
pcb = IC2.add(pcb,R1.x-.155,R1.y,z,angle=90)

pcb = wire(pcb,w,
   IC2.pad[3],
   IC1.pad[4])

C1 = C_1206('C1\n3.3uF')
pcb = C1.add(pcb,IC2.x,IC1.pad[8].y,z)

pcb = wire(pcb,w,
   C1.pad[1],
   IC2.pad[1])

pcb = wire(pcb,w,
   IC2.pad[3],
   C1.pad[2])

pcb = wire(pcb,w,
   C1.pad[1],
   point(C1.pad[1].x,IC1.pad[8].y+.1,z),
   IC1.pad[8])

J1 = MTA_serial('J1')
pcb = J1.add(pcb,IC2.x-.26,IC1.y-.01,z,angle=-90)
```

```
pcb = wire(pcb,w,
   IC2.pad[3],
   point(IC2.pad[3].x,IC1.pad[1].y,z),
   J1.pad[1])

pcb = wire(pcb,w,
   J1.pad[4],
   IC2.pad[2])

pcb = wire(pcb,w,
   J1.pad[3],
   point(J1.pad[2].x,IC1.pad[7].y+.15,z),
   IC1.pad[7])

J2 = MTA_ICP('J2')
pcb = J2.add(pcb,IC1.x+.28,IC1.y,z,angle=90)

pcb = wire(pcb,w,
   IC1.pad[6],
   point(IC1.pad[6].x,IC1.pad[6].y+.1,z),
   point(J2.pad[1].x-.03,J2.pad[1].y,z),
   J2.pad[1])

pcb = wire(pcb,w,
   J2.pad[2],
   IC1.pad[4])

pcb = wire(pcb,w,
   J2.pad[3],
   point(J2.x,J2.pad[5].y,z),
   IC1.pad[5])

pcb = wire(pcb,w,
   J2.pad[4],
   point(J2.pad[4].x,J2.pad[3].y-.05,z),
   point(J2.pad[3].x,IC1.pad[1].y-.12,z),
   IC1.pad[1])

pcb = wire(pcb,w,
   IC1.pad[7],
   point(IC1.pad[7].x,IC1.pad[7].y+.15,z),
   point(J2.pad[1].x+.02,J2.pad[1].y+.05,z),
   J2.pad[5])

T1 = phototransistor_1206('T1') # OP522
pcb = T1.add(pcb,R1.x+.09,R1.y,z,angle=-90)

pcb = wire(pcb,w,
   IC2.pad[3],
   T1.pad[2])

R2 = R_1206('R2\n49.9k')
pcb = R2.add(pcb,T1.x+.09,R1.y,z,angle=90)

pcb = wire(pcb,w,
   IC1.pad[8],
   R2.pad[1])

pcb = wire(pcb,w,
   T1.pad[1],
   R2.pad[2])

pcb = wire(pcb,w,
   R2.pad[2],
   IC1.pad[2])

H1 = hello(w,z)
pcb = H1.add(pcb,x0+.36,y0+.035,z)

#
# assign board and exterior to cad.function for milling
#
# use single-sided FR2
# use 1/64 end-mill
# set tool diameter = .0156
# set xy, z speed = 4
# set # contours = -1
#

cad.function = add(pcb.board,pcb.exterior)

#
# uncomment to export traces
#

#cad.function = pcb.board

#
# uncomment to export exterior
#

#cad.function = pcb.exterior

#
# uncomment to export solder mask
#

#cad.function = pcb.mask

#
```

```
# uncomment to mill out board
#
# use 1/32 end-mill
# set tool diameter = .0312
# set xy, z speed = .5
# set # contours = 1
#

#cad.function = pcb.interior
#z = -.065

#
# define limits and parameters
#

d = 0.05
cad.xmin = x0-d # min x to render
cad.xmax = x0+width+d # max x to render
cad.ymin = y0-d # min y to render
cad.ymax = y0+height+d # max y to render
cad.zmin = z
cad.zmax = .050
dpi = 200 # low resolution for previewing
#dpi = 500 # high resolution for machining
nxy = int(dpi*(cad.xmax-cad.xmin))
cad.nx = int(dpi*(cad.xmax-cad.xmin)) # x points to render
cad.ny = int(dpi*(cad.ymax-cad.ymin)) # y points to render
cad.nz = 1
cad.inches_per_unit = 1.0 # use inch units
cad.view('xy') # 2D view
```

```
#
# hello.temp.45.cad
#
# temperature sensor hello-world
#
# Neil Gershenfeld
# CBA MIT 10/25/07
#
# (c) Massachusetts Institute of Technology 2007
# Permission granted for experimental and personal use;
# license for commercial sale available from MIT.
#

#
# define shapes and transformation
#
# circle(x0, y0, r)
# cylinder(x0, y0, z0, z1, r)
# cone(x0, y0, z0, z1, r0)
# sphere(x0, y0, z0, r)
# torus(x0, y0, z0, r0, r1)
# rectangle(x0, x1, y0, y1)
# cube(x0, x1, y0, y1, z0, z1)
# triangle(x0, y0, x1, y1, x2, y2) (points in clockwise order)
# pyramid(x0, x1, y0, y1, z0, z1)
# function(Z_of_XY)
# functions(upper_Z_of_XY,lower_Z_of_XY)
# add(part1, part2)
# subtract(part1, part2)
# intersect(part1, part2)
# move(part,dx,dy)
# translate(part,dx,dy,dz)
# rotate(part, angle)
# rotate_x(part, angle)
# rotate_y(part, angle)
# rotate_z(part, angle)
# rotate_90(part)
# rotate_180(part)
# rotate_270(part)
# reflect_x(part)
# reflect_y(part)
# reflect_z(part)
# reflect_xy(part)
# reflect_xz(part)
# reflect_yz(part)
# scale_x(part, x0, sx)
# scale_y(part, y0, sy)
# scale_z(part, z0, sz)
# scale_xy(part, x0, y0, sxy)
# scale_xyz(part, x0, y0, z0, sxyz)
# coscale_x_y(part, x0, y0, y1, angle0, angle1, amplitude, offset)
# coscale_x_z(part, x0, z0, z1, angle0, angle1, amplitude, offset)
# coscale_xy_z(part, x0, y0, z0, z1, angle0, angle1, amplitude, offset)
# taper_x_y(part, x0, y0, y1, s0, s1)
# taper_x_z(part, x0, z0, z1, s0, s1)
# taper_xy_z(part, x0, y0, z0, z1, s0, s1)
# shear_x_y(part, y0, y1, dx0, dx1)
# shear_x_z(part, z0, z1, dx0, dx1)
# (more to come)

def circle(x0, y0, r):
   part = "(((X-x0)**2 + (Y-y0)**2) <= r**2)"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'r',str(r))
   return part

def cylinder(x0, y0, z0, z1, r):
   part = "(((X-x0)**2 + (Y-y0)**2 <= r**2) & (Z >= z0) & (Z <= z1))"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'r',str(r))
   return part

def cone(x0, y0, z0, z1, r0):
   part = cylinder(x0, y0, z0, z1, r0)
   part = taper_xy_z(part, x0, y0, z0, z1, 1.0, 0.0)
   return part

def sphere(x0, y0, z0, r):
   part = "(((X-x0)**2 + (Y-y0)**2 + (Z-z0)**2) <= r**2)"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'r',str(r))
   return part

def torus(x0, y0, z0, r0, r1):
   part = "(((r0 - sqrt((X-x0)**2 + (Y-y0)**2))**2 + (Z-z0)**2) <= r1**2)"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'r0',str(r0))
   part = replace(part,'r1',str(r1))
   return part

def rectangle(x0, x1, y0, y1):
```

```
    part = "((X >= x0) & (X <= x1) & (Y >= y0) & (Y <= y1))"
    part = replace(part,'x0',str(x0))
    part = replace(part,'x1',str(x1))
    part = replace(part,'y0',str(y0))
    part = replace(part,'y1',str(y1))
    return part

def cube(x0, x1, y0, y1, z0, z1):
    part = "((X >= x0) & (X <= x1) & (Y >= y0) & (Y <= y1) & (Z >= z0) & (Z <= z1))"
    part = replace(part,'x0',str(x0))
    part = replace(part,'x1',str(x1))
    part = replace(part,'y0',str(y0))
    part = replace(part,'y1',str(y1))
    part = replace(part,'z0',str(z0))
    part = replace(part,'z1',str(z1))
    return part

def triangle(x0, y0, x1, y1, x2, y2): # points in clockwise order
    part = "((((y1-y0)*(X-x0)-(x1-x0)*(Y-y0)) >= 0) & (((y2-y1)*(X-x1)-(x2-x1)*(Y-y1)) >= 0) & (((y0-y2)*(X-x2)-(x0-x2)*(Y-y2)) >= 0))"
    part = replace(part,'x0',str(x0))
    part = replace(part,'y0',str(y0))
    part = replace(part,'x1',str(x1))
    part = replace(part,'y1',str(y1))
    part = replace(part,'x2',str(x2))
    part = replace(part,'y2',str(y2))
    return part

def pyramid(x0, x1, y0, y1, z0, z1):
    part = cube(x0, x1, y0, y1, z0, z1)
    part = taper_xy_z(part, (x0+x1)/2., (y0+y1)/2., z0, z1, 1.0, 0.0)
    return part

def function(Z_of_XY):
    part = '(Z <= '+Z_of_XY+')'
    return part

def functions(upper_Z_of_XY,lower_Z_of_XY):
    part = '(Z <= '+upper_Z_of_XY+') & (Z >= '+lower_Z_of_XY+')'
    return part

def add(part1, part2):
    part = "part1 | part2"
    part = replace(part,'part1',part1)
    part = replace(part,'part2',part2)
    return part

def subtract(part1, part2):
    part = "(part1) & ~(part2)"
    part = replace(part,'part1',part1)
    part = replace(part,'part2',part2)
    return part

def intersect(part1, part2):
    part = "(part1) & (part2)"
    part = replace(part,'part1',part1)
    part = replace(part,'part2',part2)
    return part

def move(part,dx,dy):
    part = replace(part,'X','(X-'+str(dx)+')')
    part = replace(part,'Y','(Y-'+str(dy)+')')
    return part

def translate(part,dx,dy,dz):
    part = replace(part,'X','(X-'+str(dx)+')')
    part = replace(part,'Y','(Y-'+str(dy)+')')
    part = replace(part,'Z','(Z-'+str(dz)+')')
    return part

def rotate(part, angle):
    angle = angle*pi/180
    part = replace(part,'X','(cos(angle)*X+sin(angle)*y)')
    part = replace(part,'Y','(-sin(angle)*X+cos(angle)*y)')
    part = replace(part,'y','Y')
    part = replace(part,'angle',str(angle))
    return part

def rotate_x(part, angle):
    angle = angle*pi/180
    part = replace(part,'Y','(cos(angle)*Y+sin(angle)*z)')
    part = replace(part,'Z','(-sin(angle)*Y+cos(angle)*z)')
    part = replace(part,'z','Z')
    part = replace(part,'angle',str(angle))
    return part

def rotate_y(part, angle):
    angle = angle*pi/180
    part = replace(part,'X','(cos(angle)*X+sin(angle)*z)')
    part = replace(part,'Z','(-sin(angle)*X+cos(angle)*z)')
    part = replace(part,'z','Z')
    part = replace(part,'angle',str(angle))
    return part

def rotate_z(part, angle):
    angle = angle*pi/180
    part = replace(part,'X','(cos(angle)*X+sin(angle)*y)')
    part = replace(part,'Y','(-sin(angle)*X+cos(angle)*y)')
    part = replace(part,'y','Y')
    part = replace(part,'angle',str(angle))
    return part
```

```
def rotate_90(part):
   part = reflect_xy(part)
   part = reflect_y(part)
   return part

def rotate_180(part):
   part = reflect_xy(part)
   part = reflect_y(part)
   part = reflect_xy(part)
   part = reflect_y(part)
   return part

def rotate_270(part):
   part = reflect_xy(part)
   part = reflect_y(part)
   part = reflect_xy(part)
   part = reflect_y(part)
   part = reflect_xy(part)
   part = reflect_y(part)
   return part

def reflect_x(part):
   part = replace(part,'X','-X')
   return part

def reflect_y(part):
   part = replace(part,'Y','-Y')
   return part

def reflect_z(part):
   part = replace(part,'Z','-Z')
   return part

def reflect_xy(part):
   part = replace(part,'X','temp')
   part = replace(part,'Y','X')
   part = replace(part,'temp','Y')
   return part

def reflect_xz(part):
   part = replace(part,'X','temp')
   part = replace(part,'Z','X')
   part = replace(part,'temp','Z')
   return part

def reflect_yz(part):
   part = replace(part,'Y','temp')
   part = replace(part,'Z','Y')
   part = replace(part,'temp','Z')
   return part

def scale_x(part, x0, sx):
   part = replace(part,'X','(x0 + (X-x0)/sx)')
   part = replace(part,'x0',str(x0))
   part = replace(part,'sx',str(sx))
   return part

def scale_y(part, y0, sy):
   part = replace(part,'Y','(y0 + (Y-y0)/sy)')
   part = replace(part,'y0',str(y0))
   part = replace(part,'sy',str(sy))
   return part

def scale_z(part, z0, sz):
   part = replace(part,'Z','(z0 + (Z-z0)/sz)')
   part = replace(part,'z0',str(z0))
   part = replace(part,'sz',str(sz))
   return part

def scale_xy(part, x0, y0, sxy):
   part = replace(part,'X','(x0 + (X-x0)/sxy)')
   part = replace(part,'Y','(y0 + (Y-y0)/sxy)')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'sxy',str(sxy))
   return part

def scale_xyz(part, x0, y0, z0, sxyz):
   part = replace(part,'X','(x0 + (X-x0)/sxyz)')
   part = replace(part,'Y','(y0 + (Y-y0)/sxyz)')
   part = replace(part,'Z','(z0 + (Z-z0)/sxyz)')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'sxyz',str(sxyz))
   return part

def coscale_x_y(part, x0, y0, y1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
   part = replace(part,'X','(x0 + (X-x0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Y-y0)/(y1-y0))))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

def coscale_x_z(part, x0, z0, z1, angle0, angle1, amplitude, offset):
```

```
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
   part = replace(part,'X','(x0 + (X-x0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0))))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

def coscale_xy_z(part, x0, y0, z0, z1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
   part = replace(part,'X','(x0 + (X-x0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0))))')
   part = replace(part,'Y','(y0 + (Y-y0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0))))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

def taper_x_y(part, x0, y0, y1, s0, s1):
   part = replace(part,'X','(x0 + (X-x0)*(y1-y0)/(s1*(Y-y0) + s0*(y1-Y)))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'s0',str(s0))
   part = replace(part,'s1',str(s1))
   return part

def taper_x_z(part, x0, z0, z1, s0, s1):
   part = replace(part,'X','(x0 + (X-x0)*(z1-z0)/(s1*(Z-z0) + s0*(z1-Z)))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'s0',str(s0))
   part = replace(part,'s1',str(s1))
   return part

def taper_xy_z(part, x0, y0, z0, z1, s0, s1):
   part = replace(part,'X','(x0 + (X-x0)*(z1-z0)/(s1*(Z-z0) + s0*(z1-Z)))')
   part = replace(part,'Y','(y0 + (Y-y0)*(z1-z0)/(s1*(Z-z0) + s0*(z1-Z)))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'s0',str(s0))
   part = replace(part,'s1',str(s1))
   return part

def shear_x_y(part, y0, y1, dx0, dx1):
   part = replace(part,'X','(X - dx0 - (dx1-dx0)*(Y-y0)/(y1-y0))')
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'dx0',str(dx0))
   part = replace(part,'dx1',str(dx1))
   return part

def shear_x_z(part, z0, z1, dx0, dx1):
   part = replace(part,'X','(X - dx0 - (dx1-dx0)*(Z-z0)/(z1-z0))')
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'dx0',str(dx0))
   part = replace(part,'dx1',str(dx1))
   return part

def coshear_x_z(part, z0, z1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
   part = replace(part,'X','(X - offset - amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0)))')
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

#
# PCB classes and definitions
#

cad.labels = []

class PCB:
   def __init__(self,x0,y0,width,height,mask):
      self.board = "False"
      self.interior = rectangle(x0,x0+width,y0,y0+height)
      self.exterior = subtract("True",rectangle(x0,x0+width,y0,y0+height))
      self.mask = "False"
   def add(self,part):
      self.board = add(self.board,part)
      self.mask = add(self.mask,move(part,-mask,mask))
      self.mask = add(self.mask,move(part,-mask,-mask))
      self.mask = add(self.mask,move(part,mask,mask))
```

```
      self.mask = add(self.mask,move(part,mask,-mask))
      return self

class part:
   def add(self,pcb,x,y,z=0,angle=0):
      self.x = x
      self.y = y
      self.z = z
      self.angle = angle
      if (angle == 90):
         self.shape = rotate_90(self.shape)
      elif (angle == 180):
         self.shape = rotate_180(self.shape)
      elif ((angle == 270) | (angle == -90)):
         self.shape = rotate_270(self.shape)
      angle = pi*angle/180
      self.shape = translate(self.shape,x,y,z)
      for i in range(len(self.pad)):
         xnew = cos(angle)*self.pad[i].x + sin(angle)*self.pad[i].y
         ynew = -sin(angle)*self.pad[i].x + cos(angle)*self.pad[i].y
           self.pad[i].x = x + xnew
           self.pad[i].y = y + ynew
         self.pad[i].z += z
      cad.labels.append(cad_text(x,y,z,self.value,size=14))
      for i in range(len(self.labels)):
         xnew = cos(angle)*self.labels[i].x + sin(angle)*self.labels[i].y
         ynew = -sin(angle)*self.labels[i].x + cos(angle)*self.labels[i].y
           self.labels[i].x = x + xnew
           self.labels[i].y = y + ynew
         self.labels[i].z += z
           cad.labels.append(self.labels[i])
      pcb = pcb.add(self.shape)
      return pcb

def wire(pcb,width,*points):
   for i in range(1,len(points)):
      x0 = points[i-1].x
      y0 = points[i-1].y
      z0 = points[i-1].z
      x1 = points[i].x
      y1 = points[i].y
      z1 = points[i].z
      if (x0 < x1):
         pcb.board = add(pcb.board,cube(x0-width/2,x1+width/2,y0-width/2,y0+width/2,z0,z0))
      elif (x1 < x0):
         pcb.board = add(pcb.board,cube(x1-width/2,x0+width/2,y0-width/2,y0+width/2,z0,z0))
      if (y0 < y1):
         pcb.board = add(pcb.board,cube(x1-width/2,x1+width/2,y0-width/2,y1+width/2,z0,z0))
      elif (y1 < y0):
         pcb.board = add(pcb.board,cube(x1-width/2,x1+width/2,y1-width/2,y0+width/2,z0,z0))
   return pcb

#
# PCB library
#

#
# discretes
#

pad_0402 = cube(-.0175,.0175,-.014,.014,0,0)

class R_0402(part):
   #
   # 0402 resistor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_0402,-.0265,0,0)
      self.pad.append(point(-.0265,0,0))
      self.shape = add(self.shape,translate(pad_0402,.0265,0,0))
      self.pad.append(point(.0265,0,0))

pad_1206 = cube(-.032,.032,-.034,.034,0,0)

class R_1206(part):
   #
   # 1206 resistor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.06,0,0))
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.06,0,0))

class C_1206(part):
   #
   # 1206 capacitor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.06,0,0))
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
```

```
      self.pad.append(point(.06,0,0))

pad_1210 = cube(-.032,.032,-.048,.048,0,0)

class L_1210(part):
   #
   # 1210 inductor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_1210,-.06,0,0)
      self.pad.append(point(-.06,0,0))
      self.shape = add(self.shape,translate(pad_1210,.06,0,0))
      self.pad.append(point(.06,0,0))

pad_choke = cube(-.06,.06,-.06,.06,0,0)

class choke(part):
   #
   # Panasonic ELLCTV
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_choke,-.177,-.177,0)
      self.pad.append(point(-.177,-.177,0))
      self.shape = add(self.shape,translate(pad_choke,.177,.177,0))
      self.pad.append(point(.177,.177,0))

#
# connectors
#

pad_MTA = cube(-.021,.021,-.041,.041,0,0)
pad_MTA_solder = cube(-.071,.071,-.041,.041,0,0)

class MTA_2(part):
   #
   # AMP 1445121-2
   # MTA .050 SMT 2-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1
      #
      self.shape = translate(pad_MTA,-.025,-.1,0)
      self.pad.append(point(-.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,.1,0))
      self.pad.append(point(.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'2',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.187,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.187,0,0))

class MTA_power(part):
   #
   # AMP 1445121-2
   # MTA .050 SMT 2-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: Gnd
      #
      self.shape = translate(pad_MTA,-.025,-.1,0)
      self.pad.append(point(-.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: Vcc
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,.1,0))
      self.pad.append(point(.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Vcc',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.187,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.187,0,0))

class MTA_i0(part):
   #
   # AMP 1445121-3
   # MTA .050 SMT 3-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
```

```
      self.labels = []
      #
      # pin 1: GND
      #
      self.shape = translate(pad_MTA,.05,.1,0)
      self.pad.append(point(.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1\nGND',14))
      #
      # pin 2: power
      #
      self.shape = add(self.shape,translate(pad_MTA,-.05,.1,0))
      self.pad.append(point(-.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V',14))
      #
      # pin 3: data
      #
      self.shape = add(self.shape,translate(pad_MTA,0,-.1,0))
      self.pad.append(point(0,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'data',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.212,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.212,0,0))

class MTA_serial(part):
   #
   # AMP 1445121-4
   # MTA .050 SMT 4-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: Gnd
      #
      self.shape = translate(pad_MTA,-.075,-.1,0)
      self.pad.append(point(-.075,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: Tx
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,-.1,0))
      self.pad.append(point(.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Tx',14))
      #
      # pin 3: Rx
      #
      self.shape = add(self.shape,translate(pad_MTA,.075,.1,0))
      self.pad.append(point(.075,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Rx',14))
      #
      # pin 4: DTR
      #
      self.shape = add(self.shape,translate(pad_MTA,-.025,.1,0))
      self.pad.append(point(-.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DTR',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.237,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.237,0,0))

class MTA_ICP(part):
   #
   # AMP 1445121-5
   # MTA .050 SMT 4-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: MISO
      #
      self.shape = translate(pad_MTA,-.1,-.1,0)
      self.pad.append(point(-.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'MISO',14))
      #
      # pin 2: GND
      #
      self.shape = add(self.shape,translate(pad_MTA,0,-.1,0))
      self.pad.append(point(0,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))
      #
      # pin 3: MOSI
      #
      self.shape = add(self.shape,translate(pad_MTA,.1,-.1,0))
      self.pad.append(point(.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'MOSI',14))
      #
      # pin 4: -RESET
      #
      self.shape = add(self.shape,translate(pad_MTA,.05,.1,0))
      self.pad.append(point(.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'-RESET',14))
      #
      # pin 5: SCK
      #
      self.shape = add(self.shape,translate(pad_MTA,-.05,.1,0))
```

```
      self.pad.append(point(-.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'SCK',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.262,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.262,0,0))

class power_65mm(part):
   #
   # CUI PJ1-023-SMT
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: power
      #
      self.shape = cube(.433,.512,-.047,.047,0,0)
      self.pad.append(point(.467,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'P',14))
      #
      # pin 2: ground
      #
      self.shape = add(self.shape,cube(.285,.423,-.189,-.098,0,0))
      self.pad.append(point(.354,-.144,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 3: contact
      #
      self.shape = add(self.shape,cube(.325,.463,.098,.189,0,0))
      self.pad.append(point(.394,.144,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,cube(.108,.246,-.169,-.110,0,0))
      self.shape = add(self.shape,cube(.069,.207,.110,.169,0,0))

pad_stereo_2_5mm = cube(-.03,.03,-.05,.05,0,0)

class stereo_2_5mm(part):
   #
   # CUI SJ1-2533-SMT
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: base
      #
      self.shape = translate(pad_stereo_2_5mm,-.130,-.16,0)
      self.pad.append(point(-.130,-.149,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'base',14))
      #
      # pin 2: tip
      #
      self.shape = add(self.shape,translate(pad_stereo_2_5mm,.197,.15,0))
      self.pad.append(point(.197,.141,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'tip',14))
      #
      # pin 3: middle
      #
      self.shape = add(self.shape,translate(pad_stereo_2_5mm,-.012,-.16,0))
      self.pad.append(point(-.012,-.149,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'middle',14))

pad_Molex = cube(-.0155,.0155,-.0265,.0265,0,0)
pad_Molex_solder = cube(-.055,.055,-.065,.065,0,0)

class Molex_serial(part):
   #
   # Molex 53261-0471
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: Rx
      #
      self.shape = translate(pad_Molex,-.075,.064,0)
      self.pad.append(point(-.075,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Rx',14))
      #
      # pin 2: Tx
      #
      self.shape = add(self.shape,translate(pad_Molex,-.025,.064,0))
      self.pad.append(point(-.025,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Tx',14))
      #
      # pin 3: DTR
      #
      self.shape = add(self.shape,translate(pad_Molex,.025,.064,0))
      self.pad.append(point(.025,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DTR',14))
      #
      # pin 4: GND
      #
```

```
      self.shape = add(self.shape,translate(pad_Molex,.075,.064,0))
      self.pad.append(point(.075,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_Molex_solder,-.16,-.065,0))
      self.shape = add(self.shape,translate(pad_Molex_solder,.16,-.065,0))

#
# switches
#

pad_button_6mm = cube(-.04,.04,-.03,.03,0,0)

class button_6mm(part):
   #
   # Omron 6mm pushbutton
   # B3SN-3112P
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # left 1
      #
      self.shape = translate(pad_button_6mm,-.125,.08,0)
      self.pad.append(point(-.125,.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'L1',14))
      #
      # right 1
      #
      self.shape = add(self.shape,translate(pad_button_6mm,-.125,-.08,0))
      self.pad.append(point(-.125,-.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'R1',14))
      #
      # right 2
      #
      self.shape = add(self.shape,translate(pad_button_6mm,.125,-.08,0))
      self.pad.append(point(.125,-.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'R2',14))
      #
      # left 2
      #
      self.shape = add(self.shape,translate(pad_button_6mm,.125,.08,0))
      self.pad.append(point(.125,.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'L2',14))

#
# crystals and resonators
#

pad_XTAL = cube(-.016,.016,-.077,.077,0,0)

class XTAL(part):
   #
   # Panasonic EFOBM series
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # left
      #
      self.shape = translate(pad_XTAL,-.053,0,0)
      self.pad.append(point(-.053,0,0))
      #
      # ground
      #
      self.shape = add(self.shape,translate(pad_XTAL,0,0,0))
      self.pad.append(point(0,0,0))
      #
      # right
      #
      self.shape = add(self.shape,translate(pad_XTAL,.053,0,0))
      self.pad.append(point(.053,0,0))

#
# diodes, transistors, regulators
#

class D_1206(part):
   #
   # 1206 diode
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # anode
      #
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A',14))
      #
      # cathode
      #
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
```

```
      self.pad.append(point(.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))

class LED_1206(part):
   #
   # 1206 LED
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # anode
      #
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A',14))
      #
      # cathode
      #
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))

class phototransistor_1206(part):
   #
   # 1206 phototransistor
   # OPTEK 520,521,522
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # collector
      #
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))
      #
      # emitter
      #
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'E',14))

pad_SOD_123 = cube(-.02,.02,-.024,.024,0,0)

class D_SOD_123(part):
   #
   # SOD-123 diode
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # anode
      #
      self.shape = translate(pad_SOD_123,-.07,0,0)
      self.pad.append(point(-.07,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A',14))
      #
      # cathode
      #
      self.shape = add(self.shape,translate(pad_SOD_123,.07,0,0))
      self.pad.append(point(.07,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))

pad_SOT23 = cube(-.015,.015,-.02,.02,0,0)

class NMOSFET_SOT23(part):
   #
   # Fairchild NDS355AN
   #
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: gate
      #
      self.shape = translate(pad_SOT23,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 2: source
      #
      self.shape = add(self.shape,translate(pad_SOT23,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'S',14))
      #
      # pin 3: drain
      #
      self.shape = add(self.shape,translate(pad_SOT23,0,.045,0))
      self.pad.append(point(0,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'D',14))
```

```
class PMOSFET_SOT23(part):
   #
   # Fairchild NDS356AP
   #
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: gate
      #
      self.shape = translate(pad_SOT23,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 2: source
      #
      self.shape = add(self.shape,translate(pad_SOT23,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'S',14))
      #
      # pin 3: drain
      #
      self.shape = add(self.shape,translate(pad_SOT23,0,.045,0))
      self.pad.append(point(0,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'D',14))

class regulator_SOT23(part):
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: output
      #
      self.shape = translate(pad_SOT23,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: input
      #
      self.shape = add(self.shape,translate(pad_SOT23,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'in',14))
      #
      # pin 3: ground
      #
      self.shape = add(self.shape,translate(pad_SOT23,0,.045,0))
      self.pad.append(point(0,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'gnd',14))

pad_SOT223 = cube(-.02,.02,-.03,.03,0,0)
pad_SOT223_ground = cube(-.065,.065,-.03,.03,0,0)

class regulator_SOT223(part):
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: input
      #
      self.shape = translate(pad_SOT223,-.09,-.12,0)
      self.pad.append(point(-.09,-.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1I',14))
      #
      # pin 2: ground
      #
      self.shape = add(self.shape,translate(pad_SOT223,0,-.12,0))
      self.pad.append(point(0,-.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 3: output
      #
      self.shape = add(self.shape,translate(pad_SOT223,.09,-.12,0))
      self.pad.append(point(.09,-.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'O',14))
      #
      # pin 4: ground
      #
      self.shape = add(self.shape,translate(pad_SOT223_ground,0,.12,0))
      self.pad.append(point(0,.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))

pad_SM8 = cube(-.035,.035,-.016,.016,0,0)

class H_bridge_SM8(part):
   #
   # Zetex ZXMHC3A01T8
   #
   def __init__(self,value=''):
      self.value = value
```

```
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      d = .13
      #
      # pin 1: G3 (right N gate)
      #
      self.shape = translate(pad_SM8,-d,.09,0)
      self.pad.append(point(-d,.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GRN',14))
      #
      # pin 2: S2 S3 (N source)
      #
      self.shape = add(self.shape,translate(pad_SM8,-d,.03,0))
      self.pad.append(point(-d,.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'SN',14))
      #
      # pin 3: G2 (left N gate)
      #
      self.shape = add(self.shape,translate(pad_SM8,-d,-.03,0))
      self.pad.append(point(-d,-.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GLN',14))
      #
      # pin 4: G1 (left P gate)
      #
      self.shape = add(self.shape,translate(pad_SM8,-d,-.09,0))
      self.pad.append(point(-d,-.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GLP',14))
      #
      # pin 5: D1 D2 (left drain)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,-.09,0))
      self.pad.append(point(d,-.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DL',14))
      #
      # pin 6: S1 S4 (P source)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,-.03,0))
      self.pad.append(point(d,-.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'SP',14))
      #
      # pin 7: D3 D4 (right drain)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,.03,0))
      self.pad.append(point(d,.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DR',14))
      #
      # pin 8: G4 (right N gate)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,.09,0))
      self.pad.append(point(d,.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GRP',14))

#
# ICs
#

pad_SOT23_5 = cube(-.01,.01,-.02,.02,0,0)

class op_amp_SOT23_5(part):
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: output
      #
      self.shape = translate(pad_SOT23_5,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'O',14))
      #
      # pin 2: V-
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,0,-.045,0))
      self.pad.append(point(0,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V-',14))
      #
      # pin 3: I+
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'I+',14))
      #
      # pin 4: I-
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,.0375,.045,0))
      self.pad.append(point(.0375,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'I-',14))
      #
      # pin 5: V+
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,-.0375,.045,0))
      self.pad.append(point(-.0375,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V+',14))

pad_SOICN = cube(-.035,.035,-.015,.015,0,0)
```

```
class op_amp_SOICN(part):
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: A out
      #
      self.shape = translate(pad_SOICN,-.12,.075,0)
      self.pad.append(point(-.12,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1 Ao',14))
      #
      # pin 2: A-
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.025,0))
      self.pad.append(point(-.12,.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A-',14))
      #
      # pin 3: A+
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.025,0))
      self.pad.append(point(-.12,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A+',14))
      #
      # pin 4: V-
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.075,0))
      self.pad.append(point(-.12,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V-',14))
      #
      # pin 5: B+
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.075,0))
      self.pad.append(point(.12,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'B+',14))
      #
      # pin 6: B-
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.025,0))
      self.pad.append(point(.12,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'B-',14))
      #
      # pin 7: B out
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.025,0))
      self.pad.append(point(.12,.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Bo',14))
      #
      # pin 8: V+
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.075,0))
      self.pad.append(point(.12,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V+',14))

pad_SOIC = cube(-.043,.043,-.015,.015,0,0)

class ATtiny45_SOIC(part):
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: PB5/dW/ADC0/-RESET/PCINT5
      #
      self.shape = translate(pad_SOIC,-.14,.075,0)
      self.pad.append(point(-.14,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: PB3/ADC3/-OC1B/CLKI/XTAL1/PCINT3
      #
      self.shape = add(self.shape,translate(pad_SOIC,-.14,.025,0))
      self.pad.append(point(-.14,.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB3',14))
      #
      # pin 3: PB4/ADC2/OC1B/CLKO/XTAL2/PCINT4
      #
      self.shape = add(self.shape,translate(pad_SOIC,-.14,-.025,0))
      self.pad.append(point(-.14,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB4',14))
      #
      # pin 4: GND
      #
      self.shape = add(self.shape,translate(pad_SOIC,-.14,-.075,0))
      self.pad.append(point(-.14,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))
      #
      # pin 5: PB0/MOSI/DI/SDA/AIN0/OC0A/-OC1A/AREF/PCINT0
      #
      self.shape = add(self.shape,translate(pad_SOIC,.14,-.075,0))
      self.pad.append(point(.14,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 6: PB1/MISO/DO/AIN1/OC0B/OC1A/PCINT1
      #
      self.shape = add(self.shape,translate(pad_SOIC,.14,-.025,0))
      self.pad.append(point(.14,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB1',14))
      #
      # pin 7: PB2/SCK/USCK/SCL/ADC1/T0/INT0/PCINT2
      #
```

```
      self.shape = add(self.shape,translate(pad_SOIC,.14,.025,0))
      self.pad.append(point(.14,.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB2',14))
      #
      # pin 8: VCC
      #
      self.shape = add(self.shape,translate(pad_SOIC,.14,.075,0))
      self.pad.append(point(.14,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'VCC',14))

class ATtiny44_SOICN(part):
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: VCC
      #
      self.shape = translate(pad_SOICN,-.12,.15,0)
      self.pad.append(point(-.12,.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: PB0/XTAL1/PCINT8
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 3: PB1/XTAL2/PCINT9
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.050,0))
      self.pad.append(point(-.12,.05,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB1',14))
      #
      # pin 4: PB3/dW/-RESET/PCINT11
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,0,0))
      self.pad.append(point(-.12,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB3',14))
      #
      # pin 5: PB2/CKOUT/OC0A/INT0/PCINT10
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.05,0))
      self.pad.append(point(-.12,-.05,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB2',14))
      #
      # pin 6: PA7/ADC7/OC0B/ICP/PCINT7
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.1,0))
      self.pad.append(point(-.12,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA7',14))
      #
      # pin 7: PA6/ADC6/MOSI/SDA/OC1A/PCINT6
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.15,0))
      self.pad.append(point(-.12,-.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA6',14))
      #
      # pin 8: PA5/ADC5/DO/MISO/OC1B/PCINT5
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.15,0))
      self.pad.append(point(.12,-.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA5',14))
      #
      # pin 9: PA4/ADC4/USCK/SCL/T1/PCINT4
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.1,0))
      self.pad.append(point(.12,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA4',14))
      #
      # pin 10: PA3/ADC3/T0/PCINT3
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.05,0))
      self.pad.append(point(.12,-.05,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA3',14))
      #
      # pin 11: PA2/ADC2/AIN1/PCINT2
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,0,0))
      self.pad.append(point(.12,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA2',14))
      #
      # pin 12: PA1/ADC1/AIN0/PCINT1
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.050,0))
      self.pad.append(point(.12,.05,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA1',14))
      #
      # pin 13: PA0/ADC0/AREF/PCINT0
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.1,0))
      self.pad.append(point(.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA0',14))
      #
      # pin 14: GND
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.15,0))
      self.pad.append(point(.12,.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))

pad_TQFP = cube(-.043,.043,-.015,.015,0,0)
```

```
class ATmega88_TQFP(part):
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []

      #
      # pin 1: PD3/PCINT19/OC2B/INT1
      #
      self.shape = translate(pad_SOICN,-.12,.15,0)
      self.pad.append(point(-.12,.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: PD4/PCINT20/XCK/T0
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 3: GND
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 4: VCC
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 5: GND
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 6: VCC
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 7: PB6/PCINT6/XTAL1/TOSC1
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 8: PB7/PCINT7/XTAL2/TOSC2
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

      #
      # pin 9: PD5/PCINT21/OC0B/T1
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 10: PD6/PCINT22/OC0A/AIN0
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 11: PD7/PCINT23/AIN1
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 12: PB0/PCINT0/CLKO/ICP1
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 13: PB1/PCINT1/OC1A
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 14: PB2/PCINT2/-SS/OC1B
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 15: PB3/PCINT3/OC2A/MOSI
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 16: PB4/PCINT4/MISO
      #
```

```
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

#
# pin 17: PB5/SCK/PCINT5
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 18: AVCC
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 19: ADC6
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 20: AREF
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 21: GND
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 22: ADC7
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 23: PC0/ADC0/PCINT8
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 24: PC1/ADC1/PCINT9
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

#
# pin 25: PC2/ADC2/PCINT10
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 26: PC2/ADC3/PCINT11
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 27: PC4/ADC4/SDA/PCINT12
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 28: PC5/ADC5/SCL/PCINT13
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 29: PC6/-RESET/PCINT14
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 30: PD0/RXD/PCINT16
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 31: PD1/TXD/PCINT17
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 32: PD2/INT0/PCINT18
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
```

```
#
# graphics
#

class CBA(part):
   def __init__(self,r=.02):
      self.value = ''
      self.pad = [point(0,0,0)]
      self.labels = []
      d = 3*r
      self.shape = circle(0,0,r)
      self.shape = add(self.shape,translate(circle(0,0,r),-d,d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),-d,0,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),-d,-d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),0,-d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),d,-d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),d,0,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),d,d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),0,d,0))

class hello(part):
   #
   # HELLO
   #
   def __init__(self,w,z):
      self.value = ''
      self.pad = [point(0,0,0)]
      self.labels = []
      # H
      self.shape = rectangle(-.01,.01,0,.1)
      self.shape = add(self.shape,rectangle(0,.05,.04,.06))
      self.shape = add(self.shape,rectangle(.04,.06,0,.1))
      # E
      self.shape = add(self.shape,rectangle(.08,.1,0,.1))
      self.shape = add(self.shape,rectangle(.1,.14,0,.02))
      self.shape = add(self.shape,rectangle(.1,.13,.04,.06))
      self.shape = add(self.shape,rectangle(.1,.14,.08,.1))
      # L
      self.shape = add(self.shape,rectangle(.16,.18,0,.1))
      self.shape = add(self.shape,rectangle(.18,.21,0,.02))
      # L
      self.shape = add(self.shape,rectangle(.23,.25,0,.1))
      self.shape = add(self.shape,rectangle(.25,.28,0,.02))
      # 0
      self.shape = add(self.shape,rectangle(.3,.32,0,.1))
      self.shape = add(self.shape,rectangle(.32,.345,0,.02))
      self.shape = add(self.shape,rectangle(.32,.345,.08,.1))
      self.shape = add(self.shape,rectangle(.345,.365,0,.1))

#
# define board
#

x0 = 1
y0 = 1
width = 1.36
height = .7
z = -.005
w = .018
mask = .004

pcb = PCB(x0,y0,width,height,mask)

IC1 = ATtiny45_SOIC('IC1\nt45')
pcb = IC1.add(pcb,x0+.92,y0+.35,z,angle=-90)

R1 = R_1206('R1\n10k')
pcb = R1.add(pcb,IC1.pad[1].x-.27,IC1.y,z,angle=90)

pcb = wire(pcb,w,
   IC1.pad[1],
   R1.pad[2])

pcb = wire(pcb,w,
   IC1.pad[8],
   R1.pad[1])

IC2 = regulator_SOT23('IC2\n5V')
pcb = IC2.add(pcb,R1.x-.155,R1.y,z,angle=90)

pcb = wire(pcb,w,
   IC2.pad[3],
   IC1.pad[4])

C1 = C_1206('C1\n3.3uF')
pcb = C1.add(pcb,IC2.x,IC1.pad[8].y,z)

pcb = wire(pcb,w,
   C1.pad[1],
   IC2.pad[1])

pcb = wire(pcb,w,
   IC2.pad[3],
   C1.pad[2])

pcb = wire(pcb,w,
   C1.pad[1],
   point(C1.pad[1].x,IC1.pad[8].y+.1,z),
   IC1.pad[8])
```

```
J1 = MTA_serial('J1')
pcb = J1.add(pcb,IC2.x-.26,IC1.y-.01,z,angle=-90)

pcb = wire(pcb,w,
   IC2.pad[3],
   point(IC2.pad[3].x,IC1.pad[1].y,z),
   J1.pad[1])

pcb = wire(pcb,w,
   J1.pad[4],
   IC2.pad[2])

pcb = wire(pcb,w,
   J1.pad[3],
   point(J1.pad[2].x,IC1.pad[7].y+.15,z),
   IC1.pad[7])

J2 = MTA_ICP('J2')
pcb = J2.add(pcb,IC1.x+.28,IC1.y,z,angle=90)

pcb = wire(pcb,w,
   IC1.pad[6],
   point(IC1.pad[6].x,IC1.pad[6].y+.1,z),
   point(J2.pad[1].x-.03,J2.pad[1].y,z),
   J2.pad[1])

pcb = wire(pcb,w,
   J2.pad[2],
   IC1.pad[4])

pcb = wire(pcb,w,
   J2.pad[3],
   point(J2.x,J2.pad[5].y,z),
   IC1.pad[5])

pcb = wire(pcb,w,
   J2.pad[4],
   point(J2.pad[4].x,J2.pad[3].y-.05,z),
   point(J2.pad[3].x,IC1.pad[1].y-.12,z),
   IC1.pad[1])

pcb = wire(pcb,w,
   IC1.pad[7],
   point(IC1.pad[7].x,IC1.pad[7].y+.15,z),
   point(J2.pad[1].x+.02,J2.pad[1].y+.05,z),
   J2.pad[5])

R2 = R_1206('R2\n10k\nNTC') #
pcb = R2.add(pcb,R1.x+.09,R1.y,z,angle=-90)

pcb = wire(pcb,w,
   IC2.pad[3],
   R2.pad[2])

R3 = R_1206('R3\n10k')
pcb = R3.add(pcb,R2.x+.09,R2.y,z,angle=90)

pcb = wire(pcb,w,
   IC1.pad[8],
   R3.pad[1])

pcb = wire(pcb,w,
   R2.pad[1],
   R3.pad[2])

pcb = wire(pcb,w,
   R3.pad[2],
   IC1.pad[2])

H1 = hello(w,z)
pcb = H1.add(pcb,x0+.36,y0+.035,z)

#
# assign board and exterior to cad.function for milling
#
# use single-sided FR2
# use 1/64 end-mill
# set tool diameter = .0156
# set xy, z speed = 4
# set # contours = -1
#

cad.function = add(pcb.board,pcb.exterior)

#
# uncomment to export traces
#

#cad.function = pcb.board

#
# uncomment to export exterior
#

#cad.function = pcb.exterior

#
# uncomment to export solder mask
#

#cad.function = pcb.mask
```

```
#
# uncomment to mill out board
#
# use 1/32 end-mill
# set tool diameter = .0312
# set xy, z speed = .5
# set # contours = 1
#

#cad.function = pcb.interior
#z = -.065

#
# define limits and parameters
#

d = 0.05
cad.xmin = x0-d # min x to render
cad.xmax = x0+width+d # max x to render
cad.ymin = y0-d # min y to render
cad.ymax = y0+height+d # max y to render
cad.zmin = z
cad.zmax = .050
dpi = 200 # low resolution for previewing
#dpi = 500 # high resolution for machining
nxy = int(dpi*(cad.xmax-cad.xmin))
cad.nx = int(dpi*(cad.xmax-cad.xmin)) # x points to render
cad.ny = int(dpi*(cad.ymax-cad.ymin)) # y points to render
cad.nz = 1
cad.inches_per_unit = 1.0 # use inch units
cad.view('xy') # 2D view
```

```
#
# hello.step.45.cad
#
# step-response hello-world
#
# Neil Gershenfeld
# CBA MIT 10/27/07
#
# (c) Massachusetts Institute of Technology 2007
# Permission granted for experimental and personal use;
# license for commercial sale available from MIT.
#

#
# define shapes and transformation
#
# circle(x0, y0, r)
# cylinder(x0, y0, z0, z1, r)
# cone(x0, y0, z0, z1, r0)
# sphere(x0, y0, z0, r)
# torus(x0, y0, z0, r0, r1)
# rectangle(x0, x1, y0, y1)
# cube(x0, x1, y0, y1, z0, z1)
# triangle(x0, y0, x1, y1, x2, y2) (points in clockwise order)
# pyramid(x0, x1, y0, y1, z0, z1)
# function(Z_of_XY)
# functions(upper_Z_of_XY,lower_Z_of_XY)
# add(part1, part2)
# subtract(part1, part2)
# intersect(part1, part2)
# move(part,dx,dy)
# translate(part,dx,dy,dz)
# rotate(part, angle)
# rotate_x(part, angle)
# rotate_y(part, angle)
# rotate_z(part, angle)
# rotate_90(part)
# rotate_180(part)
# rotate_270(part)
# reflect_x(part)
# reflect_y(part)
# reflect_z(part)
# reflect_xy(part)
# reflect_xz(part)
# reflect_yz(part)
# scale_x(part, x0, sx)
# scale_y(part, y0, sy)
# scale_z(part, z0, sz)
# scale_xy(part, x0, y0, sxy)
# scale_xyz(part, x0, y0, z0, sxyz)
# coscale_x_y(part, x0, y0, y1, angle0, angle1, amplitude, offset)
# coscale_x_z(part, x0, z0, z1, angle0, angle1, amplitude, offset)
# coscale_xy_z(part, x0, y0, z0, z1, angle0, angle1, amplitude, offset)
# taper_x_y(part, x0, y0, y1, s0, s1)
# taper_x_z(part, x0, z0, z1, s0, s1)
# taper_xy_z(part, x0, y0, z0, z1, s0, s1)
# shear_x_y(part, y0, y1, dx0, dx1)
# shear_x_z(part, z0, z1, dx0, dx1)
# (more to come)

def circle(x0, y0, r):
   part = "(((X-x0)**2 + (Y-y0)**2) <= r**2)"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'r',str(r))
   return part

def cylinder(x0, y0, z0, z1, r):
   part = "(((X-x0)**2 + (Y-y0)**2 <= r**2) & (Z >= z0) & (Z <= z1))"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'r',str(r))
   return part

def cone(x0, y0, z0, z1, r0):
   part = cylinder(x0, y0, z0, z1, r0)
   part = taper_xy_z(part, x0, y0, z0, z1, 1.0, 0.0)
   return part

def sphere(x0, y0, z0, r):
   part = "(((X-x0)**2 + (Y-y0)**2 + (Z-z0)**2) <= r**2)"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'r',str(r))
   return part

def torus(x0, y0, z0, r0, r1):
   part = "(((r0 - sqrt((X-x0)**2 + (Y-y0)**2))**2 + (Z-z0)**2) <= r1**2)"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'r0',str(r0))
   part = replace(part,'r1',str(r1))
   return part

def rectangle(x0, x1, y0, y1):
   part = "((X >= x0) & (X <= x1) & (Y >= y0) & (Y <= y1))"
   part = replace(part,'x0',str(x0))
```

```
   part = replace(part,'x1',str(x1))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   return part

def cube(x0, x1, y0, y1, z0, z1):
   part = "((X >= x0) & (X <= x1) & (Y >= y0) & (Y <= y1) & (Z >= z0) & (Z <= z1))"
   part = replace(part,'x0',str(x0))
   part = replace(part,'x1',str(x1))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   return part

def triangle(x0, y0, x1, y1, x2, y2): # points in clockwise order
   part = "((((y1-y0)*(X-x0)-(x1-x0)*(Y-y0)) >= 0) & (((y2-y1)*(X-x1)-(x2-x1)*(Y-y1)) >= 0) & (((y0-y2)*(X-x2)-(x0-x2)*(Y-y2)) >= 0))"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'x1',str(x1))
   part = replace(part,'y1',str(y1))
   part = replace(part,'x2',str(x2))
   part = replace(part,'y2',str(y2))
   return part

def pyramid(x0, x1, y0, y1, z0, z1):
   part = cube(x0, x1, y0, y1, z0, z1)
   part = taper_xy_z(part, (x0+x1)/2., (y0+y1)/2., z0, z1, 1.0, 0.0)
   return part

def function(Z_of_XY):
   part = '(Z <= '+Z_of_XY+')'
   return part

def functions(upper_Z_of_XY,lower_Z_of_XY):
   part = '(Z <= '+upper_Z_of_XY+') & (Z >= '+lower_Z_of_XY+')'
   return part

def add(part1, part2):
   part = "part1 | part2"
   part = replace(part,'part1',part1)
   part = replace(part,'part2',part2)
   return part

def subtract(part1, part2):
   part = "(part1) & ~(part2)"
   part = replace(part,'part1',part1)
   part = replace(part,'part2',part2)
   return part

def intersect(part1, part2):
   part = "(part1) & (part2)"
   part = replace(part,'part1',part1)
   part = replace(part,'part2',part2)
   return part

def move(part,dx,dy):
   part = replace(part,'X','(X-'+str(dx)+')')
   part = replace(part,'Y','(Y-'+str(dy)+')')
   return part

def translate(part,dx,dy,dz):
   part = replace(part,'X','(X-'+str(dx)+')')
   part = replace(part,'Y','(Y-'+str(dy)+')')
   part = replace(part,'Z','(Z-'+str(dz)+')')
   return part

def rotate(part, angle):
   angle = angle*pi/180
   part = replace(part,'X','(cos(angle)*X+sin(angle)*y)')
   part = replace(part,'Y','(-sin(angle)*X+cos(angle)*y)')
   part = replace(part,'y','Y')
   part = replace(part,'angle',str(angle))
   return part

def rotate_x(part, angle):
   angle = angle*pi/180
   part = replace(part,'Y','(cos(angle)*Y+sin(angle)*z)')
   part = replace(part,'Z','(-sin(angle)*Y+cos(angle)*z)')
   part = replace(part,'z','Z')
   part = replace(part,'angle',str(angle))
   return part

def rotate_y(part, angle):
   angle = angle*pi/180
   part = replace(part,'X','(cos(angle)*X+sin(angle)*z)')
   part = replace(part,'Z','(-sin(angle)*X+cos(angle)*z)')
   part = replace(part,'z','Z')
   part = replace(part,'angle',str(angle))
   return part

def rotate_z(part, angle):
   angle = angle*pi/180
   part = replace(part,'X','(cos(angle)*X+sin(angle)*y)')
   part = replace(part,'Y','(-sin(angle)*X+cos(angle)*y)')
   part = replace(part,'y','Y')
   part = replace(part,'angle',str(angle))
   return part

def rotate_90(part):
   part = reflect_xy(part)
```

```
   part = reflect_y(part)
   return part

def rotate_180(part):
   part = reflect_xy(part)
   part = reflect_y(part)
   part = reflect_xy(part)
   part = reflect_y(part)
   return part

def rotate_270(part):
   part = reflect_xy(part)
   part = reflect_y(part)
   part = reflect_xy(part)
   part = reflect_y(part)
   part = reflect_xy(part)
   part = reflect_y(part)
   return part

def reflect_x(part):
   part = replace(part,'X','-X')
   return part

def reflect_y(part):
   part = replace(part,'Y','-Y')
   return part

def reflect_z(part):
   part = replace(part,'Z','-Z')
   return part

def reflect_xy(part):
   part = replace(part,'X','temp')
   part = replace(part,'Y','X')
   part = replace(part,'temp','Y')
   return part

def reflect_xz(part):
   part = replace(part,'X','temp')
   part = replace(part,'Z','X')
   part = replace(part,'temp','Z')
   return part

def reflect_yz(part):
   part = replace(part,'Y','temp')
   part = replace(part,'Z','Y')
   part = replace(part,'temp','Z')
   return part

def scale_x(part, x0, sx):
   part = replace(part,'X','(x0 + (X-x0)/sx)')
   part = replace(part,'x0',str(x0))
   part = replace(part,'sx',str(sx))
   return part

def scale_y(part, y0, sy):
   part = replace(part,'Y','(y0 + (Y-y0)/sy)')
   part = replace(part,'y0',str(y0))
   part = replace(part,'sy',str(sy))
   return part

def scale_z(part, z0, sz):
   part = replace(part,'Z','(z0 + (Z-z0)/sz)')
   part = replace(part,'z0',str(z0))
   part = replace(part,'sz',str(sz))
   return part

def scale_xy(part, x0, y0, sxy):
   part = replace(part,'X','(x0 + (X-x0)/sxy)')
   part = replace(part,'Y','(y0 + (Y-y0)/sxy)')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'sxy',str(sxy))
   return part

def scale_xyz(part, x0, y0, z0, sxyz):
   part = replace(part,'X','(x0 + (X-x0)/sxyz)')
   part = replace(part,'Y','(y0 + (Y-y0)/sxyz)')
   part = replace(part,'Z','(z0 + (Z-z0)/sxyz)')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'sxyz',str(sxyz))
   return part

def coscale_x_y(part, x0, y0, y1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
   part = replace(part,'X','(x0 + (X-x0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Y-y0)/(y1-y0))))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

def coscale_x_z(part, x0, z0, z1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
```

```
   part = replace(part,'X','(x0 + (X-x0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0))))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

def coscale_xy_z(part, x0, y0, z0, z1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
   part = replace(part,'X','(x0 + (X-x0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0))))')
   part = replace(part,'Y','(y0 + (Y-y0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0))))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

def taper_x_y(part, x0, y0, y1, s0, s1):
   part = replace(part,'X','(x0 + (X-x0)*(y1-y0)/(s1*(Y-y0) + s0*(y1-Y)))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'s0',str(s0))
   part = replace(part,'s1',str(s1))
   return part

def taper_x_z(part, x0, z0, z1, s0, s1):
   part = replace(part,'X','(x0 + (X-x0)*(z1-z0)/(s1*(Z-z0) + s0*(z1-Z)))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'s0',str(s0))
   part = replace(part,'s1',str(s1))
   return part

def taper_xy_z(part, x0, y0, z0, z1, s0, s1):
   part = replace(part,'X','(x0 + (X-x0)*(z1-z0)/(s1*(Z-z0) + s0*(z1-Z)))')
   part = replace(part,'Y','(y0 + (Y-y0)*(z1-z0)/(s1*(Z-z0) + s0*(z1-Z)))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'s0',str(s0))
   part = replace(part,'s1',str(s1))
   return part

def shear_x_y(part, y0, y1, dx0, dx1):
   part = replace(part,'X','(X - dx0 - (dx1-dx0)*(Y-y0)/(y1-y0))')
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'dx0',str(dx0))
   part = replace(part,'dx1',str(dx1))
   return part

def shear_x_z(part, z0, z1, dx0, dx1):
   part = replace(part,'X','(X - dx0 - (dx1-dx0)*(Z-z0)/(z1-z0))')
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'dx0',str(dx0))
   part = replace(part,'dx1',str(dx1))
   return part

def coshear_x_z(part, z0, z1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
   part = replace(part,'X','(X - offset - amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0)))')
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

#
# PCB classes and definitions
#

cad.labels = []

class PCB:
   def __init__(self,x0,y0,width,height,mask):
      self.board = "False"
      self.interior = rectangle(x0,x0+width,y0,y0+height)
      self.exterior = subtract("True",rectangle(x0,x0+width,y0,y0+height))
      self.mask = "False"
   def add(self,part):
      self.board = add(self.board,part)
      self.mask = add(self.mask,move(part,-mask,mask))
      self.mask = add(self.mask,move(part,-mask,-mask))
      self.mask = add(self.mask,move(part,mask,mask))
      self.mask = add(self.mask,move(part,mask,-mask))
      return self
```

```
class part:
   def add(self,pcb,x,y,z=0,angle=0):
      self.x = x
      self.y = y
      self.z = z
      self.angle = angle
      if (angle == 90):
         self.shape = rotate_90(self.shape)
      elif (angle == 180):
         self.shape = rotate_180(self.shape)
      elif ((angle == 270) | (angle == -90)):
         self.shape = rotate_270(self.shape)
      angle = pi*angle/180
      self.shape = translate(self.shape,x,y,z)
      for i in range(len(self.pad)):
         xnew = cos(angle)*self.pad[i].x + sin(angle)*self.pad[i].y
         ynew = -sin(angle)*self.pad[i].x + cos(angle)*self.pad[i].y
           self.pad[i].x = x + xnew
           self.pad[i].y = y + ynew
         self.pad[i].z += z
      cad.labels.append(cad_text(x,y,z,self.value,size=14))
      for i in range(len(self.labels)):
         xnew = cos(angle)*self.labels[i].x + sin(angle)*self.labels[i].y
         ynew = -sin(angle)*self.labels[i].x + cos(angle)*self.labels[i].y
           self.labels[i].x = x + xnew
           self.labels[i].y = y + ynew
         self.labels[i].z += z
           cad.labels.append(self.labels[i])
      pcb = pcb.add(self.shape)
      return pcb

def wire(pcb,width,*points):
   for i in range(1,len(points)):
      x0 = points[i-1].x
      y0 = points[i-1].y
      z0 = points[i-1].z
      x1 = points[i].x
      y1 = points[i].y
      z1 = points[i].z
      if (x0 < x1):
         pcb.board = add(pcb.board,cube(x0-width/2,x1+width/2,y0-width/2,y0+width/2,z0,z0))
      elif (x1 < x0):
         pcb.board = add(pcb.board,cube(x1-width/2,x0+width/2,y0-width/2,y0+width/2,z0,z0))
      if (y0 < y1):
         pcb.board = add(pcb.board,cube(x1-width/2,x1+width/2,y0-width/2,y1+width/2,z0,z0))
      elif (y1 < y0):
         pcb.board = add(pcb.board,cube(x1-width/2,x1+width/2,y1-width/2,y0+width/2,z0,z0))
   return pcb

#
# PCB library
#

#
# discretes
#

pad_0402 = cube(-.0175,.0175,-.014,.014,0,0)

class R_0402(part):
   #
   # 0402 resistor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_0402,-.0265,0,0)
      self.pad.append(point(-.0265,0,0))
      self.shape = add(self.shape,translate(pad_0402,.0265,0,0))
      self.pad.append(point(.0265,0,0))

pad_1206 = cube(-.032,.032,-.034,.034,0,0)

class R_1206(part):
   #
   # 1206 resistor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.06,0,0))
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.06,0,0))

class C_1206(part):
   #
   # 1206 capacitor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.06,0,0))
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.06,0,0))
```

```
pad_1210 = cube(-.032,.032,-.048,.048,0,0)

class L_1210(part):
   #
   # 1210 inductor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_1210,-.06,0,0)
      self.pad.append(point(-.06,0,0))
      self.shape = add(self.shape,translate(pad_1210,.06,0,0))
      self.pad.append(point(.06,0,0))

pad_choke = cube(-.06,.06,-.06,.06,0,0)

class choke(part):
   #
   # Panasonic ELLCTV
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_choke,-.177,-.177,0)
      self.pad.append(point(-.177,-.177,0))
      self.shape = add(self.shape,translate(pad_choke,.177,.177,0))
      self.pad.append(point(.177,.177,0))

#
# connectors
#

pad_MTA = cube(-.021,.021,-.041,.041,0,0)
pad_MTA_solder = cube(-.071,.071,-.041,.041,0,0)

class MTA_2(part):
   #
   # AMP 1445121-2
   # MTA .050 SMT 2-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1
      #
      self.shape = translate(pad_MTA,-.025,-.1,0)
      self.pad.append(point(-.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,.1,0))
      self.pad.append(point(.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'2',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.187,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.187,0,0))

class MTA_power(part):
   #
   # AMP 1445121-2
   # MTA .050 SMT 2-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: Gnd
      #
      self.shape = translate(pad_MTA,-.025,-.1,0)
      self.pad.append(point(-.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: Vcc
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,.1,0))
      self.pad.append(point(.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Vcc',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.187,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.187,0,0))

class MTA_3(part):
   #
   # AMP 1445121-3
   # MTA .050 SMT 3-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
```

```
      # pin 1: GND
      #
      self.shape = translate(pad_MTA,.05,.1,0)
      self.pad.append(point(.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: power
      #
      self.shape = add(self.shape,translate(pad_MTA,-.05,.1,0))
      self.pad.append(point(-.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'2',14))
      #
      # pin 3: data
      #
      self.shape = add(self.shape,translate(pad_MTA,0,-.1,0))
      self.pad.append(point(0,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'3',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.212,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.212,0,0))

class MTA_i0(part):
   #
   # AMP 1445121-3
   # MTA .050 SMT 3-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: GND
      #
      self.shape = translate(pad_MTA,.05,.1,0)
      self.pad.append(point(.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1\nGND',14))
      #
      # pin 2: power
      #
      self.shape = add(self.shape,translate(pad_MTA,-.05,.1,0))
      self.pad.append(point(-.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V',14))
      #
      # pin 3: data
      #
      self.shape = add(self.shape,translate(pad_MTA,0,-.1,0))
      self.pad.append(point(0,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'data',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.212,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.212,0,0))

class MTA_serial(part):
   #
   # AMP 1445121-4
   # MTA .050 SMT 4-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: Gnd
      #
      self.shape = translate(pad_MTA,-.075,-.1,0)
      self.pad.append(point(-.075,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: Tx
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,-.1,0))
      self.pad.append(point(.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Tx',14))
      #
      # pin 3: Rx
      #
      self.shape = add(self.shape,translate(pad_MTA,.075,.1,0))
      self.pad.append(point(.075,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Rx',14))
      #
      # pin 4: DTR
      #
      self.shape = add(self.shape,translate(pad_MTA,-.025,.1,0))
      self.pad.append(point(-.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DTR',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.237,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.237,0,0))

class MTA_ICP(part):
   #
   # AMP 1445121-5
   # MTA .050 SMT 4-pin vertical
   #
   def __init__(self,value=''):
```

```
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: MISO
      #
      self.shape = translate(pad_MTA,-.1,-.1,0)
      self.pad.append(point(-.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'MISO',14))
      #
      # pin 2: GND
      #
      self.shape = add(self.shape,translate(pad_MTA,0,-.1,0))
      self.pad.append(point(0,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))
      #
      # pin 3: MOSI
      #
      self.shape = add(self.shape,translate(pad_MTA,.1,-.1,0))
      self.pad.append(point(.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'MOSI',14))
      #
      # pin 4: -RESET
      #
      self.shape = add(self.shape,translate(pad_MTA,.05,.1,0))
      self.pad.append(point(.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'-RESET',14))
      #
      # pin 5: SCK
      #
      self.shape = add(self.shape,translate(pad_MTA,-.05,.1,0))
      self.pad.append(point(-.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'SCK',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.262,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.262,0,0))

class power_65mm(part):
   #
   # CUI PJ1-023-SMT
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: power
      #
      self.shape = cube(.433,.512,-.047,.047,0,0)
      self.pad.append(point(.467,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'P',14))
      #
      # pin 2: ground
      #
      self.shape = add(self.shape,cube(.285,.423,-.189,-.098,0,0))
      self.pad.append(point(.354,-.144,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 3: contact
      #
      self.shape = add(self.shape,cube(.325,.463,.098,.189,0,0))
      self.pad.append(point(.394,.144,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,cube(.108,.246,-.169,-.110,0,0))
      self.shape = add(self.shape,cube(.069,.207,.110,.169,0,0))

pad_stereo_2_5mm = cube(-.03,.03,-.05,.05,0,0)

class stereo_2_5mm(part):
   #
   # CUI SJ1-2533-SMT
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: base
      #
      self.shape = translate(pad_stereo_2_5mm,-.130,-.16,0)
      self.pad.append(point(-.130,-.149,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'base',14))
      #
      # pin 2: tip
      #
      self.shape = add(self.shape,translate(pad_stereo_2_5mm,.197,.15,0))
      self.pad.append(point(.197,.141,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'tip',14))
      #
      # pin 3: middle
      #
      self.shape = add(self.shape,translate(pad_stereo_2_5mm,-.012,-.16,0))
      self.pad.append(point(-.012,-.149,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'middle',14))

pad_Molex = cube(-.0155,.0155,-.0265,.0265,0,0)
```

```
pad_Molex_solder = cube(-.055,.055,-.065,.065,0,0)

class Molex_serial(part):
   #
   # Molex 53261-0471
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: Rx
      #
      self.shape = translate(pad_Molex,-.075,.064,0)
      self.pad.append(point(-.075,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Rx',14))
      #
      # pin 2: Tx
      #
      self.shape = add(self.shape,translate(pad_Molex,-.025,.064,0))
      self.pad.append(point(-.025,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Tx',14))
      #
      # pin 3: DTR
      #
      self.shape = add(self.shape,translate(pad_Molex,.025,.064,0))
      self.pad.append(point(.025,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DTR',14))
      #
      # pin 4: GND
      #
      self.shape = add(self.shape,translate(pad_Molex,.075,.064,0))
      self.pad.append(point(.075,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_Molex_solder,-.16,-.065,0))
      self.shape = add(self.shape,translate(pad_Molex_solder,.16,-.065,0))

#
# switches
#

pad_button_6mm = cube(-.04,.04,-.03,.03,0,0)

class button_6mm(part):
   #
   # Omron 6mm pushbutton
   # B3SN-3112P
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # left 1
      #
      self.shape = translate(pad_button_6mm,-.125,.08,0)
      self.pad.append(point(-.125,.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'L1',14))
      #
      # right 1
      #
      self.shape = add(self.shape,translate(pad_button_6mm,-.125,-.08,0))
      self.pad.append(point(-.125,-.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'R1',14))
      #
      # right 2
      #
      self.shape = add(self.shape,translate(pad_button_6mm,.125,-.08,0))
      self.pad.append(point(.125,-.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'R2',14))
      #
      # left 2
      #
      self.shape = add(self.shape,translate(pad_button_6mm,.125,.08,0))
      self.pad.append(point(.125,.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'L2',14))

#
# crystals and resonators
#

pad_XTAL = cube(-.016,.016,-.077,.077,0,0)

class XTAL(part):
   #
   # Panasonic EFOBM series
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # left
      #
      self.shape = translate(pad_XTAL,-.053,0,0)
      self.pad.append(point(-.053,0,0))
      #
      # ground
```

```
      #
      self.shape = add(self.shape,translate(pad_XTAL,0,0,0))
      self.pad.append(point(0,0,0))
      #
      # right
      #
      self.shape = add(self.shape,translate(pad_XTAL,.053,0,0))
      self.pad.append(point(.053,0,0))

#
# diodes, transistors, regulators
#

class D_1206(part):
   #
   # 1206 diode
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # anode
      #
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A',14))
      #
      # cathode
      #
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))

class LED_1206(part):
   #
   # 1206 LED
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # anode
      #
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A',14))
      #
      # cathode
      #
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))

class phototransistor_1206(part):
   #
   # 1206 phototransistor
   # OPTEK 520,521
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # collector
      #
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))
      #
      # emitter
      #
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'E',14))

pad_SOD_123 = cube(-.02,.02,-.024,.024,0,0)

class D_SOD_123(part):
   #
   # SOD-123 diode
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # anode
      #
      self.shape = translate(pad_SOD_123,-.07,0,0)
      self.pad.append(point(-.07,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A',14))
      #
      # cathode
      #
      self.shape = add(self.shape,translate(pad_SOD_123,.07,0,0))
      self.pad.append(point(.07,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))

pad_SOT23 = cube(-.015,.015,-.02,.02,0,0)
```

```
class NMOSFET_SOT23(part):
   #
   # Fairchild NDS355AN
   #
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: gate
      #
      self.shape = translate(pad_SOT23,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 2: source
      #
      self.shape = add(self.shape,translate(pad_SOT23,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'S',14))
      #
      # pin 3: drain
      #
      self.shape = add(self.shape,translate(pad_SOT23,0,.045,0))
      self.pad.append(point(0,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'D',14))

class PMOSFET_SOT23(part):
   #
   # Fairchild NDS356AP
   #
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: gate
      #
      self.shape = translate(pad_SOT23,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 2: source
      #
      self.shape = add(self.shape,translate(pad_SOT23,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'S',14))
      #
      # pin 3: drain
      #
      self.shape = add(self.shape,translate(pad_SOT23,0,.045,0))
      self.pad.append(point(0,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'D',14))

class regulator_SOT23(part):
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: output
      #
      self.shape = translate(pad_SOT23,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: input
      #
      self.shape = add(self.shape,translate(pad_SOT23,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'in',14))
      #
      # pin 3: ground
      #
      self.shape = add(self.shape,translate(pad_SOT23,0,.045,0))
      self.pad.append(point(0,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'gnd',14))

pad_SOT223 = cube(-.02,.02,-.03,.03,0,0)
pad_SOT223_ground = cube(-.065,.065,-.03,.03,0,0)

class regulator_SOT223(part):
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: input
```

```
      #
      self.shape = translate(pad_SOT223,-.09,-.12,0)
      self.pad.append(point(-.09,-.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1I',14))
      #
      # pin 2: ground
      #
      self.shape = add(self.shape,translate(pad_SOT223,0,-.12,0))
      self.pad.append(point(0,-.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 3: output
      #
      self.shape = add(self.shape,translate(pad_SOT223,.09,-.12,0))
      self.pad.append(point(.09,-.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'O',14))
      #
      # pin 4: ground
      #
      self.shape = add(self.shape,translate(pad_SOT223_ground,0,.12,0))
      self.pad.append(point(0,.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))

pad_SM8 = cube(-.035,.035,-.016,.016,0,0)

class H_bridge_SM8(part):
   #
   # Zetex ZXMHC3A01T8
   #
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      d = .13
      #
      # pin 1: G3 (right N gate)
      #
      self.shape = translate(pad_SM8,-d,.09,0)
      self.pad.append(point(-d,.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GRN',14))
      #
      # pin 2: S2 S3 (N source)
      #
      self.shape = add(self.shape,translate(pad_SM8,-d,.03,0))
      self.pad.append(point(-d,.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'SN',14))
      #
      # pin 3: G2 (left N gate)
      #
      self.shape = add(self.shape,translate(pad_SM8,-d,-.03,0))
      self.pad.append(point(-d,-.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GLN',14))
      #
      # pin 4: G1 (left P gate)
      #
      self.shape = add(self.shape,translate(pad_SM8,-d,-.09,0))
      self.pad.append(point(-d,-.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GLP',14))
      #
      # pin 5: D1 D2 (left drain)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,-.09,0))
      self.pad.append(point(d,-.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DL',14))
      #
      # pin 6: S1 S4 (P source)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,-.03,0))
      self.pad.append(point(d,-.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'SP',14))
      #
      # pin 7: D3 D4 (right drain)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,.03,0))
      self.pad.append(point(d,.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DR',14))
      #
      # pin 8: G4 (right N gate)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,.09,0))
      self.pad.append(point(d,.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GRP',14))

#
# ICs
#

pad_SOT23_5 = cube(-.01,.01,-.02,.02,0,0)

class op_amp_SOT23_5(part):
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
```

```
      # pin 1: output
      #
      self.shape = translate(pad_SOT23_5,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'O',14))
      #
      # pin 2: V-
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,0,-.045,0))
      self.pad.append(point(0,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V-',14))
      #
      # pin 3: I+
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'I+',14))
      #
      # pin 4: I-
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,.0375,.045,0))
      self.pad.append(point(.0375,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'I-',14))
      #
      # pin 5: V+
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,-.0375,.045,0))
      self.pad.append(point(-.0375,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V+',14))

pad_SOICN = cube(-.035,.035,-.015,.015,0,0)

class op_amp_SOICN(part):
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: A out
      #
      self.shape = translate(pad_SOICN,-.12,.075,0)
      self.pad.append(point(-.12,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1 Ao',14))
      #
      # pin 2: A-
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.025,0))
      self.pad.append(point(-.12,.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A-',14))
      #
      # pin 3: A+
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.025,0))
      self.pad.append(point(-.12,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A+',14))
      #
      # pin 4: V-
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.075,0))
      self.pad.append(point(-.12,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V-',14))
      #
      # pin 5: B+
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.075,0))
      self.pad.append(point(.12,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'B+',14))
      #
      # pin 6: B-
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.025,0))
      self.pad.append(point(.12,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'B-',14))
      #
      # pin 7: B out
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.025,0))
      self.pad.append(point(.12,.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Bo',14))
      #
      # pin 8: V+
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.075,0))
      self.pad.append(point(.12,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V+',14))

pad_SOIC = cube(-.043,.043,-.015,.015,0,0)

class ATtiny45_SOIC(part):
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: PB5/dW/ADC0/-RESET/PCINT5
      #
      self.shape = translate(pad_SOIC,-.14,.075,0)
      self.pad.append(point(-.14,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: PB3/ADC3/-OC1B/CLKI/XTAL1/PCINT3
```

```
        #
        self.shape = add(self.shape,translate(pad_SOIC,-.14,.025,0))
        self.pad.append(point(-.14,.025,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB3',14))
        #
        # pin 3: PB4/ADC2/OC1B/CLKO/XTAL2/PCINT4
        #
        self.shape = add(self.shape,translate(pad_SOIC,-.14,-.025,0))
        self.pad.append(point(-.14,-.025,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB4',14))
        #
        # pin 4: GND
        #
        self.shape = add(self.shape,translate(pad_SOIC,-.14,-.075,0))
        self.pad.append(point(-.14,-.075,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))
        #
        # pin 5: PB0/MOSI/DI/SDA/AIN0/OC0A/-OC1A/AREF/PCINT0
        #
        self.shape = add(self.shape,translate(pad_SOIC,.14,-.075,0))
        self.pad.append(point(.14,-.075,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 6: PB1/MISO/DO/AIN1/OC0B/OC1A/PCINT1
        #
        self.shape = add(self.shape,translate(pad_SOIC,.14,-.025,0))
        self.pad.append(point(.14,-.025,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB1',14))
        #
        # pin 7: PB2/SCK/USCK/SCL/ADC1/T0/INT0/PCINT2
        #
        self.shape = add(self.shape,translate(pad_SOIC,.14,.025,0))
        self.pad.append(point(.14,.025,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB2',14))
        #
        # pin 8: VCC
        #
        self.shape = add(self.shape,translate(pad_SOIC,.14,.075,0))
        self.pad.append(point(.14,.075,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'VCC',14))

class ATtiny44_SOICN(part):
   def __init__(self,value=''):
        self.value = value
        self.pad = [point(0,0,0)]
        self.labels = []
        #
        # pin 1: VCC
        #
        self.shape = translate(pad_SOICN,-.12,.15,0)
        self.pad.append(point(-.12,.15,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
        #
        # pin 2: PB0/XTAL1/PCINT8
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 3: PB1/XTAL2/PCINT9
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.050,0))
        self.pad.append(point(-.12,.05,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB1',14))
        #
        # pin 4: PB3/dW/-RESET/PCINT11
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,0,0))
        self.pad.append(point(-.12,0,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB3',14))
        #
        # pin 5: PB2/CKOUT/OC0A/INT0/PCINT10
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,-.05,0))
        self.pad.append(point(-.12,-.05,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB2',14))
        #
        # pin 6: PA7/ADC7/OC0B/ICP/PCINT7
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,-.1,0))
        self.pad.append(point(-.12,-.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA7',14))
        #
        # pin 7: PA6/ADC6/MOSI/SDA/OC1A/PCINT6
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,-.15,0))
        self.pad.append(point(-.12,-.15,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA6',14))
        #
        # pin 8: PA5/ADC5/DO/MISO/OC1B/PCINT5
        #
        self.shape = add(self.shape,translate(pad_SOICN,.12,-.15,0))
        self.pad.append(point(.12,-.15,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA5',14))
        #
        # pin 9: PA4/ADC4/USCK/SCL/T1/PCINT4
        #
        self.shape = add(self.shape,translate(pad_SOICN,.12,-.1,0))
        self.pad.append(point(.12,-.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA4',14))
        #
```

```
      # pin 10: PA3/ADC3/T0/PCINT3
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.05,0))
      self.pad.append(point(.12,-.05,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA3',14))
      #
      # pin 11: PA2/ADC2/AIN1/PCINT2
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,0,0))
      self.pad.append(point(.12,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA2',14))
      #
      # pin 12: PA1/ADC1/AIN0/PCINT1
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.050,0))
      self.pad.append(point(.12,.05,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA1',14))
      #
      # pin 13: PA0/ADC0/AREF/PCINT0
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.1,0))
      self.pad.append(point(.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA0',14))
      #
      # pin 14: GND
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.15,0))
      self.pad.append(point(.12,.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))

pad_TQFP = cube(-.043,.043,-.015,.015,0,0)

class ATmega88_TQFP(part):
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []

      #
      # pin 1: PD3/PCINT19/OC2B/INT1
      #
      self.shape = translate(pad_SOICN,-.12,.15,0)
      self.pad.append(point(-.12,.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: PD4/PCINT20/XCK/T0
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 3: GND
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 4: VCC
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 5: GND
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 6: VCC
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 7: PB6/PCINT6/XTAL1/TOSC1
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 8: PB7/PCINT7/XTAL2/TOSC2
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

      #
      # pin 9: PD5/PCINT21/OC0B/T1
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 10: PD6/PCINT22/OC0A/AIN0
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 11: PD7/PCINT23/AIN1
```

```
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 12: PB0/PCINT0/CLKO/ICP1
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 13: PB1/PCINT1/OC1A
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 14: PB2/PCINT2/-SS/OC1B
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 15: PB3/PCINT3/OC2A/MOSI
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 16: PB4/PCINT4/MISO
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

#
# pin 17: PB5/SCK/PCINT5
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 18: AVCC
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 19: ADC6
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 20: AREF
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 21: GND
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 22: ADC7
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 23: PC0/ADC0/PCINT8
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 24: PC1/ADC1/PCINT9
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

#
# pin 25: PC2/ADC2/PCINT10
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 26: PC2/ADC3/PCINT11
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 27: PC4/ADC4/SDA/PCINT12
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
```

```
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 28: PC5/ADC5/SCL/PCINT13
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 29: PC6/-RESET/PCINT14
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 30: PD0/RXD/PCINT16
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 31: PD1/TXD/PCINT17
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 32: PD2/INT0/PCINT18
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

#
# graphics
#

class CBA(part):
   def __init__(self,r=.02):
      self.value = ''
      self.pad = [point(0,0,0)]
      self.labels = []
      d = 3*r
      self.shape = circle(0,0,r)
      self.shape = add(self.shape,translate(circle(0,0,r),-d,d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),-d,0,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),-d,-d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),0,-d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),d,-d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),d,0,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),d,d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),0,d,0))

class hello(part):
   #
   # HELLO
   #
   def __init__(self,w,z):
      self.value = ''
      self.pad = [point(0,0,0)]
      self.labels = []
      # H
      self.shape = rectangle(-.01,.01,0,.1)
      self.shape = add(self.shape,rectangle(0,.05,.04,.06))
      self.shape = add(self.shape,rectangle(.04,.06,0,.1))
      # E
      self.shape = add(self.shape,rectangle(.08,.1,0,.1))
      self.shape = add(self.shape,rectangle(.1,.14,0,.02))
      self.shape = add(self.shape,rectangle(.1,.13,.04,.06))
      self.shape = add(self.shape,rectangle(.1,.14,.08,.1))
      # L
      self.shape = add(self.shape,rectangle(.16,.18,0,.1))
      self.shape = add(self.shape,rectangle(.18,.21,0,.02))
      # L
      self.shape = add(self.shape,rectangle(.23,.25,0,.1))
      self.shape = add(self.shape,rectangle(.25,.28,0,.02))
      # 0
      self.shape = add(self.shape,rectangle(.3,.32,0,.1))
      self.shape = add(self.shape,rectangle(.32,.345,0,.02))
      self.shape = add(self.shape,rectangle(.32,.345,.08,.1))
      self.shape = add(self.shape,rectangle(.345,.365,0,.1))

#
# define board
#

x0 = 1
y0 = 1
width = 1.18
height = .86
z = -.005
w = .018
mask = .004

pcb = PCB(x0,y0,width,height,mask)

IC1 = ATtiny45_SOIC('IC1\nt45')
pcb = IC1.add(pcb,x0+.74,y0+.51,z,angle=-90)

R1 = R_1206('R1\n10k')
pcb = R1.add(pcb,IC1.pad[1].x-.09,IC1.y,z,angle=90)
```

```
pcb = wire(pcb,w,
   IC1.pad[1],
   R1.pad[2])

pcb = wire(pcb,w,
   IC1.pad[8],
   R1.pad[1])

IC2 = regulator_SOT23('IC2\n5V')
pcb = IC2.add(pcb,R1.x-.155,R1.y,z,angle=90)

pcb = wire(pcb,w,
   IC2.pad[3],
   IC1.pad[4])

C1 = C_1206('C1\n3.3uF')
pcb = C1.add(pcb,IC2.x,IC1.pad[8].y,z)

pcb = wire(pcb,w,
   C1.pad[1],
   IC2.pad[1])

pcb = wire(pcb,w,
   IC2.pad[3],
   C1.pad[2])

pcb = wire(pcb,w,
   C1.pad[1],
   point(C1.pad[1].x,IC1.pad[8].y+.1,z),
   IC1.pad[8])

J1 = MTA_serial('J1')
pcb = J1.add(pcb,IC2.x-.26,IC1.y-.01,z,angle=-90)

pcb = wire(pcb,w,
   IC2.pad[3],
   point(IC2.pad[3].x,IC1.pad[1].y,z),
   J1.pad[1])

pcb = wire(pcb,w,
   J1.pad[4],
   IC2.pad[2])

pcb = wire(pcb,w,
   J1.pad[3],
   point(J1.pad[2].x,IC1.pad[7].y+.15,z),
   IC1.pad[7])

J2 = MTA_ICP('J2')
pcb = J2.add(pcb,IC1.x+.28,IC1.y,z,angle=90)

pcb = wire(pcb,w,
   IC1.pad[6],
   point(IC1.pad[6].x,IC1.pad[6].y+.1,z),
   point(J2.pad[1].x-.03,J2.pad[1].y,z),
   J2.pad[1])

pcb = wire(pcb,w,
   J2.pad[2],
   IC1.pad[4])

pcb = wire(pcb,w,
   J2.pad[3],
   point(J2.x,J2.pad[5].y,z),
   IC1.pad[5])

pcb = wire(pcb,w,
   IC1.pad[7],
   point(IC1.pad[7].x,IC1.pad[7].y+.15,z),
   point(J2.pad[1].x+.02,J2.pad[1].y+.05,z),
   J2.pad[5])

J3 = MTA_3('J3')
pcb = J3.add(pcb,x0+.525,y0+.19,z)

pcb = wire(pcb,w,
   J3.pad[1],
   IC1.pad[2])

pcb = wire(pcb,w,
   J3.pad[2],
   IC2.pad[3])

pcb = wire(pcb,w,
   J2.pad[4],
   point(J2.pad[4].x,y0+.025,z),
   point(J3.pad[2].x-.05,J3.y,z),
   point(J3.x,IC1.pad[1].y,z),
   IC1.pad[1])

R2 = R_1206('R2\n1M')
pcb = R2.add(pcb,J3.x+.35,J3.y-.03,z,angle=90)

pcb = wire(pcb,w,
   IC1.pad[2],
   point(IC1.pad[2].x,R2.pad[1].y+.04,z),
   R2.pad[1])

pcb = wire(pcb,w,
   IC1.pad[3],
   point(IC1.pad[3].x,IC1.pad[3].y-.075,z),
```

```
    point(R2.x+.07,R2.pad[2].y,z),
    R2.pad[2])

pcb = wire(pcb,w,
    J3.pad[3],
    R2.pad[2])

H1 = hello(w,z)
pcb = H1.add(pcb,x0+.03,y0+.02,z)

#
# assign board and exterior to cad.function for milling
#
# use single-sided FR2
# use 1/64 end-mill
# set tool diameter = .0156
# set xy, z speed = 4
# set # contours = -1
#

cad.function = add(pcb.board,pcb.exterior)

#
# uncomment to export traces
#

#cad.function = pcb.board

#
# uncomment to export exterior
#

#cad.function = pcb.exterior

#
# uncomment to export solder mask
#

#cad.function = pcb.mask

#
# uncomment to mill out board
#
# use 1/32 end-mill
# set tool diameter = .0312
# set xy, z speed = .5
# set # contours = 1
#

#cad.function = pcb.interior
#z = -.065

#
# define limits and parameters
#

d = 0.05
cad.xmin = x0-d # min x to render
cad.xmax = x0+width+d # max x to render
cad.ymin = y0-d # min y to render
cad.ymax = y0+height+d # max y to render
cad.zmin = z
cad.zmax = .050
dpi = 200 # low resolution for previewing
#dpi = 500 # high resolution for machining
nxy = int(dpi*(cad.xmax-cad.xmin))
cad.nx = int(dpi*(cad.xmax-cad.xmin)) # x points to render
cad.ny = int(dpi*(cad.ymax-cad.ymin)) # y points to render
cad.nz = 1
cad.inches_per_unit = 1.0 # use inch units
cad.view('xy') # 2D view
```

```
#
# hello.mic.45.cad
#
# electret microphone hello-world
#
# Neil Gershenfeld
# CBA MIT 10/28/07
#
# (c) Massachusetts Institute of Technology 2007
# Permission granted for experimental and personal use;
# license for commercial sale available from MIT.
#

#
# define shapes and transformation
#
# circle(x0, y0, r)
# cylinder(x0, y0, z0, z1, r)
# cone(x0, y0, z0, z1, r0)
# sphere(x0, y0, z0, r)
# torus(x0, y0, z0, r0, r1)
# rectangle(x0, x1, y0, y1)
# cube(x0, x1, y0, y1, z0, z1)
# triangle(x0, y0, x1, y1, x2, y2) (points in clockwise order)
# pyramid(x0, x1, y0, y1, z0, z1)
# function(Z_of_XY)
# functions(upper_Z_of_XY,lower_Z_of_XY)
# add(part1, part2)
# subtract(part1, part2)
# intersect(part1, part2)
# move(part,dx,dy)
# translate(part,dx,dy,dz)
# rotate(part, angle)
# rotate_x(part, angle)
# rotate_y(part, angle)
# rotate_z(part, angle)
# rotate_90(part)
# rotate_180(part)
# rotate_270(part)
# reflect_x(part)
# reflect_y(part)
# reflect_z(part)
# reflect_xy(part)
# reflect_xz(part)
# reflect_yz(part)
# scale_x(part, x0, sx)
# scale_y(part, y0, sy)
# scale_z(part, z0, sz)
# scale_xy(part, x0, y0, sxy)
# scale_xyz(part, x0, y0, z0, sxyz)
# coscale_x_y(part, x0, y0, y1, angle0, angle1, amplitude, offset)
# coscale_x_z(part, x0, z0, z1, angle0, angle1, amplitude, offset)
# coscale_xy_z(part, x0, y0, z0, z1, angle0, angle1, amplitude, offset)
# taper_x_y(part, x0, y0, y1, s0, s1)
# taper_x_z(part, x0, z0, z1, s0, s1)
# taper_xy_z(part, x0, y0, z0, z1, s0, s1)
# shear_x_y(part, y0, y1, dx0, dx1)
# shear_x_z(part, z0, z1, dx0, dx1)
# (more to come)

def circle(x0, y0, r):
   part = "(((X-x0)**2 + (Y-y0)**2) <= r**2)"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'r',str(r))
   return part

def cylinder(x0, y0, z0, z1, r):
   part = "(((X-x0)**2 + (Y-y0)**2 <= r**2) & (Z >= z0) & (Z <= z1))"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'r',str(r))
   return part

def cone(x0, y0, z0, z1, r0):
   part = cylinder(x0, y0, z0, z1, r0)
   part = taper_xy_z(part, x0, y0, z0, z1, 1.0, 0.0)
   return part

def sphere(x0, y0, z0, r):
   part = "(((X-x0)**2 + (Y-y0)**2 + (Z-z0)**2) <= r**2)"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'r',str(r))
   return part

def torus(x0, y0, z0, r0, r1):
   part = "(((r0 - sqrt((X-x0)**2 + (Y-y0)**2))**2 + (Z-z0)**2) <= r1**2)"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'r0',str(r0))
   part = replace(part,'r1',str(r1))
   return part

def rectangle(x0, x1, y0, y1):
   part = "((X >= x0) & (X <= x1) & (Y >= y0) & (Y <= y1))"
   part = replace(part,'x0',str(x0))
```

```
   part = replace(part,'x1',str(x1))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   return part

def cube(x0, x1, y0, y1, z0, z1):
   part = "((X >= x0) & (X <= x1) & (Y >= y0) & (Y <= y1) & (Z >= z0) & (Z <= z1))"
   part = replace(part,'x0',str(x0))
   part = replace(part,'x1',str(x1))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   return part

def triangle(x0, y0, x1, y1, x2, y2): # points in clockwise order
   part = "((((y1-y0)*(X-x0)-(x1-x0)*(Y-y0)) >= 0) & (((y2-y1)*(X-x1)-(x2-x1)*(Y-y1)) >= 0) & (((y0-y2)*(X-x2)-(x0-x2)*(Y-y2)) >= 0))"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'x1',str(x1))
   part = replace(part,'y1',str(y1))
   part = replace(part,'x2',str(x2))
   part = replace(part,'y2',str(y2))
   return part

def pyramid(x0, x1, y0, y1, z0, z1):
   part = cube(x0, x1, y0, y1, z0, z1)
   part = taper_xy_z(part, (x0+x1)/2., (y0+y1)/2., z0, z1, 1.0, 0.0)
   return part

def function(Z_of_XY):
   part = '(Z <= '+Z_of_XY+')'
   return part

def functions(upper_Z_of_XY,lower_Z_of_XY):
   part = '(Z <= '+upper_Z_of_XY+') & (Z >= '+lower_Z_of_XY+')'
   return part

def add(part1, part2):
   part = "part1 | part2"
   part = replace(part,'part1',part1)
   part = replace(part,'part2',part2)
   return part

def subtract(part1, part2):
   part = "(part1) & ~(part2)"
   part = replace(part,'part1',part1)
   part = replace(part,'part2',part2)
   return part

def intersect(part1, part2):
   part = "(part1) & (part2)"
   part = replace(part,'part1',part1)
   part = replace(part,'part2',part2)
   return part

def move(part,dx,dy):
   part = replace(part,'X','(X-'+str(dx)+')')
   part = replace(part,'Y','(Y-'+str(dy)+')')
   return part

def translate(part,dx,dy,dz):
   part = replace(part,'X','(X-'+str(dx)+')')
   part = replace(part,'Y','(Y-'+str(dy)+')')
   part = replace(part,'Z','(Z-'+str(dz)+')')
   return part

def rotate(part, angle):
   angle = angle*pi/180
   part = replace(part,'X','(cos(angle)*X+sin(angle)*y)')
   part = replace(part,'Y','(-sin(angle)*X+cos(angle)*y)')
   part = replace(part,'y','Y')
   part = replace(part,'angle',str(angle))
   return part

def rotate_x(part, angle):
   angle = angle*pi/180
   part = replace(part,'Y','(cos(angle)*Y+sin(angle)*z)')
   part = replace(part,'Z','(-sin(angle)*Y+cos(angle)*z)')
   part = replace(part,'z','Z')
   part = replace(part,'angle',str(angle))
   return part

def rotate_y(part, angle):
   angle = angle*pi/180
   part = replace(part,'X','(cos(angle)*X+sin(angle)*z)')
   part = replace(part,'Z','(-sin(angle)*X+cos(angle)*z)')
   part = replace(part,'z','Z')
   part = replace(part,'angle',str(angle))
   return part

def rotate_z(part, angle):
   angle = angle*pi/180
   part = replace(part,'X','(cos(angle)*X+sin(angle)*y)')
   part = replace(part,'Y','(-sin(angle)*X+cos(angle)*y)')
   part = replace(part,'y','Y')
   part = replace(part,'angle',str(angle))
   return part

def rotate_90(part):
   part = reflect_xy(part)
```

```
   part = reflect_y(part)
   return part

def rotate_180(part):
   part = reflect_xy(part)
   part = reflect_y(part)
   part = reflect_xy(part)
   part = reflect_y(part)
   return part

def rotate_270(part):
   part = reflect_xy(part)
   part = reflect_y(part)
   part = reflect_xy(part)
   part = reflect_y(part)
   part = reflect_xy(part)
   part = reflect_y(part)
   return part

def reflect_x(part):
   part = replace(part,'X','-X')
   return part

def reflect_y(part):
   part = replace(part,'Y','-Y')
   return part

def reflect_z(part):
   part = replace(part,'Z','-Z')
   return part

def reflect_xy(part):
   part = replace(part,'X','temp')
   part = replace(part,'Y','X')
   part = replace(part,'temp','Y')
   return part

def reflect_xz(part):
   part = replace(part,'X','temp')
   part = replace(part,'Z','X')
   part = replace(part,'temp','Z')
   return part

def reflect_yz(part):
   part = replace(part,'Y','temp')
   part = replace(part,'Z','Y')
   part = replace(part,'temp','Z')
   return part

def scale_x(part, x0, sx):
   part = replace(part,'X','(x0 + (X-x0)/sx)')
   part = replace(part,'x0',str(x0))
   part = replace(part,'sx',str(sx))
   return part

def scale_y(part, y0, sy):
   part = replace(part,'Y','(y0 + (Y-y0)/sy)')
   part = replace(part,'y0',str(y0))
   part = replace(part,'sy',str(sy))
   return part

def scale_z(part, z0, sz):
   part = replace(part,'Z','(z0 + (Z-z0)/sz)')
   part = replace(part,'z0',str(z0))
   part = replace(part,'sz',str(sz))
   return part

def scale_xy(part, x0, y0, sxy):
   part = replace(part,'X','(x0 + (X-x0)/sxy)')
   part = replace(part,'Y','(y0 + (Y-y0)/sxy)')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'sxy',str(sxy))
   return part

def scale_xyz(part, x0, y0, z0, sxyz):
   part = replace(part,'X','(x0 + (X-x0)/sxyz)')
   part = replace(part,'Y','(y0 + (Y-y0)/sxyz)')
   part = replace(part,'Z','(z0 + (Z-z0)/sxyz)')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'sxyz',str(sxyz))
   return part

def coscale_x_y(part, x0, y0, y1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
   part = replace(part,'X','(x0 + (X-x0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Y-y0)/(y1-y0))))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

def coscale_x_z(part, x0, z0, z1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
```

```
   part = replace(part,'X','(x0 + (X-x0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0))))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

def coscale_xy_z(part, x0, y0, z0, z1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
   part = replace(part,'X','(x0 + (X-x0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0))))')
   part = replace(part,'Y','(y0 + (Y-y0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0))))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

def taper_x_y(part, x0, y0, y1, s0, s1):
   part = replace(part,'X','(x0 + (X-x0)*(y1-y0)/(s1*(Y-y0) + s0*(y1-Y)))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'s0',str(s0))
   part = replace(part,'s1',str(s1))
   return part

def taper_x_z(part, x0, z0, z1, s0, s1):
   part = replace(part,'X','(x0 + (X-x0)*(z1-z0)/(s1*(Z-z0) + s0*(z1-Z)))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'s0',str(s0))
   part = replace(part,'s1',str(s1))
   return part

def taper_xy_z(part, x0, y0, z0, z1, s0, s1):
   part = replace(part,'X','(x0 + (X-x0)*(z1-z0)/(s1*(Z-z0) + s0*(z1-Z)))')
   part = replace(part,'Y','(y0 + (Y-y0)*(z1-z0)/(s1*(Z-z0) + s0*(z1-Z)))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'s0',str(s0))
   part = replace(part,'s1',str(s1))
   return part

def shear_x_y(part, y0, y1, dx0, dx1):
   part = replace(part,'X','(X - dx0 - (dx1-dx0)*(Y-y0)/(y1-y0))')
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'dx0',str(dx0))
   part = replace(part,'dx1',str(dx1))
   return part

def shear_x_z(part, z0, z1, dx0, dx1):
   part = replace(part,'X','(X - dx0 - (dx1-dx0)*(Z-z0)/(z1-z0))')
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'dx0',str(dx0))
   part = replace(part,'dx1',str(dx1))
   return part

def coshear_x_z(part, z0, z1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
   part = replace(part,'X','(X - offset - amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0)))')
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

#
# PCB classes and definitions
#

cad.labels = []

class PCB:
   def __init__(self,x0,y0,width,height,mask):
      self.board = "False"
      self.interior = rectangle(x0,x0+width,y0,y0+height)
      self.exterior = subtract("True",rectangle(x0,x0+width,y0,y0+height))
      self.mask = "False"
   def add(self,part):
      self.board = add(self.board,part)
      self.mask = add(self.mask,move(part,-mask,mask))
      self.mask = add(self.mask,move(part,-mask,-mask))
      self.mask = add(self.mask,move(part,mask,mask))
      self.mask = add(self.mask,move(part,mask,-mask))
      return self
```

```
class part:
   def add(self,pcb,x,y,z=0,angle=0):
      self.x = x
      self.y = y
      self.z = z
      self.angle = angle
      if (angle == 90):
         self.shape = rotate_90(self.shape)
      elif (angle == 180):
         self.shape = rotate_180(self.shape)
      elif ((angle == 270) | (angle == -90)):
         self.shape = rotate_270(self.shape)
      angle = pi*angle/180
      self.shape = translate(self.shape,x,y,z)
      for i in range(len(self.pad)):
         xnew = cos(angle)*self.pad[i].x + sin(angle)*self.pad[i].y
         ynew = -sin(angle)*self.pad[i].x + cos(angle)*self.pad[i].y
           self.pad[i].x = x + xnew
           self.pad[i].y = y + ynew
         self.pad[i].z += z
      cad.labels.append(cad_text(x,y,z,self.value,size=14))
      for i in range(len(self.labels)):
         xnew = cos(angle)*self.labels[i].x + sin(angle)*self.labels[i].y
         ynew = -sin(angle)*self.labels[i].x + cos(angle)*self.labels[i].y
           self.labels[i].x = x + xnew
           self.labels[i].y = y + ynew
         self.labels[i].z += z
           cad.labels.append(self.labels[i])
      pcb = pcb.add(self.shape)
      return pcb

def wire(pcb,width,*points):
   for i in range(1,len(points)):
      x0 = points[i-1].x
      y0 = points[i-1].y
      z0 = points[i-1].z
      x1 = points[i].x
      y1 = points[i].y
      z1 = points[i].z
      if (x0 < x1):
         pcb.board = add(pcb.board,cube(x0-width/2,x1+width/2,y0-width/2,y0+width/2,z0,z0))
      elif (x1 < x0):
         pcb.board = add(pcb.board,cube(x1-width/2,x0+width/2,y0-width/2,y0+width/2,z0,z0))
      if (y0 < y1):
         pcb.board = add(pcb.board,cube(x1-width/2,x1+width/2,y0-width/2,y1+width/2,z0,z0))
      elif (y1 < y0):
         pcb.board = add(pcb.board,cube(x1-width/2,x1+width/2,y1-width/2,y0+width/2,z0,z0))
   return pcb

#
# PCB library
#

#
# discretes
#

pad_0402 = cube(-.0175,.0175,-.014,.014,0,0)

class R_0402(part):
   #
   # 0402 resistor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_0402,-.0265,0,0)
      self.pad.append(point(-.0265,0,0))
      self.shape = add(self.shape,translate(pad_0402,.0265,0,0))
      self.pad.append(point(.0265,0,0))

pad_1206 = cube(-.032,.032,-.034,.034,0,0)

class R_1206(part):
   #
   # 1206 resistor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.06,0,0))
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.06,0,0))

class C_1206(part):
   #
   # 1206 capacitor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.06,0,0))
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.06,0,0))
```

```
pad_1210 = cube(-.032,.032,-.048,.048,0,0)

class L_1210(part):
   #
   # 1210 inductor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_1210,-.06,0,0)
      self.pad.append(point(-.06,0,0))
      self.shape = add(self.shape,translate(pad_1210,.06,0,0))
      self.pad.append(point(.06,0,0))

pad_choke = cube(-.06,.06,-.06,.06,0,0)

class choke(part):
   #
   # Panasonic ELLCTV
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_choke,-.177,-.177,0)
      self.pad.append(point(-.177,-.177,0))
      self.shape = add(self.shape,translate(pad_choke,.177,.177,0))
      self.pad.append(point(.177,.177,0))

#
# connectors
#

pad_MTA = cube(-.021,.021,-.041,.041,0,0)
pad_MTA_solder = cube(-.071,.071,-.041,.041,0,0)

class MTA_2(part):
   #
   # AMP 1445121-2
   # MTA .050 SMT 2-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1
      #
      self.shape = translate(pad_MTA,-.025,-.1,0)
      self.pad.append(point(-.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,.1,0))
      self.pad.append(point(.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'2',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.187,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.187,0,0))

class MTA_power(part):
   #
   # AMP 1445121-2
   # MTA .050 SMT 2-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: Gnd
      #
      self.shape = translate(pad_MTA,-.025,-.1,0)
      self.pad.append(point(-.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: Vcc
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,.1,0))
      self.pad.append(point(.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Vcc',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.187,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.187,0,0))

class MTA_i0(part):
   #
   # AMP 1445121-3
   # MTA .050 SMT 3-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
```

```
      # pin 1: GND
      #
      self.shape = translate(pad_MTA,.05,.1,0)
      self.pad.append(point(.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1\nGND',14))
      #
      # pin 2: power
      #
      self.shape = add(self.shape,translate(pad_MTA,-.05,.1,0))
      self.pad.append(point(-.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V',14))
      #
      # pin 3: data
      #
      self.shape = add(self.shape,translate(pad_MTA,0,-.1,0))
      self.pad.append(point(0,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'data',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.212,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.212,0,0))

class MTA_serial(part):
   #
   # AMP 1445121-4
   # MTA .050 SMT 4-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: Gnd
      #
      self.shape = translate(pad_MTA,-.075,-.1,0)
      self.pad.append(point(-.075,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: Tx
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,-.1,0))
      self.pad.append(point(.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Tx',14))
      #
      # pin 3: Rx
      #
      self.shape = add(self.shape,translate(pad_MTA,.075,.1,0))
      self.pad.append(point(.075,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Rx',14))
      #
      # pin 4: DTR
      #
      self.shape = add(self.shape,translate(pad_MTA,-.025,.1,0))
      self.pad.append(point(-.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DTR',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.237,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.237,0,0))

class MTA_ICP(part):
   #
   # AMP 1445121-5
   # MTA .050 SMT 4-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: MISO
      #
      self.shape = translate(pad_MTA,-.1,-.1,0)
      self.pad.append(point(-.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'MISO',14))
      #
      # pin 2: GND
      #
      self.shape = add(self.shape,translate(pad_MTA,0,-.1,0))
      self.pad.append(point(0,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))
      #
      # pin 3: MOSI
      #
      self.shape = add(self.shape,translate(pad_MTA,.1,-.1,0))
      self.pad.append(point(.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'MOSI',14))
      #
      # pin 4: -RESET
      #
      self.shape = add(self.shape,translate(pad_MTA,.05,.1,0))
      self.pad.append(point(.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'-RESET',14))
      #
      # pin 5: SCK
      #
      self.shape = add(self.shape,translate(pad_MTA,-.05,.1,0))
      self.pad.append(point(-.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'SCK',14))
```

```
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.262,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.262,0,0))

class power_65mm(part):
   #
   # CUI PJ1-023-SMT
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: power
      #
      self.shape = cube(.433,.512,-.047,.047,0,0)
      self.pad.append(point(.467,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'P',14))
      #
      # pin 2: ground
      #
      self.shape = add(self.shape,cube(.285,.423,-.189,-.098,0,0))
      self.pad.append(point(.354,-.144,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 3: contact
      #
      self.shape = add(self.shape,cube(.325,.463,.098,.189,0,0))
      self.pad.append(point(.394,.144,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,cube(.108,.246,-.169,-.110,0,0))
      self.shape = add(self.shape,cube(.069,.207,.110,.169,0,0))

pad_stereo_2_5mm = cube(-.03,.03,-.05,.05,0,0)

class stereo_2_5mm(part):
   #
   # CUI SJ1-2533-SMT
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: base
      #
      self.shape = translate(pad_stereo_2_5mm,-.130,-.16,0)
      self.pad.append(point(-.130,-.149,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'base',14))
      #
      # pin 2: tip
      #
      self.shape = add(self.shape,translate(pad_stereo_2_5mm,.197,.15,0))
      self.pad.append(point(.197,.141,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'tip',14))
      #
      # pin 3: middle
      #
      self.shape = add(self.shape,translate(pad_stereo_2_5mm,-.012,-.16,0))
      self.pad.append(point(-.012,-.149,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'middle',14))

pad_Molex = cube(-.0155,.0155,-.0265,.0265,0,0)
pad_Molex_solder = cube(-.055,.055,-.065,.065,0,0)

class Molex_serial(part):
   #
   # Molex 53261-0471
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: Rx
      #
      self.shape = translate(pad_Molex,-.075,.064,0)
      self.pad.append(point(-.075,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Rx',14))
      #
      # pin 2: Tx
      #
      self.shape = add(self.shape,translate(pad_Molex,-.025,.064,0))
      self.pad.append(point(-.025,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Tx',14))
      #
      # pin 3: DTR
      #
      self.shape = add(self.shape,translate(pad_Molex,.025,.064,0))
      self.pad.append(point(.025,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DTR',14))
      #
      # pin 4: GND
      #
      self.shape = add(self.shape,translate(pad_Molex,.075,.064,0))
      self.pad.append(point(.075,.064,0))
```

```
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_Molex_solder,-.16,-.065,0))
      self.shape = add(self.shape,translate(pad_Molex_solder,.16,-.065,0))

#
# switches
#

pad_button_6mm = cube(-.04,.04,-.03,.03,0,0)

class button_6mm(part):
   #
   # Omron 6mm pushbutton
   # B3SN-3112P
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # left 1
      #
      self.shape = translate(pad_button_6mm,-.125,.08,0)
      self.pad.append(point(-.125,.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'L1',14))
      #
      # right 1
      #
      self.shape = add(self.shape,translate(pad_button_6mm,-.125,-.08,0))
      self.pad.append(point(-.125,-.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'R1',14))
      #
      # right 2
      #
      self.shape = add(self.shape,translate(pad_button_6mm,.125,-.08,0))
      self.pad.append(point(.125,-.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'R2',14))
      #
      # left 2
      #
      self.shape = add(self.shape,translate(pad_button_6mm,.125,.08,0))
      self.pad.append(point(.125,.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'L2',14))

#
# crystals and resonators
#

pad_XTAL = cube(-.016,.016,-.077,.077,0,0)

class XTAL(part):
   #
   # Panasonic EFOBM series
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # left
      #
      self.shape = translate(pad_XTAL,-.053,0,0)
      self.pad.append(point(-.053,0,0))
      #
      # ground
      #
      self.shape = add(self.shape,translate(pad_XTAL,0,0,0))
      self.pad.append(point(0,0,0))
      #
      # right
      #
      self.shape = add(self.shape,translate(pad_XTAL,.053,0,0))
      self.pad.append(point(.053,0,0))

#
# diodes, transistors, regulators
#

class D_1206(part):
   #
   # 1206 diode
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # anode
      #
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A',14))
      #
      # cathode
      #
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))
```

```
class LED_1206(part):
   #
   # 1206 LED
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # anode
      #
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A',14))
      #
      # cathode
      #
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))

class phototransistor_1206(part):
   #
   # 1206 phototransistor
   # OPTEK 520,521
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # collector
      #
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))
      #
      # emitter
      #
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'E',14))

pad_SOD_123 = cube(-.02,.02,-.024,.024,0,0)

class D_SOD_123(part):
   #
   # SOD-123 diode
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # anode
      #
      self.shape = translate(pad_SOD_123,-.07,0,0)
      self.pad.append(point(-.07,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A',14))
      #
      # cathode
      #
      self.shape = add(self.shape,translate(pad_SOD_123,.07,0,0))
      self.pad.append(point(.07,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))

pad_SOT23 = cube(-.015,.015,-.02,.02,0,0)

class NMOSFET_SOT23(part):
   #
   # Fairchild NDS355AN
   #
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: gate
      #
      self.shape = translate(pad_SOT23,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 2: source
      #
      self.shape = add(self.shape,translate(pad_SOT23,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'S',14))
      #
      # pin 3: drain
      #
      self.shape = add(self.shape,translate(pad_SOT23,0,.045,0))
      self.pad.append(point(0,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'D',14))

class PMOSFET_SOT23(part):
   #
```

```
   # Fairchild NDS356AP
   #
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: gate
      #
      self.shape = translate(pad_SOT23,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 2: source
      #
      self.shape = add(self.shape,translate(pad_SOT23,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'S',14))
      #
      # pin 3: drain
      #
      self.shape = add(self.shape,translate(pad_SOT23,0,.045,0))
      self.pad.append(point(0,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'D',14))

class regulator_SOT23(part):
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: output
      #
      self.shape = translate(pad_SOT23,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: input
      #
      self.shape = add(self.shape,translate(pad_SOT23,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'in',14))
      #
      # pin 3: ground
      #
      self.shape = add(self.shape,translate(pad_SOT23,0,.045,0))
      self.pad.append(point(0,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'gnd',14))

pad_SOT223 = cube(-.02,.02,-.03,.03,0,0)
pad_SOT223_ground = cube(-.065,.065,-.03,.03,0,0)

class regulator_SOT223(part):
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: input
      #
      self.shape = translate(pad_SOT223,-.09,-.12,0)
      self.pad.append(point(-.09,-.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1I',14))
      #
      # pin 2: ground
      #
      self.shape = add(self.shape,translate(pad_SOT223,0,-.12,0))
      self.pad.append(point(0,-.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 3: output
      #
      self.shape = add(self.shape,translate(pad_SOT223,.09,-.12,0))
      self.pad.append(point(.09,-.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'O',14))
      #
      # pin 4: ground
      #
      self.shape = add(self.shape,translate(pad_SOT223_ground,0,.12,0))
      self.pad.append(point(0,.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))

pad_SM8 = cube(-.035,.035,-.016,.016,0,0)

class H_bridge_SM8(part):
   #
   # Zetex ZXMHC3A01T8
   #
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
```

```
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      d = .13
      #
      # pin 1: G3 (right N gate)
      #
      self.shape = translate(pad_SM8,-d,.09,0)
      self.pad.append(point(-d,.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GRN',14))
      #
      # pin 2: S2 S3 (N source)
      #
      self.shape = add(self.shape,translate(pad_SM8,-d,.03,0))
      self.pad.append(point(-d,.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'SN',14))
      #
      # pin 3: G2 (left N gate)
      #
      self.shape = add(self.shape,translate(pad_SM8,-d,-.03,0))
      self.pad.append(point(-d,-.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GLN',14))
      #
      # pin 4: G1 (left P gate)
      #
      self.shape = add(self.shape,translate(pad_SM8,-d,-.09,0))
      self.pad.append(point(-d,-.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GLP',14))
      #
      # pin 5: D1 D2 (left drain)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,-.09,0))
      self.pad.append(point(d,-.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DL',14))
      #
      # pin 6: S1 S4 (P source)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,-.03,0))
      self.pad.append(point(d,-.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'SP',14))
      #
      # pin 7: D3 D4 (right drain)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,.03,0))
      self.pad.append(point(d,.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DR',14))
      #
      # pin 8: G4 (right N gate)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,.09,0))
      self.pad.append(point(d,.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GRP',14))

#
# ICs
#

pad_SOT23_5 = cube(-.01,.01,-.02,.02,0,0)

class op_amp_SOT23_5(part):
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: output
      #
      self.shape = translate(pad_SOT23_5,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'O',14))
      #
      # pin 2: V-
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,0,-.045,0))
      self.pad.append(point(0,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V-',14))
      #
      # pin 3: I+
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'I+',14))
      #
      # pin 4: I-
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,.0375,.045,0))
      self.pad.append(point(.0375,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'I-',14))
      #
      # pin 5: V+
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,-.0375,.045,0))
      self.pad.append(point(-.0375,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V+',14))

pad_SOICN = cube(-.035,.035,-.015,.015,0,0)

class op_amp_SOICN(part):
```

```
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: A out
      #
      self.shape = translate(pad_SOICN,-.12,.075,0)
      self.pad.append(point(-.12,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1 Ao',14))
      #
      # pin 2: A-
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.025,0))
      self.pad.append(point(-.12,.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A-',14))
      #
      # pin 3: A+
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.025,0))
      self.pad.append(point(-.12,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A+',14))
      #
      # pin 4: V-
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.075,0))
      self.pad.append(point(-.12,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V-',14))
      #
      # pin 5: B+
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.075,0))
      self.pad.append(point(.12,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'B+',14))
      #
      # pin 6: B-
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.025,0))
      self.pad.append(point(.12,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'B-',14))
      #
      # pin 7: B out
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.025,0))
      self.pad.append(point(.12,.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Bo',14))
      #
      # pin 8: V+
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.075,0))
      self.pad.append(point(.12,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V+',14))

pad_SOIC = cube(-.043,.043,-.015,.015,0,0)

class ATtiny45_SOIC(part):
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: PB5/dW/ADC0/-RESET/PCINT5
      #
      self.shape = translate(pad_SOIC,-.14,.075,0)
      self.pad.append(point(-.14,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: PB3/ADC3/-OC1B/CLKI/XTAL1/PCINT3
      #
      self.shape = add(self.shape,translate(pad_SOIC,-.14,.025,0))
      self.pad.append(point(-.14,.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB3',14))
      #
      # pin 3: PB4/ADC2/OC1B/CLKO/XTAL2/PCINT4
      #
      self.shape = add(self.shape,translate(pad_SOIC,-.14,-.025,0))
      self.pad.append(point(-.14,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB4',14))
      #
      # pin 4: GND
      #
      self.shape = add(self.shape,translate(pad_SOIC,-.14,-.075,0))
      self.pad.append(point(-.14,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))
      #
      # pin 5: PB0/MOSI/DI/SDA/AIN0/OC0A/-OC1A/AREF/PCINT0
      #
      self.shape = add(self.shape,translate(pad_SOIC,.14,-.075,0))
      self.pad.append(point(.14,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 6: PB1/MISO/DO/AIN1/OC0B/OC1A/PCINT1
      #
      self.shape = add(self.shape,translate(pad_SOIC,.14,-.025,0))
      self.pad.append(point(.14,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB1',14))
      #
      # pin 7: PB2/SCK/USCK/SCL/ADC1/T0/INT0/PCINT2
      #
      self.shape = add(self.shape,translate(pad_SOIC,.14,.025,0))
      self.pad.append(point(.14,.025,0))
```

```
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB2',14))
      #
      # pin 8: VCC
      #
      self.shape = add(self.shape,translate(pad_SOIC,.14,.075,0))
      self.pad.append(point(.14,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'VCC',14))

class ATtiny44_SOICN(part):
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: VCC
      #
      self.shape = translate(pad_SOICN,-.12,.15,0)
      self.pad.append(point(-.12,.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: PB0/XTAL1/PCINT8
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 3: PB1/XTAL2/PCINT9
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.050,0))
      self.pad.append(point(-.12,.05,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB1',14))
      #
      # pin 4: PB3/dW/-RESET/PCINT11
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,0,0))
      self.pad.append(point(-.12,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB3',14))
      #
      # pin 5: PB2/CKOUT/OC0A/INT0/PCINT10
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.05,0))
      self.pad.append(point(-.12,-.05,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB2',14))
      #
      # pin 6: PA7/ADC7/OC0B/ICP/PCINT7
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.1,0))
      self.pad.append(point(-.12,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA7',14))
      #
      # pin 7: PA6/ADC6/MOSI/SDA/OC1A/PCINT6
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.15,0))
      self.pad.append(point(-.12,-.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA6',14))
      #
      # pin 8: PA5/ADC5/DO/MISO/OC1B/PCINT5
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.15,0))
      self.pad.append(point(.12,-.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA5',14))
      #
      # pin 9: PA4/ADC4/USCK/SCL/T1/PCINT4
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.1,0))
      self.pad.append(point(.12,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA4',14))
      #
      # pin 10: PA3/ADC3/T0/PCINT3
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.05,0))
      self.pad.append(point(.12,-.05,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA3',14))
      #
      # pin 11: PA2/ADC2/AIN1/PCINT2
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,0,0))
      self.pad.append(point(.12,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA2',14))
      #
      # pin 12: PA1/ADC1/AIN0/PCINT1
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.050,0))
      self.pad.append(point(.12,.05,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA1',14))
      #
      # pin 13: PA0/ADC0/AREF/PCINT0
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.1,0))
      self.pad.append(point(.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA0',14))
      #
      # pin 14: GND
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.15,0))
      self.pad.append(point(.12,.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))

pad_TQFP = cube(-.043,.043,-.015,.015,0,0)

class ATmega88_TQFP(part):
```

```
def __init__(self,value=''):
   self.value = value
   self.pad = [point(0,0,0)]
   self.labels = []

   #
   # pin 1: PD3/PCINT19/OC2B/INT1
   #
   self.shape = translate(pad_SOICN,-.12,.15,0)
   self.pad.append(point(-.12,.15,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
   #
   # pin 2: PD4/PCINT20/XCK/T0
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 3: GND
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 4: VCC
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 5: GND
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 6: VCC
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 7: PB6/PCINT6/XTAL1/TOSC1
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 8: PB7/PCINT7/XTAL2/TOSC2
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

   #
   # pin 9: PD5/PCINT21/OC0B/T1
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 10: PD6/PCINT22/OC0A/AIN0
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 11: PD7/PCINT23/AIN1
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 12: PB0/PCINT0/CLKO/ICP1
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 13: PB1/PCINT1/OC1A
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 14: PB2/PCINT2/-SS/OC1B
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 15: PB3/PCINT3/OC2A/MOSI
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 16: PB4/PCINT4/MISO
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
```

```
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

        #
        # pin 17: PB5/SCK/PCINT5
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 18: AVCC
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 19: ADC6
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 20: AREF
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 21: GND
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 22: ADC7
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 23: PC0/ADC0/PCINT8
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 24: PC1/ADC1/PCINT9
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

        #
        # pin 25: PC2/ADC2/PCINT10
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 26: PC2/ADC3/PCINT11
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 27: PC4/ADC4/SDA/PCINT12
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 28: PC5/ADC5/SCL/PCINT13
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 29: PC6/-RESET/PCINT14
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 30: PD0/RXD/PCINT16
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 31: PD1/TXD/PCINT17
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 32: PD2/INT0/PCINT18
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

#
```

```
# graphics
#

class CBA(part):
   def __init__(self,r=.02):
      self.value = ''
      self.pad = [point(0,0,0)]
      self.labels = []
      d = 3*r
      self.shape = circle(0,0,r)
      self.shape = add(self.shape,translate(circle(0,0,r),-d,d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),-d,0,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),-d,-d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),0,-d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),d,-d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),d,0,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),d,d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),0,d,0))

class hello(part):
   #
   # HELLO
   #
   def __init__(self,w,z):
      self.value = ''
      self.pad = [point(0,0,0)]
      self.labels = []
      # H
      self.shape = rectangle(-.01,.01,0,.1)
      self.shape = add(self.shape,rectangle(0,.05,.04,.06))
      self.shape = add(self.shape,rectangle(.04,.06,0,.1))
      # E
      self.shape = add(self.shape,rectangle(.08,.1,0,.1))
      self.shape = add(self.shape,rectangle(.1,.14,0,.02))
      self.shape = add(self.shape,rectangle(.1,.13,.04,.06))
      self.shape = add(self.shape,rectangle(.1,.14,.08,.1))
      # L
      self.shape = add(self.shape,rectangle(.16,.18,0,.1))
      self.shape = add(self.shape,rectangle(.18,.21,0,.02))
      # L
      self.shape = add(self.shape,rectangle(.23,.25,0,.1))
      self.shape = add(self.shape,rectangle(.25,.28,0,.02))
      # 0
      self.shape = add(self.shape,rectangle(.3,.32,0,.1))
      self.shape = add(self.shape,rectangle(.32,.345,0,.02))
      self.shape = add(self.shape,rectangle(.32,.345,.08,.1))
      self.shape = add(self.shape,rectangle(.345,.365,0,.1))

#
# define board
#

x0 = 1
y0 = 1
width = 1.4
height = .74
z = -.005
w = .018
mask = .004

pcb = PCB(x0,y0,width,height,mask)

IC1 = ATtiny45_SOIC('IC1\nt45')
pcb = IC1.add(pcb,x0+0.96,y0+.39,z,angle=-90)

R1 = R_1206('R1\n1M')
pcb = R1.add(pcb,IC1.pad[1].x-.1,IC1.y,z,angle=90)

pcb = wire(pcb,w,
   IC1.pad[8],
   R1.pad[1])

pcb = wire(pcb,w,
   point(R1.x,R1.pad[2].y+.025,z),
   IC1.pad[3])

pcb = wire(pcb,w,
   IC1.pad[2],
   point(IC1.pad[2].x,IC1.pad[2].y+.07,z),
   point(R1.x+.06,R1.pad[2].y-.06,z),
   point(R1.x-.06,IC1.y,z))

R2 = R_1206('R2\n10k')
pcb = R2.add(pcb,R1.x-.12,R1.y,z,angle=90)

pcb = wire(pcb,w,
   IC1.pad[8],
   R2.pad[1])

pcb = wire(pcb,w,
   point(IC1.pad[1].x,IC1.pad[1].y-.018,z),
   R2.pad[2])

R3 = R_1206('R3\n10k')
pcb = R3.add(pcb,R2.x-.09,R2.y,z,angle=90)

pcb = wire(pcb,w,
   IC1.pad[8],
   R3.pad[1])

IC2 = regulator_SOT23('IC2\n5V')
```

```
pcb = IC2.add(pcb,R3.x-.15,R3.y,z,angle=90)

pcb = wire(pcb,w,
   IC2.pad[3],
   IC1.pad[4])

C1 = C_1206('C1\n3.3uF')
pcb = C1.add(pcb,IC2.x,IC1.pad[8].y,z)

pcb = wire(pcb,w,
   C1.pad[1],
   IC2.pad[1])

pcb = wire(pcb,w,
   IC2.pad[3],
   C1.pad[2])

pcb = wire(pcb,w,
   C1.pad[1],
   point(C1.pad[1].x,IC1.pad[8].y+.1,z),
   IC1.pad[8])

J1 = MTA_serial('J1')
pcb = J1.add(pcb,IC2.x-.26,IC1.y-.01,z,angle=-90)

pcb = wire(pcb,w,
   J1.pad[1],
   IC2.pad[3])

pcb = wire(pcb,w,
   J1.pad[4],
   IC2.pad[2])

pcb = wire(pcb,w,
   J1.pad[3],
   point(J1.pad[2].x,IC1.pad[7].y+.15,z),
   IC1.pad[7])

J2 = MTA_ICP('J2')
pcb = J2.add(pcb,IC1.x+.28,IC1.y,z,angle=90)

pcb = wire(pcb,w,
   IC1.pad[6],
   point(IC1.pad[6].x,IC1.pad[6].y+.1,z),
   point(J2.pad[1].x-.03,J2.pad[1].y,z),
   J2.pad[1])

pcb = wire(pcb,w,
   J2.pad[2],
   IC1.pad[4])

pcb = wire(pcb,w,
   J2.pad[3],
   point(J2.x,J2.pad[5].y,z),
   IC1.pad[5])

pcb = wire(pcb,w,
   IC1.pad[7],
   point(IC1.pad[7].x,IC1.pad[7].y+.15,z),
   point(J2.pad[1].x+.02,J2.pad[1].y+.05,z),
   J2.pad[5])

R4 = R_1206('R4\n10k')
pcb = R4.add(pcb,IC1.pad[4].x+.05,y0+.09,z)

pcb = wire(pcb,w,
   R4.pad[1],
   IC1.pad[3])

pcb = wire(pcb,w,
   R4.pad[2],
   point(R4.pad[2].x,IC1.pad[4].y-.03,z),
   IC1.pad[4])

C2 = C_1206('C2\n0.1uF')
pcb = C2.add(pcb,IC1.pad[1].x,IC1.y-.25,z)

pcb = wire(pcb,w,
   C2.pad[2],
   IC1.pad[3])

pcb = wire(pcb,w,
   J2.pad[4],
   point(J2.pad[4].x,y0+.025,z),
   point(C2.x,IC1.pad[1].y-.02,z),
   IC1.pad[1])

# pads for 423-1043 electret

#padx = R3.x+.0
padx = C2.pad[1].x-.07
pady = y0

pcb = wire(pcb,.04,
   point(padx,pady,z),
   point(padx,pady+.075,z))

pcb = wire(pcb,.04,
   point(padx-.1,pady,z),
   point(padx-.1,pady+.075,z))
```

```
pcb = wire(pcb,w,
   C2.pad[1],
   point(padx,pady,z))

pcb = wire(pcb,w,
   point(padx-.1,pady,z),
   point(padx-.1,C2.y+.025,z),
   IC2.pad[3])

pcb = wire(pcb,w,
   point(padx,pady,z),
   point(padx,C2.y+.06,z),
   R3.pad[2])

H1 = hello(w,z)
pcb = H1.add(pcb,J1.x+.075,y0+.02,z)

#
# assign board and exterior to cad.function for milling
#
# use single-sided FR2
# use 1/64 end-mill
# set tool diameter = .0156
# set xy, z speed = 4
# set # contours = -1
#

cad.function = add(pcb.board,pcb.exterior)

#
# uncomment to export traces
#

#cad.function = pcb.board

#
# uncomment to export exterior
#

#cad.function = pcb.exterior

#
# uncomment to export solder mask
#

#cad.function = pcb.mask

#
# uncomment to mill out board
#
# use 1/32 end-mill
# set tool diameter = .0312
# set xy, z speed = .5
# set # contours = 1
#

#cad.function = pcb.interior
#z = -.065

#
# define limits and parameters
#

d = 0.05
cad.xmin = x0-d # min x to render
cad.xmax = x0+width+d # max x to render
cad.ymin = y0-d # min y to render
cad.ymax = y0+height+d # max y to render
cad.zmin = z
cad.zmax = .050
dpi = 200 # low resolution for previewing
#dpi = 500 # high resolution for machining
nxy = int(dpi*(cad.xmax-cad.xmin))
cad.nx = int(dpi*(cad.xmax-cad.xmin)) # x points to render
cad.ny = int(dpi*(cad.ymax-cad.ymin)) # y points to render
cad.nz = 1
cad.inches_per_unit = 1.0 # use inch units
cad.view('xy') # 2D view
```

```
#
# hello.mic.44.cad
#
# microphone hello-world
#
# Neil Gershenfeld
# CBA MIT 10/24/07
#
# (c) Massachusetts Institute of Technology 2007
# Permission granted for experimental and personal use;
# license for commercial sale available from MIT.
#
#

#
# define shapes and transformation
#
# circle(x0, y0, r)
# cylinder(x0, y0, z0, z1, r)
# cone(x0, y0, z0, z1, r0)
# sphere(x0, y0, z0, r)
# torus(x0, y0, z0, r0, r1)
# rectangle(x0, x1, y0, y1)
# cube(x0, x1, y0, y1, z0, z1)
# triangle(x0, y0, x1, y1, x2, y2) (points in clockwise order)
# pyramid(x0, x1, y0, y1, z0, z1)
# function(Z_of_XY)
# functions(upper_Z_of_XY,lower_Z_of_XY)
# add(part1, part2)
# subtract(part1, part2)
# intersect(part1, part2)
# move(part,dx,dy)
# translate(part,dx,dy,dz)
# rotate(part, angle)
# rotate_x(part, angle)
# rotate_y(part, angle)
# rotate_z(part, angle)
# rotate_90(part)
# rotate_180(part)
# rotate_270(part)
# reflect_x(part)
# reflect_y(part)
# reflect_z(part)
# reflect_xy(part)
# reflect_xz(part)
# reflect_yz(part)
# scale_x(part, x0, sx)
# scale_y(part, y0, sy)
# scale_z(part, z0, sz)
# scale_xy(part, x0, y0, sxy)
# scale_xyz(part, x0, y0, z0, sxyz)
# coscale_x_y(part, x0, y0, y1, angle0, angle1, amplitude, offset)
# coscale_x_z(part, x0, z0, z1, angle0, angle1, amplitude, offset)
# coscale_xy_z(part, x0, y0, z0, z1, angle0, angle1, amplitude, offset)
# taper_x_y(part, x0, y0, y1, s0, s1)
# taper_x_z(part, x0, z0, z1, s0, s1)
# taper_xy_z(part, x0, y0, z0, z1, s0, s1)
# shear_x_y(part, y0, y1, dx0, dx1)
# shear_x_z(part, z0, z1, dx0, dx1)
# (more to come)

def circle(x0, y0, r):
   part = "(((X-x0)**2 + (Y-y0)**2) <= r**2)"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'r',str(r))
   return part

def cylinder(x0, y0, z0, z1, r):
   part = "(((X-x0)**2 + (Y-y0)**2 <= r**2) & (Z >= z0) & (Z <= z1))"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'r',str(r))
   return part

def cone(x0, y0, z0, z1, r0):
   part = cylinder(x0, y0, z0, z1, r0)
   part = taper_xy_z(part, x0, y0, z0, z1, 1.0, 0.0)
   return part

def sphere(x0, y0, z0, r):
   part = "(((X-x0)**2 + (Y-y0)**2 + (Z-z0)**2) <= r**2)"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'r',str(r))
   return part

def torus(x0, y0, z0, r0, r1):
   part = "(((r0 - sqrt((X-x0)**2 + (Y-y0)**2))**2 + (Z-z0)**2) <= r1**2)"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'r0',str(r0))
   part = replace(part,'r1',str(r1))
   return part

def rectangle(x0, x1, y0, y1):
   part = "((X >= x0) & (X <= x1) & (Y >= y0) & (Y <= y1))"
```

```
   part = replace(part,'x0',str(x0))
   part = replace(part,'x1',str(x1))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   return part

def cube(x0, x1, y0, y1, z0, z1):
   part = "((X >= x0) & (X <= x1) & (Y >= y0) & (Y <= y1) & (Z >= z0) & (Z <= z1))"
   part = replace(part,'x0',str(x0))
   part = replace(part,'x1',str(x1))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   return part

def triangle(x0, y0, x1, y1, x2, y2): # points in clockwise order
   part = "((((y1-y0)*(X-x0)-(x1-x0)*(Y-y0)) >= 0) & (((y2-y1)*(X-x1)-(x2-x1)*(Y-y1)) >= 0) & (((y0-y2)*(X-x2)-(x0-x2)*(Y-y2)) >= 0))"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'x1',str(x1))
   part = replace(part,'y1',str(y1))
   part = replace(part,'x2',str(x2))
   part = replace(part,'y2',str(y2))
   return part

def pyramid(x0, x1, y0, y1, z0, z1):
   part = cube(x0, x1, y0, y1, z0, z1)
   part = taper_xy_z(part, (x0+x1)/2., (y0+y1)/2., z0, z1, 1.0, 0.0)
   return part

def function(Z_of_XY):
   part = '(Z <= '+Z_of_XY+')'
   return part

def functions(upper_Z_of_XY,lower_Z_of_XY):
   part = '(Z <= '+upper_Z_of_XY+') & (Z >= '+lower_Z_of_XY+')'
   return part

def add(part1, part2):
   part = "part1 | part2"
   part = replace(part,'part1',part1)
   part = replace(part,'part2',part2)
   return part

def subtract(part1, part2):
   part = "(part1) & ~(part2)"
   part = replace(part,'part1',part1)
   part = replace(part,'part2',part2)
   return part

def intersect(part1, part2):
   part = "(part1) & (part2)"
   part = replace(part,'part1',part1)
   part = replace(part,'part2',part2)
   return part

def move(part,dx,dy):
   part = replace(part,'X','(X-'+str(dx)+')')
   part = replace(part,'Y','(Y-'+str(dy)+')')
   return part

def translate(part,dx,dy,dz):
   part = replace(part,'X','(X-'+str(dx)+')')
   part = replace(part,'Y','(Y-'+str(dy)+')')
   part = replace(part,'Z','(Z-'+str(dz)+')')
   return part

def rotate(part, angle):
   angle = angle*pi/180
   part = replace(part,'X','(cos(angle)*X+sin(angle)*y)')
   part = replace(part,'Y','(-sin(angle)*X+cos(angle)*y)')
   part = replace(part,'y','Y')
   part = replace(part,'angle',str(angle))
   return part

def rotate_x(part, angle):
   angle = angle*pi/180
   part = replace(part,'Y','(cos(angle)*Y+sin(angle)*z)')
   part = replace(part,'Z','(-sin(angle)*Y+cos(angle)*z)')
   part = replace(part,'z','Z')
   part = replace(part,'angle',str(angle))
   return part

def rotate_y(part, angle):
   angle = angle*pi/180
   part = replace(part,'X','(cos(angle)*X+sin(angle)*z)')
   part = replace(part,'Z','(-sin(angle)*X+cos(angle)*z)')
   part = replace(part,'z','Z')
   part = replace(part,'angle',str(angle))
   return part

def rotate_z(part, angle):
   angle = angle*pi/180
   part = replace(part,'X','(cos(angle)*X+sin(angle)*y)')
   part = replace(part,'Y','(-sin(angle)*X+cos(angle)*y)')
   part = replace(part,'y','Y')
   part = replace(part,'angle',str(angle))
   return part

def rotate_90(part):
```

```
   part = reflect_xy(part)
   part = reflect_y(part)
   return part

def rotate_180(part):
   part = reflect_xy(part)
   part = reflect_y(part)
   part = reflect_xy(part)
   part = reflect_y(part)
   return part

def rotate_270(part):
   part = reflect_xy(part)
   part = reflect_y(part)
   part = reflect_xy(part)
   part = reflect_y(part)
   part = reflect_xy(part)
   part = reflect_y(part)
   return part

def reflect_x(part):
   part = replace(part,'X','-X')
   return part

def reflect_y(part):
   part = replace(part,'Y','-Y')
   return part

def reflect_z(part):
   part = replace(part,'Z','-Z')
   return part

def reflect_xy(part):
   part = replace(part,'X','temp')
   part = replace(part,'Y','X')
   part = replace(part,'temp','Y')
   return part

def reflect_xz(part):
   part = replace(part,'X','temp')
   part = replace(part,'Z','X')
   part = replace(part,'temp','Z')
   return part

def reflect_yz(part):
   part = replace(part,'Y','temp')
   part = replace(part,'Z','Y')
   part = replace(part,'temp','Z')
   return part

def scale_x(part, x0, sx):
   part = replace(part,'X','(x0 + (X-x0)/sx)')
   part = replace(part,'x0',str(x0))
   part = replace(part,'sx',str(sx))
   return part

def scale_y(part, y0, sy):
   part = replace(part,'Y','(y0 + (Y-y0)/sy)')
   part = replace(part,'y0',str(y0))
   part = replace(part,'sy',str(sy))
   return part

def scale_z(part, z0, sz):
   part = replace(part,'Z','(z0 + (Z-z0)/sz)')
   part = replace(part,'z0',str(z0))
   part = replace(part,'sz',str(sz))
   return part

def scale_xy(part, x0, y0, sxy):
   part = replace(part,'X','(x0 + (X-x0)/sxy)')
   part = replace(part,'Y','(y0 + (Y-y0)/sxy)')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'sxy',str(sxy))
   return part

def scale_xyz(part, x0, y0, z0, sxyz):
   part = replace(part,'X','(x0 + (X-x0)/sxyz)')
   part = replace(part,'Y','(y0 + (Y-y0)/sxyz)')
   part = replace(part,'Z','(z0 + (Z-z0)/sxyz)')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'sxyz',str(sxyz))
   return part

def coscale_x_y(part, x0, y0, y1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
   part = replace(part,'X','(x0 + (X-x0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Y-y0)/(y1-y0))))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

def coscale_x_z(part, x0, z0, z1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
```

```
   phase1 = pi*angle1/180.
   part = replace(part,'X','(x0 + (X-x0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0))))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

def coscale_xy_z(part, x0, y0, z0, z1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
   part = replace(part,'X','(x0 + (X-x0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0))))')
   part = replace(part,'Y','(y0 + (Y-y0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0))))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

def taper_x_y(part, x0, y0, y1, s0, s1):
   part = replace(part,'X','(x0 + (X-x0)*(y1-y0)/(s1*(Y-y0) + s0*(y1-Y)))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'s0',str(s0))
   part = replace(part,'s1',str(s1))
   return part

def taper_x_z(part, x0, z0, z1, s0, s1):
   part = replace(part,'X','(x0 + (X-x0)*(z1-z0)/(s1*(Z-z0) + s0*(z1-Z)))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'s0',str(s0))
   part = replace(part,'s1',str(s1))
   return part

def taper_xy_z(part, x0, y0, z0, z1, s0, s1):
   part = replace(part,'X','(x0 + (X-x0)*(z1-z0)/(s1*(Z-z0) + s0*(z1-Z)))')
   part = replace(part,'Y','(y0 + (Y-y0)*(z1-z0)/(s1*(Z-z0) + s0*(z1-Z)))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'s0',str(s0))
   part = replace(part,'s1',str(s1))
   return part

def shear_x_y(part, y0, y1, dx0, dx1):
   part = replace(part,'X','(X - dx0 - (dx1-dx0)*(Y-y0)/(y1-y0))')
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'dx0',str(dx0))
   part = replace(part,'dx1',str(dx1))
   return part

def shear_x_z(part, z0, z1, dx0, dx1):
   part = replace(part,'X','(X - dx0 - (dx1-dx0)*(Z-z0)/(z1-z0))')
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'dx0',str(dx0))
   part = replace(part,'dx1',str(dx1))
   return part

def coshear_x_z(part, z0, z1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
   part = replace(part,'X','(X - offset - amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0)))')
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

#
# PCB classes and definitions
#

cad.labels = []

class part:
   def add(self,pcb,x,y,z=0,angle=0):
      self.x = x
      self.y = y
      self.z = z
      self.angle = angle
      if (angle == 90):
         self.shape = rotate_90(self.shape)
      elif (angle == 180):
         self.shape = rotate_180(self.shape)
      elif ((angle == 270) | (angle == -90)):
         self.shape = rotate_270(self.shape)
```

```
      angle = pi*angle/180
      self.shape = translate(self.shape,x,y,z)
      for i in range(len(self.pad)):
         xnew = cos(angle)*self.pad[i].x + sin(angle)*self.pad[i].y
         ynew = -sin(angle)*self.pad[i].x + cos(angle)*self.pad[i].y
           self.pad[i].x = x + xnew
           self.pad[i].y = y + ynew
         self.pad[i].z += z
      cad.labels.append(cad_text(x,y,z,self.value,size=14))
      for i in range(len(self.labels)):
         xnew = cos(angle)*self.labels[i].x + sin(angle)*self.labels[i].y
         ynew = -sin(angle)*self.labels[i].x + cos(angle)*self.labels[i].y
           self.labels[i].x = x + xnew
           self.labels[i].y = y + ynew
         self.labels[i].z += z
           cad.labels.append(self.labels[i])
      return add(pcb,self.shape)

def wire(pcb,width,*points):
   for i in range(1,len(points)):
      x0 = points[i-1].x
      y0 = points[i-1].y
      z0 = points[i-1].z
      x1 = points[i].x
      y1 = points[i].y
      z1 = points[i].z
      if (x0 < x1):
         pcb = add(pcb,cube(x0-width/2,x1+width/2,y0-width/2,y0+width/2,z0,z0))
      elif (x1 < x0):
         pcb = add(pcb,cube(x1-width/2,x0+width/2,y0-width/2,y0+width/2,z0,z0))
      if (y0 < y1):
         pcb = add(pcb,cube(x1-width/2,x1+width/2,y0-width/2,y1+width/2,z0,z0))
      elif (y1 < y0):
         pcb = add(pcb,cube(x1-width/2,x1+width/2,y1-width/2,y0+width/2,z0,z0))
   return pcb

#
# PCB library
#

#
# discretes
#

pad_0402 = cube(-.0175,.0175,-.014,.014,0,0)

class R_0402(part):
   #
   # 0402 resistor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_0402,-.0265,0,0)
      self.pad.append(point(-.0265,0,0))
      self.shape = add(self.shape,translate(pad_0402,.0265,0,0))
      self.pad.append(point(.0265,0,0))

pad_1206 = cube(-.032,.032,-.034,.034,0,0)

class R_1206(part):
   #
   # 1206 resistor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))

class C_1206(part):
   #
   # 1206 capacitor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))

pad_1210 = cube(-.032,.032,-.048,.048,0,0)

class L_1210(part):
   #
   # 1210 inductor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_1210,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.shape = add(self.shape,translate(pad_1210,.06,0,0))
      self.pad.append(point(.055,0,0))
```

```
pad_choke = cube(-.06,.06,-.06,.06,0,0)

class choke(part):
   #
   # Panasonic ELLCTV
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_choke,-.177,-.177,0)
      self.pad.append(point(-.177,-.177,0))
      self.shape = add(self.shape,translate(pad_choke,.177,.177,0))
      self.pad.append(point(.177,.177,0))

#
# connectors
#

pad_MTA = cube(-.021,.021,-.041,.041,0,0)
pad_MTA_solder = cube(-.071,.071,-.041,.041,0,0)

class MTA_2(part):
   #
   # AMP 1445121-2
   # MTA .050 SMT 2-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1
      #
      self.shape = translate(pad_MTA,-.025,-.1,0)
      self.pad.append(point(-.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,.1,0))
      self.pad.append(point(.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'2',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.187,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.187,0,0))

class MTA_power(part):
   #
   # AMP 1445121-2
   # MTA .050 SMT 2-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: Gnd
      #
      self.shape = translate(pad_MTA,-.025,-.1,0)
      self.pad.append(point(-.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: Vcc
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,.1,0))
      self.pad.append(point(.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Vcc',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.187,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.187,0,0))

class MTA_i0(part):
   #
   # AMP 1445121-3
   # MTA .050 SMT 3-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: GND
      #
      self.shape = translate(pad_MTA,.05,.1,0)
      self.pad.append(point(.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1\nGND',14))
      #
      # pin 2: power
      #
      self.shape = add(self.shape,translate(pad_MTA,-.05,.1,0))
      self.pad.append(point(-.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Vcc',14))
      #
      # pin 3: data
      #
```

```
      self.shape = add(self.shape,translate(pad_MTA,0,-.1,0))
      self.pad.append(point(0,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'data',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.212,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.212,0,0))

class MTA_serial(part):
   #
   # AMP 1445121-4
   # MTA .050 SMT 4-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: Gnd
      #
      self.shape = translate(pad_MTA,-.075,-.1,0)
      self.pad.append(point(-.075,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: Tx
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,-.1,0))
      self.pad.append(point(.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Tx',14))
      #
      # pin 3: Rx
      #
      self.shape = add(self.shape,translate(pad_MTA,.075,.1,0))
      self.pad.append(point(.075,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Rx',14))
      #
      # pin 4: DTR
      #
      self.shape = add(self.shape,translate(pad_MTA,-.025,.1,0))
      self.pad.append(point(-.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DTR',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.237,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.237,0,0))

class MTA_ICP(part):
   #
   # AMP 1445121-5
   # MTA .050 SMT 4-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: MISO
      #
      self.shape = translate(pad_MTA,-.1,-.1,0)
      self.pad.append(point(-.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'MISO',14))
      #
      # pin 2: GND
      #
      self.shape = add(self.shape,translate(pad_MTA,0,-.1,0))
      self.pad.append(point(0,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))
      #
      # pin 3: MOSI
      #
      self.shape = add(self.shape,translate(pad_MTA,.1,-.1,0))
      self.pad.append(point(.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'MOSI',14))
      #
      # pin 4: -RESET
      #
      self.shape = add(self.shape,translate(pad_MTA,.05,.1,0))
      self.pad.append(point(.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'-RESET',14))
      #
      # pin 5: SCK
      #
      self.shape = add(self.shape,translate(pad_MTA,-.05,.1,0))
      self.pad.append(point(-.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'SCK',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.262,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.262,0,0))

class power_65mm(part):
   #
   # CUI PJ1-023-SMT
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
```

```
      #
      # pin 1: power
      #
      self.shape = cube(.433,.512,-.047,.047,0,0)
      self.pad.append(point(.467,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'P',14))
      #
      # pin 2: ground
      #
      self.shape = add(self.shape,cube(.285,.423,-.189,-.098,0,0))
      self.pad.append(point(.354,-.144,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 3: contact
      #
      self.shape = add(self.shape,cube(.325,.463,.098,.189,0,0))
      self.pad.append(point(.394,.144,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,cube(.108,.246,-.169,-.110,0,0))
      self.shape = add(self.shape,cube(.069,.207,.110,.169,0,0))

pad_stereo_2_5mm = cube(-.03,.03,-.05,.05,0,0)

class stereo_2_5mm(part):
   #
   # CUI SJ1-2533-SMT
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: base
      #
      self.shape = translate(pad_stereo_2_5mm,-.130,-.16,0)
      self.pad.append(point(-.130,-.149,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'base',14))
      #
      # pin 2: tip
      #
      self.shape = add(self.shape,translate(pad_stereo_2_5mm,.197,.15,0))
      self.pad.append(point(.197,.141,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'tip',14))
      #
      # pin 3: middle
      #
      self.shape = add(self.shape,translate(pad_stereo_2_5mm,-.012,-.16,0))
      self.pad.append(point(-.012,-.149,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'middle',14))

pad_Molex = cube(-.0155,.0155,-.0265,.0265,0,0)
pad_Molex_solder = cube(-.055,.055,-.065,.065,0,0)

class Molex_serial(part):
   #
   # Molex 53261-0471
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: Rx
      #
      self.shape = translate(pad_Molex,-.075,.064,0)
      self.pad.append(point(-.075,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Rx',14))
      #
      # pin 2: Tx
      #
      self.shape = add(self.shape,translate(pad_Molex,-.025,.064,0))
      self.pad.append(point(-.025,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Tx',14))
      #
      # pin 3: DTR
      #
      self.shape = add(self.shape,translate(pad_Molex,.025,.064,0))
      self.pad.append(point(.025,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DTR',14))
      #
      # pin 4: GND
      #
      self.shape = add(self.shape,translate(pad_Molex,.075,.064,0))
      self.pad.append(point(.075,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_Molex_solder,-.16,-.065,0))
      self.shape = add(self.shape,translate(pad_Molex_solder,.16,-.065,0))

#
# switches
#

pad_button_6mm = cube(-.04,.04,-.03,.03,0,0)

class button_6mm(part):
```

```
   #
   # Omron 6mm pushbutton
   # B3SN-3112P
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # left 1
      #
      self.shape = translate(pad_button_6mm,-.125,.08,0)
      self.pad.append(point(-.125,.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'L1',14))
      #
      # right 1
      #
      self.shape = add(self.shape,translate(pad_button_6mm,-.125,-.08,0))
      self.pad.append(point(-.125,-.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'R1',14))
      #
      # right 2
      #
      self.shape = add(self.shape,translate(pad_button_6mm,.125,-.08,0))
      self.pad.append(point(.125,-.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'R2',14))
      #
      # left 2
      #
      self.shape = add(self.shape,translate(pad_button_6mm,.125,.08,0))
      self.pad.append(point(.125,.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'L2',14))

#
# crystals and resonators
#

pad_XTAL = cube(-.016,.016,-.077,.077,0,0)

class XTAL(part):
   #
   # Panasonic EFOBM series
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # left
      #
      self.shape = translate(pad_XTAL,-.053,0,0)
      self.pad.append(point(-.053,0,0))
      #
      # ground
      #
      self.shape = add(self.shape,translate(pad_XTAL,0,0,0))
      self.pad.append(point(0,0,0))
      #
      # right
      #
      self.shape = add(self.shape,translate(pad_XTAL,.053,0,0))
      self.pad.append(point(.053,0,0))

#
# diodes, transistors, regulators
#

class D_1206(part):
   #
   # 1206 diode
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # anode
      #
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A',14))
      #
      # cathode
      #
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))

class LED_1206(part):
   #
   # 1206 LED
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # anode
      #
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
```

```
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A',14))
      #
      # cathode
      #
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))

pad_SOD_123 = cube(-.02,.02,-.024,.024,0,0)

class D_SOD_123(part):
   #
   # SOD-123 diode
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # anode
      #
      self.shape = translate(pad_SOD_123,-.07,0,0)
      self.pad.append(point(-.07,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A',14))
      #
      # cathode
      #
      self.shape = add(self.shape,translate(pad_SOD_123,.07,0,0))
      self.pad.append(point(.07,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))

pad_SOT23 = cube(-.015,.015,-.02,.02,0,0)

class NMOSFET_SOT23(part):
   #
   # Fairchild NDS355AN
   #
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: gate
      #
      self.shape = translate(pad_SOT23,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 2: source
      #
      self.shape = add(self.shape,translate(pad_SOT23,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'S',14))
      #
      # pin 3: drain
      #
      self.shape = add(self.shape,translate(pad_SOT23,0,.045,0))
      self.pad.append(point(0,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'D',14))

class PMOSFET_SOT23(part):
   #
   # Fairchild NDS356AP
   #
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: gate
      #
      self.shape = translate(pad_SOT23,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 2: source
      #
      self.shape = add(self.shape,translate(pad_SOT23,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'S',14))
      #
      # pin 3: drain
      #
      self.shape = add(self.shape,translate(pad_SOT23,0,.045,0))
      self.pad.append(point(0,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'D',14))

class regulator_SOT23(part):
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
```

```
      #
      # pin 1: output
      #
      self.shape = translate(pad_SOT23,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: input
      #
      self.shape = add(self.shape,translate(pad_SOT23,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'in',14))
      #
      # pin 3: ground
      #
      self.shape = add(self.shape,translate(pad_SOT23,0,.045,0))
      self.pad.append(point(0,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'gnd',14))

#
# ICs
#

pad_SOT23_5 = cube(-.01,.01,-.02,.02,0,0)

class op_amp_SOT23_5(part):
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: output
      #
      self.shape = translate(pad_SOT23_5,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'O',14))
      #
      # pin 2: V-
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,0,-.045,0))
      self.pad.append(point(0,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V-',14))
      #
      # pin 3: I+
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'I+',14))
      #
      # pin 4: I-
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,.0375,.045,0))
      self.pad.append(point(.0375,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'I-',14))
      #
      # pin 5: V+
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,-.0375,.045,0))
      self.pad.append(point(-.0375,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V+',14))

pad_SOICN = cube(-.035,.035,-.015,.015,0,0)

class op_amp_SOICN(part):
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: A out
      #
      self.shape = translate(pad_SOICN,-.12,.075,0)
      self.pad.append(point(-.12,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1 Ao',14))
      #
      # pin 2: A-
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.025,0))
      self.pad.append(point(-.12,.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A-',14))
      #
      # pin 3: A+
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.025,0))
      self.pad.append(point(-.12,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A+',14))
      #
      # pin 4: V-
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.075,0))
      self.pad.append(point(-.12,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V-',14))
      #
      # pin 5: B+
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.075,0))
      self.pad.append(point(.12,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'B+',14))
```

```
      #
      # pin 6: B-
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.025,0))
      self.pad.append(point(.12,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'B-',14))
      #
      # pin 7: B out
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.025,0))
      self.pad.append(point(.12,.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Bo',14))
      #
      # pin 8: V+
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.075,0))
      self.pad.append(point(.12,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V+',14))

pad_SOIC = cube(-.043,.043,-.015,.015,0,0)

class ATtiny45_SOIC(part):
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: PB5/dW/ADC0/-RESET/PCINT5
      #
      self.shape = translate(pad_SOIC,-.14,.075,0)
      self.pad.append(point(-.14,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: PB3/ADC3/-OC1B/CLKI/XTAL1/PCINT3
      #
      self.shape = add(self.shape,translate(pad_SOIC,-.14,.025,0))
      self.pad.append(point(-.14,.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB3',14))
      #
      # pin 3: PB4/ADC2/OC1B/CLKO/XTAL2/PCINT4
      #
      self.shape = add(self.shape,translate(pad_SOIC,-.14,-.025,0))
      self.pad.append(point(-.14,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB4',14))
      #
      # pin 4: GND
      #
      self.shape = add(self.shape,translate(pad_SOIC,-.14,-.075,0))
      self.pad.append(point(-.14,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))
      #
      # pin 5: PB0/MOSI/DI/SDA/AIN0/OC0A/-OC1A/AREF/PCINT0
      #
      self.shape = add(self.shape,translate(pad_SOIC,.14,-.075,0))
      self.pad.append(point(.14,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 6: PB1/MISO/DO/AIN1/OC0B/OC1A/PCINT1
      #
      self.shape = add(self.shape,translate(pad_SOIC,.14,-.025,0))
      self.pad.append(point(.14,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB1',14))
      #
      # pin 7: PB2/SCK/USCK/SCL/ADC1/T0/INT0/PCINT2
      #
      self.shape = add(self.shape,translate(pad_SOIC,.14,.025,0))
      self.pad.append(point(.14,.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB2',14))
      #
      # pin 8: VCC
      #
      self.shape = add(self.shape,translate(pad_SOIC,.14,.075,0))
      self.pad.append(point(.14,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'VCC',14))

class ATtiny44_SOICN(part):
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: VCC
      #
      self.shape = translate(pad_SOICN,-.12,.15,0)
      self.pad.append(point(-.12,.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: PB0/XTAL1/PCINT8
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 3: PB1/XTAL2/PCINT9
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.050,0))
      self.pad.append(point(-.12,.05,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB1',14))
      #
      # pin 4: PB3/dW/-RESET/PCINT11
      #
```

```
      self.shape = add(self.shape,translate(pad_SOICN,-.12,0,0))
      self.pad.append(point(-.12,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB3',14))
      #
      # pin 5: PB2/CKOUT/OC0A/INT0/PCINT10
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.05,0))
      self.pad.append(point(-.12,-.05,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB2',14))
      #
      # pin 6: PA7/ADC7/OC0B/ICP/PCINT7
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.1,0))
      self.pad.append(point(-.12,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA7',14))
      #
      # pin 7: PA6/ADC6/MOSI/SDA/OC1A/PCINT6
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.15,0))
      self.pad.append(point(-.12,-.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA6',14))
      #
      # pin 8: PA5/ADC5/DO/MISO/OC1B/PCINT5
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.15,0))
      self.pad.append(point(.12,-.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA5',14))
      #
      # pin 9: PA4/ADC4/USCK/SCL/T1/PCINT4
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.1,0))
      self.pad.append(point(.12,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA4',14))
      #
      # pin 10: PA3/ADC3/T0/PCINT3
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.05,0))
      self.pad.append(point(.12,-.05,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA3',14))
      #
      # pin 11: PA2/ADC2/AIN1/PCINT2
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,0,0))
      self.pad.append(point(.12,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA2',14))
      #
      # pin 12: PA1/ADC1/AIN0/PCINT1
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.050,0))
      self.pad.append(point(.12,.05,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA1',14))
      #
      # pin 13: PA0/ADC0/AREF/PCINT0
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.1,0))
      self.pad.append(point(.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA0',14))
      #
      # pin 14: GND
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.15,0))
      self.pad.append(point(.12,.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))

pad_TQFP = cube(-.043,.043,-.015,.015,0,0)

class ATmega88_TQFP(part):
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []

      #
      # pin 1: PD3/PCINT19/OC2B/INT1
      #
      self.shape = translate(pad_SOICN,-.12,.15,0)
      self.pad.append(point(-.12,.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: PD4/PCINT20/XCK/T0
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 3: GND
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 4: VCC
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 5: GND
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
```

```
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 6: VCC
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 7: PB6/PCINT6/XTAL1/TOSC1
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 8: PB7/PCINT7/XTAL2/TOSC2
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

#
# pin 9: PD5/PCINT21/OC0B/T1
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 10: PD6/PCINT22/OC0A/AIN0
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 11: PD7/PCINT23/AIN1
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 12: PB0/PCINT0/CLKO/ICP1
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 13: PB1/PCINT1/OC1A
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 14: PB2/PCINT2/-SS/OC1B
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 15: PB3/PCINT3/OC2A/MOSI
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 16: PB4/PCINT4/MISO
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

#
# pin 17: PB5/SCK/PCINT5
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 18: AVCC
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 19: ADC6
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 20: AREF
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 21: GND
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 22: ADC7
```

```
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 23: PC0/ADC0/PCINT8
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 24: PC1/ADC1/PCINT9
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

      #
      # pin 25: PC2/ADC2/PCINT10
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 26: PC2/ADC3/PCINT11
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 27: PC4/ADC4/SDA/PCINT12
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 28: PC5/ADC5/SCL/PCINT13
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 29: PC6/-RESET/PCINT14
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 30: PD0/RXD/PCINT16
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 31: PD1/TXD/PCINT17
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 32: PD2/INT0/PCINT18
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

#
# graphics
#

class CBA(part):
   def __init__(self,r=.02):
      self.value = ''
      self.pad = [point(0,0,0)]
      self.labels = []
      d = 3*r
      self.shape = circle(0,0,r)
      self.shape = add(self.shape,translate(circle(0,0,r),-d,d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),-d,0,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),-d,-d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),0,-d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),d,-d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),d,0,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),d,d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),0,d,0))

class hello(part):
   #
   # HELLO
   #
   def __init__(self,w,z):
      self.value = ''
      self.pad = [point(0,0,0)]
      self.labels = []
      # H
      self.shape = rectangle(-.01,.01,0,.1)
      self.shape = add(self.shape,rectangle(0,.05,.04,.06))
      self.shape = add(self.shape,rectangle(.04,.06,0,.1))
      # E
      self.shape = add(self.shape,rectangle(.08,.1,0,.1))
      self.shape = add(self.shape,rectangle(.1,.14,0,.02))
```

```
        self.shape = add(self.shape,rectangle(.1,.13,.04,.06))
        self.shape = add(self.shape,rectangle(.1,.14,.08,.1))
        # L
        self.shape = add(self.shape,rectangle(.16,.18,0,.1))
        self.shape = add(self.shape,rectangle(.18,.21,0,.02))
        # L
        self.shape = add(self.shape,rectangle(.23,.25,0,.1))
        self.shape = add(self.shape,rectangle(.25,.28,0,.02))
        # 0
        self.shape = add(self.shape,rectangle(.3,.32,0,.1))
        self.shape = add(self.shape,rectangle(.32,.345,0,.02))
        self.shape = add(self.shape,rectangle(.32,.345,.08,.1))
        self.shape = add(self.shape,rectangle(.345,.365,0,.1))

#
# define board
#

w = .016
width = 1.38
height = 1
x = 1
y = 1
z = -.005
d = .06

board = subtract(cube(-10,10,-10,10,-10,10),
    cube(x,x+width,y,y+height,-10,10))

frame = subtract(cube(-10,10,-10,10,-10,10),board)

IC1 = ATtiny44_SOICN('IC1\nt44')
pcb = IC1.add(board,x+.85,y+.51,z)

XTAL1 = XTAL('XTAL1\n20 MHz')
pcb = XTAL1.add(pcb,IC1.pad[2].x-.18,IC1.pad[2].y-.11,z,angle=90)

pcb = wire(pcb,w,
    IC1.pad[2],
    point(XTAL1.x+.065,XTAL1.pad[1].y,z),
    XTAL1.pad[1])

pcb = wire(pcb,w,
    IC1.pad[3],
    point(XTAL1.x+.11,XTAL1.pad[3].y,z),
    XTAL1.pad[3])

J1 = MTA_ICP('J1')
pcb = J1.add(pcb,IC1.x,IC1.pad[7].y-.2,z,angle=180)

pcb = wire(pcb,w,
    IC1.pad[7],
    J1.pad[3])

pcb = wire(pcb,w,
    IC1.pad[4],
    J1.pad[4])

pcb = wire(pcb,w,
    IC1.pad[14],
    point(J1.pad[5].x,IC1.pad[10].y,z),
    J1.pad[2])

pcb = wire(pcb,w,
    IC1.pad[9],
    J1.pad[5])

pcb = wire(pcb,w,
    IC1.pad[8],
    J1.pad[1])

J2 = MTA_power('J2\npower')
pcb = J2.add(pcb,IC1.x-.39,IC1.pad[1].y+.18,z,angle=0)

pcb = wire(pcb,w,
    XTAL1.pad[2],
    J2.pad[1])

IC2 = regulator_SOT23('IC2\n5V')
pcb = IC2.add(pcb,J2.x+.38,J2.y+.02,z,angle=180)

pcb = wire(pcb,w,
    IC1.pad[1],
    point(J1.pad[4].x,IC1.pad[1].y+.09,z),
    IC2.pad[1])

pcb = wire(pcb,w,
    J2.pad[2],
    IC2.pad[2])

pcb = wire(pcb,w,
    J2.pad[1],
    point(IC1.pad[1].x+.03,IC2.pad[3].y,z),
    IC2.pad[3])

R1 = R_1206('R1\n10k')
pcb = R1.add(pcb,IC2.x+.12,IC2.y-.03,z,angle=90)

pcb = wire(pcb,w,
    IC2.pad[1],
    R1.pad[1])
```

```
pcb = wire(pcb,w,
   IC1.pad[4],
   point(IC1.x,IC1.pad[14].y+.05,z),
   R1.pad[2])

C1 = C_1206('C1\n3.3uF')
pcb = C1.add(pcb,R1.x+.1,R1.y,z,angle=90)

pcb = wire(pcb,w,
   IC1.pad[14],
   C1.pad[2])

pcb = wire(pcb,w,
   C1.pad[2],
   point(C1.x+.06,J2.pad[2].y+.02,z),
   IC2.pad[3])

pcb = wire(pcb,w,
   C1.pad[1],
   R1.pad[1])

J3 = MTA_serial('J3\nserial')
pcb = J3.add(pcb,x+width-.18,IC1.y+.025,z,angle=90)

pcb = wire(pcb,w,
   IC1.pad[14],
   J3.pad[1])

pcb = wire(pcb,w,
   IC1.pad[10],
   J3.pad[3])

R2 = R_1206('R3\n0')
pcb = R2.add(pcb,J2.pad[1].x,J2.pad[1].y-.095,z)

pcb = wire(pcb,w,
   IC1.pad[1],
   R2.pad[2])

IC3 = op_amp_SOT23_5('IC3') # AD8605ARTZ-REEL7
pcb = IC3.add(pcb,IC1.x-.56,IC1.y-.29,z)

R4 = R_1206('R4\n100k')
pcb = R4.add(pcb,IC3.pad[5].x,IC3.pad[5].y+.09,z)

pcb = wire(pcb,w,
   IC3.pad[1],
   R4.pad[1])

pcb = wire(pcb,w,
   R4.pad[2],
   IC3.pad[4])

C2 = C_1206('C2\n0.01uF')
pcb = C2.add(pcb,R4.x,R4.y+.088,z)

pcb = wire(pcb,w,
   R4.pad[1],
   C2.pad[1])

pcb = wire(pcb,w,
   C2.pad[2],
   IC3.pad[4])

R5 = R_1206('R5\n1k')
pcb = R5.add(pcb,C2.x,C2.y+.088,z)

pcb = wire(pcb,w,
   R5.pad[2],
   IC3.pad[4])

pcb = wire(pcb,w,
   R2.pad[1],
   IC3.pad[5])

R6 = R_1206('R6\n49.9k')
pcb = R6.add(pcb,IC3.pad[3].x+.1,IC3.pad[3].y,z,angle=90)

pcb = wire(pcb,w,
   R6.pad[2],
   IC3.pad[2])

pcb = wire(pcb,w,
   IC3.pad[3],
   R6.pad[1])

R7 = R_1206('R7\n499k')
pcb = R7.add(pcb,XTAL1.x-.03,R6.y+.14,z,angle=90)

pcb = wire(pcb,w,
   R7.pad[2],
   R6.pad[1])

pcb = wire(pcb,w,
   IC3.pad[1],
   point(IC3.pad[1].x,IC3.y,z),
   point(IC3.pad[3].x+.04,R7.y,z),
   point(IC1.pad[6].x-.07,IC1.pad[6].y,z),
   IC1.pad[6])
```

```
pcb = wire(pcb,w,
   R7.pad[1],
   R2.pad[1])

C3 = C_1206('C3\n0.1uF')
pcb = C3.add(pcb,IC3.x-.01,IC3.y-.17,z)

pcb = wire(pcb,w,
   IC3.pad[3],
   point(R6.x+.06,C3.y,z),
   C3.pad[2])

# pads for 423-1043 electret

padx = IC3.x-.17
pady = IC3.y-.18

pcb = wire(pcb,.04,
   point(padx,pady,z),
   point(padx,pady+.1,z))

pcb = wire(pcb,.04,
   point(padx-.075,pady,z),
   point(padx-.075,pady+.1,z))

pcb = wire(pcb,w,
   C3.pad[1],
   point(padx,pady,z))

R8 = R_1206('R8\n10k')
pcb = R8.add(pcb,padx-.015,pady+.25,z,angle=90)

pcb = wire(pcb,w,
   R8.pad[2],
   point(padx,pady,z))

pcb = wire(pcb,w,
   R6.pad[2],
   point(R8.x+.06,R8.y,z),
   point(padx-.075,pady,x))

C4 = C_1206('C4\n1uF')
pcb = C4.add(pcb,R5.pad[1].x-.005,R5.y+.14,z,angle=90)

pcb = wire(pcb,w,
   C4.pad[1],
   J2.pad[1])

pcb = wire(pcb,w,
   C4.pad[2],
   R5.pad[1])

pcb = wire(pcb,w,
   R8.pad[1],
   point(R8.x,C4.y,z),
   IC3.pad[5])

C5 = C_1206('C5\n0.1uF')
pcb = C5.add(pcb,R8.x,J2.y-.025,z,angle=90)

pcb = wire(pcb,w,
   C5.pad[2],
   R8.pad[1])

pcb = wire(pcb,w,
   C5.pad[1],
   point(padx-.075,pady,z))

pcb = wire(pcb,w,
   C5.pad[1],
   point(C4.x-.025,C4.pad[1].y,z))

H1 = hello(w,z)
#pcb = H1.add(pcb,x+width-.4,y+.02,z)
pcb = H1.add(pcb,x+width-.02,y+.03,z,angle=-90)

#
# assign pcb to cad.function
#
# use single-sided FR2
# use 1/64 end-mill
# set tool diameter = .0156
# set xy, z speed = 4
# set # contours = -1

cad.function = pcb

#
# uncomment to mill out board
#
# use 1/32 end-mill
# set tool diameter = .0312
# set xy, z speed = .5
# set # contours = 1
#

#cad.function = frame
#z = -.065

#
# define limits and parameters
```

```
#

cad.xmin = x-.1 # min x to render
cad.xmax = x+width+.1 # max x to render
cad.ymin = y-.1 # min y to render
cad.ymax = y+width+.1 # max y to render
cad.zmin = z
cad.zmax = .050
dpi = 200 # low resolution for previewing
#dpi = 500 # high resolution for machining
nxy = int(dpi*(cad.xmax-cad.xmin))
cad.nx = nxy # x points to render
cad.ny = nxy # y points to render
cad.nz = 1
cad.view('xy') # 2D view
```

```
#
# hello.RGB.45.cad
#
# RGB LED hello-world
#
# Neil Gershenfeld
# CBA MIT 10/3/07
#
# (c) Massachusetts Institute of Technology 2007
# Permission granted for experimental and personal use;
# license for commercial sale available from MIT.
#

#
# define shapes and transformation
#
# circle(x0, y0, r)
# cylinder(x0, y0, z0, z1, r)
# cone(x0, y0, z0, z1, r0)
# sphere(x0, y0, z0, r)
# torus(x0, y0, z0, r0, r1)
# rectangle(x0, x1, y0, y1)
# cube(x0, x1, y0, y1, z0, z1)
# triangle(x0, y0, x1, y1, x2, y2) (points in clockwise order)
# pyramid(x0, x1, y0, y1, z0, z1)
# function(Z_of_XY)
# functions(upper_Z_of_XY,lower_Z_of_XY)
# add(part1, part2)
# subtract(part1, part2)
# intersect(part1, part2)
# move(part,dx,dy)
# translate(part,dx,dy,dz)
# rotate(part, angle)
# rotate_x(part, angle)
# rotate_y(part, angle)
# rotate_z(part, angle)
# rotate_90(part)
# rotate_180(part)
# rotate_270(part)
# reflect_x(part)
# reflect_y(part)
# reflect_z(part)
# reflect_xy(part)
# reflect_xz(part)
# reflect_yz(part)
# scale_x(part, x0, sx)
# scale_y(part, y0, sy)
# scale_z(part, z0, sz)
# scale_xy(part, x0, y0, sxy)
# scale_xyz(part, x0, y0, z0, sxyz)
# coscale_x_y(part, x0, y0, y1, angle0, angle1, amplitude, offset)
# coscale_x_z(part, x0, z0, z1, angle0, angle1, amplitude, offset)
# coscale_xy_z(part, x0, y0, z0, z1, angle0, angle1, amplitude, offset)
# taper_x_y(part, x0, y0, y1, s0, s1)
# taper_x_z(part, x0, z0, z1, s0, s1)
# taper_xy_z(part, x0, y0, z0, z1, s0, s1)
# shear_x_y(part, y0, y1, dx0, dx1)
# shear_x_z(part, z0, z1, dx0, dx1)
# (more to come)

def circle(x0, y0, r):
   part = "(((X-x0)**2 + (Y-y0)**2) <= r**2)"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'r',str(r))
   return part

def cylinder(x0, y0, z0, z1, r):
   part = "(((X-x0)**2 + (Y-y0)**2 <= r**2) & (Z >= z0) & (Z <= z1))"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'r',str(r))
   return part

def cone(x0, y0, z0, z1, r0):
   part = cylinder(x0, y0, z0, z1, r0)
   part = taper_xy_z(part, x0, y0, z0, z1, 1.0, 0.0)
   return part

def sphere(x0, y0, z0, r):
   part = "(((X-x0)**2 + (Y-y0)**2 + (Z-z0)**2) <= r**2)"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'r',str(r))
   return part

def torus(x0, y0, z0, r0, r1):
   part = "(((r0 - sqrt((X-x0)**2 + (Y-y0)**2))**2 + (Z-z0)**2) <= r1**2)"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'r0',str(r0))
   part = replace(part,'r1',str(r1))
   return part

def rectangle(x0, x1, y0, y1):
   part = "((X >= x0) & (X <= x1) & (Y >= y0) & (Y <= y1))"
   part = replace(part,'x0',str(x0))
```

```
   part = replace(part,'x1',str(x1))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   return part

def cube(x0, x1, y0, y1, z0, z1):
   part = "((X >= x0) & (X <= x1) & (Y >= y0) & (Y <= y1) & (Z >= z0) & (Z <= z1))"
   part = replace(part,'x0',str(x0))
   part = replace(part,'x1',str(x1))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   return part

def triangle(x0, y0, x1, y1, x2, y2): # points in clockwise order
   part = "((((y1-y0)*(X-x0)-(x1-x0)*(Y-y0)) >= 0) & (((y2-y1)*(X-x1)-(x2-x1)*(Y-y1)) >= 0) & (((y0-y2)*(X-x2)-(x0-x2)*(Y-y2)) >= 0))"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'x1',str(x1))
   part = replace(part,'y1',str(y1))
   part = replace(part,'x2',str(x2))
   part = replace(part,'y2',str(y2))
   return part

def pyramid(x0, x1, y0, y1, z0, z1):
   part = cube(x0, x1, y0, y1, z0, z1)
   part = taper_xy_z(part, (x0+x1)/2., (y0+y1)/2., z0, z1, 1.0, 0.0)
   return part

def function(Z_of_XY):
   part = '(Z <= '+Z_of_XY+')'
   return part

def functions(upper_Z_of_XY,lower_Z_of_XY):
   part = '(Z <= '+upper_Z_of_XY+') & (Z >= '+lower_Z_of_XY+')'
   return part

def add(part1, part2):
   part = "part1 | part2"
   part = replace(part,'part1',part1)
   part = replace(part,'part2',part2)
   return part

def subtract(part1, part2):
   part = "(part1) & ~(part2)"
   part = replace(part,'part1',part1)
   part = replace(part,'part2',part2)
   return part

def intersect(part1, part2):
   part = "(part1) & (part2)"
   part = replace(part,'part1',part1)
   part = replace(part,'part2',part2)
   return part

def move(part,dx,dy):
   part = replace(part,'X','(X-'+str(dx)+')')
   part = replace(part,'Y','(Y-'+str(dy)+')')
   return part

def translate(part,dx,dy,dz):
   part = replace(part,'X','(X-'+str(dx)+')')
   part = replace(part,'Y','(Y-'+str(dy)+')')
   part = replace(part,'Z','(Z-'+str(dz)+')')
   return part

def rotate(part, angle):
   angle = angle*pi/180
   part = replace(part,'X','(cos(angle)*X+sin(angle)*y)')
   part = replace(part,'Y','(-sin(angle)*X+cos(angle)*y)')
   part = replace(part,'y','Y')
   part = replace(part,'angle',str(angle))
   return part

def rotate_x(part, angle):
   angle = angle*pi/180
   part = replace(part,'Y','(cos(angle)*Y+sin(angle)*z)')
   part = replace(part,'Z','(-sin(angle)*Y+cos(angle)*z)')
   part = replace(part,'z','Z')
   part = replace(part,'angle',str(angle))
   return part

def rotate_y(part, angle):
   angle = angle*pi/180
   part = replace(part,'X','(cos(angle)*X+sin(angle)*z)')
   part = replace(part,'Z','(-sin(angle)*X+cos(angle)*z)')
   part = replace(part,'z','Z')
   part = replace(part,'angle',str(angle))
   return part

def rotate_z(part, angle):
   angle = angle*pi/180
   part = replace(part,'X','(cos(angle)*X+sin(angle)*y)')
   part = replace(part,'Y','(-sin(angle)*X+cos(angle)*y)')
   part = replace(part,'y','Y')
   part = replace(part,'angle',str(angle))
   return part

def rotate_90(part):
   part = reflect_xy(part)
```

```
   part = reflect_y(part)
   return part

def rotate_180(part):
   part = reflect_xy(part)
   part = reflect_y(part)
   part = reflect_xy(part)
   part = reflect_y(part)
   return part

def rotate_270(part):
   part = reflect_xy(part)
   part = reflect_y(part)
   part = reflect_xy(part)
   part = reflect_y(part)
   part = reflect_xy(part)
   part = reflect_y(part)
   return part

def reflect_x(part):
   part = replace(part,'X','-X')
   return part

def reflect_y(part):
   part = replace(part,'Y','-Y')
   return part

def reflect_z(part):
   part = replace(part,'Z','-Z')
   return part

def reflect_xy(part):
   part = replace(part,'X','temp')
   part = replace(part,'Y','X')
   part = replace(part,'temp','Y')
   return part

def reflect_xz(part):
   part = replace(part,'X','temp')
   part = replace(part,'Z','X')
   part = replace(part,'temp','Z')
   return part

def reflect_yz(part):
   part = replace(part,'Y','temp')
   part = replace(part,'Z','Y')
   part = replace(part,'temp','Z')
   return part

def scale_x(part, x0, sx):
   part = replace(part,'X','(x0 + (X-x0)/sx)')
   part = replace(part,'x0',str(x0))
   part = replace(part,'sx',str(sx))
   return part

def scale_y(part, y0, sy):
   part = replace(part,'Y','(y0 + (Y-y0)/sy)')
   part = replace(part,'y0',str(y0))
   part = replace(part,'sy',str(sy))
   return part

def scale_z(part, z0, sz):
   part = replace(part,'Z','(z0 + (Z-z0)/sz)')
   part = replace(part,'z0',str(z0))
   part = replace(part,'sz',str(sz))
   return part

def scale_xy(part, x0, y0, sxy):
   part = replace(part,'X','(x0 + (X-x0)/sxy)')
   part = replace(part,'Y','(y0 + (Y-y0)/sxy)')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'sxy',str(sxy))
   return part

def scale_xyz(part, x0, y0, z0, sxyz):
   part = replace(part,'X','(x0 + (X-x0)/sxyz)')
   part = replace(part,'Y','(y0 + (Y-y0)/sxyz)')
   part = replace(part,'Z','(z0 + (Z-z0)/sxyz)')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'sxyz',str(sxyz))
   return part

def coscale_x_y(part, x0, y0, y1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
   part = replace(part,'X','(x0 + (X-x0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Y-y0)/(y1-y0))))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

def coscale_x_z(part, x0, z0, z1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
```

```
   part = replace(part,'X','(x0 + (X-x0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0))))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

def coscale_xy_z(part, x0, y0, z0, z1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
   part = replace(part,'X','(x0 + (X-x0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0))))')
   part = replace(part,'Y','(y0 + (Y-y0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0))))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

def taper_x_y(part, x0, y0, y1, s0, s1):
   part = replace(part,'X','(x0 + (X-x0)*(y1-y0)/(s1*(Y-y0) + s0*(y1-Y)))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'s0',str(s0))
   part = replace(part,'s1',str(s1))
   return part

def taper_x_z(part, x0, z0, z1, s0, s1):
   part = replace(part,'X','(x0 + (X-x0)*(z1-z0)/(s1*(Z-z0) + s0*(z1-Z)))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'s0',str(s0))
   part = replace(part,'s1',str(s1))
   return part

def taper_xy_z(part, x0, y0, z0, z1, s0, s1):
   part = replace(part,'X','(x0 + (X-x0)*(z1-z0)/(s1*(Z-z0) + s0*(z1-Z)))')
   part = replace(part,'Y','(y0 + (Y-y0)*(z1-z0)/(s1*(Z-z0) + s0*(z1-Z)))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'s0',str(s0))
   part = replace(part,'s1',str(s1))
   return part

def shear_x_y(part, y0, y1, dx0, dx1):
   part = replace(part,'X','(X - dx0 - (dx1-dx0)*(Y-y0)/(y1-y0))')
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'dx0',str(dx0))
   part = replace(part,'dx1',str(dx1))
   return part

def shear_x_z(part, z0, z1, dx0, dx1):
   part = replace(part,'X','(X - dx0 - (dx1-dx0)*(Z-z0)/(z1-z0))')
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'dx0',str(dx0))
   part = replace(part,'dx1',str(dx1))
   return part

def coshear_x_z(part, z0, z1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
   part = replace(part,'X','(X - offset - amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0)))')
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

#
# PCB classes and definitions
#

cad.labels = []

class PCB:
   def __init__(self,x0,y0,width,height,mask):
      self.board = "False"
      self.interior = rectangle(x0,x0+width,y0,y0+height)
      self.exterior = subtract("True",rectangle(x0,x0+width,y0,y0+height))
      self.mask = "False"
   def add(self,part):
      self.board = add(self.board,part)
      self.mask = add(self.mask,move(part,-mask,mask))
      self.mask = add(self.mask,move(part,-mask,-mask))
      self.mask = add(self.mask,move(part,mask,mask))
      self.mask = add(self.mask,move(part,mask,-mask))
      return self
```

```
class part:
   def add(self,pcb,x,y,z=0,angle=0):
      self.x = x
      self.y = y
      self.z = z
      self.angle = angle
      if (angle == 90):
         self.shape = rotate_90(self.shape)
      elif (angle == 180):
         self.shape = rotate_180(self.shape)
      elif ((angle == 270) | (angle == -90)):
         self.shape = rotate_270(self.shape)
      angle = pi*angle/180
      self.shape = translate(self.shape,x,y,z)
      for i in range(len(self.pad)):
         xnew = cos(angle)*self.pad[i].x + sin(angle)*self.pad[i].y
         ynew = -sin(angle)*self.pad[i].x + cos(angle)*self.pad[i].y
           self.pad[i].x = x + xnew
           self.pad[i].y = y + ynew
         self.pad[i].z += z
      cad.labels.append(cad_text(x,y,z,self.value,size=14))
      for i in range(len(self.labels)):
         xnew = cos(angle)*self.labels[i].x + sin(angle)*self.labels[i].y
         ynew = -sin(angle)*self.labels[i].x + cos(angle)*self.labels[i].y
           self.labels[i].x = x + xnew
           self.labels[i].y = y + ynew
         self.labels[i].z += z
           cad.labels.append(self.labels[i])
      pcb = pcb.add(self.shape)
      return pcb

def wire(pcb,width,*points):
   for i in range(1,len(points)):
      x0 = points[i-1].x
      y0 = points[i-1].y
      z0 = points[i-1].z
      x1 = points[i].x
      y1 = points[i].y
      z1 = points[i].z
      if (x0 < x1):
         pcb.board = add(pcb.board,cube(x0-width/2,x1+width/2,y0-width/2,y0+width/2,z0,z0))
      elif (x1 < x0):
         pcb.board = add(pcb.board,cube(x1-width/2,x0+width/2,y0-width/2,y0+width/2,z0,z0))
      if (y0 < y1):
         pcb.board = add(pcb.board,cube(x1-width/2,x1+width/2,y0-width/2,y1+width/2,z0,z0))
      elif (y1 < y0):
         pcb.board = add(pcb.board,cube(x1-width/2,x1+width/2,y1-width/2,y0+width/2,z0,z0))
   return pcb

#
# PCB library
#

#
# discretes
#

pad_0402 = cube(-.0175,.0175,-.014,.014,0,0)

class R_0402(part):
   #
   # 0402 resistor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_0402,-.0265,0,0)
      self.pad.append(point(-.0265,0,0))
      self.shape = add(self.shape,translate(pad_0402,.0265,0,0))
      self.pad.append(point(.0265,0,0))

pad_1206 = cube(-.032,.032,-.034,.034,0,0)

class R_1206(part):
   #
   # 1206 resistor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.06,0,0))
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.06,0,0))

class C_1206(part):
   #
   # 1206 capacitor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.06,0,0))
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.06,0,0))
```

```
pad_1210 = cube(-.032,.032,-.048,.048,0,0)

class L_1210(part):
   #
   # 1210 inductor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_1210,-.06,0,0)
      self.pad.append(point(-.06,0,0))
      self.shape = add(self.shape,translate(pad_1210,.06,0,0))
      self.pad.append(point(.06,0,0))

pad_choke = cube(-.06,.06,-.06,.06,0,0)

class choke(part):
   #
   # Panasonic ELLCTV
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_choke,-.177,-.177,0)
      self.pad.append(point(-.177,-.177,0))
      self.shape = add(self.shape,translate(pad_choke,.177,.177,0))
      self.pad.append(point(.177,.177,0))

#
# connectors
#

pad_MTA = cube(-.021,.021,-.041,.041,0,0)
pad_MTA_solder = cube(-.071,.071,-.041,.041,0,0)

class MTA_2(part):
   #
   # AMP 1445121-2
   # MTA .050 SMT 2-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1
      #
      self.shape = translate(pad_MTA,-.025,-.1,0)
      self.pad.append(point(-.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,.1,0))
      self.pad.append(point(.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'2',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.187,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.187,0,0))

class MTA_power(part):
   #
   # AMP 1445121-2
   # MTA .050 SMT 2-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: Gnd
      #
      self.shape = translate(pad_MTA,-.025,-.1,0)
      self.pad.append(point(-.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: Vcc
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,.1,0))
      self.pad.append(point(.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Vcc',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.187,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.187,0,0))

class MTA_i0(part):
   #
   # AMP 1445121-3
   # MTA .050 SMT 3-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
```

```
      # pin 1: GND
      #
      self.shape = translate(pad_MTA,.05,.1,0)
      self.pad.append(point(.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1\nGND',14))
      #
      # pin 2: power
      #
      self.shape = add(self.shape,translate(pad_MTA,-.05,.1,0))
      self.pad.append(point(-.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V',14))
      #
      # pin 3: data
      #
      self.shape = add(self.shape,translate(pad_MTA,0,-.1,0))
      self.pad.append(point(0,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'data',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.212,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.212,0,0))

class MTA_serial(part):
   #
   # AMP 1445121-4
   # MTA .050 SMT 4-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: Gnd
      #
      self.shape = translate(pad_MTA,-.075,-.1,0)
      self.pad.append(point(-.075,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: Tx
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,-.1,0))
      self.pad.append(point(.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Tx',14))
      #
      # pin 3: Rx
      #
      self.shape = add(self.shape,translate(pad_MTA,.075,.1,0))
      self.pad.append(point(.075,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Rx',14))
      #
      # pin 4: DTR
      #
      self.shape = add(self.shape,translate(pad_MTA,-.025,.1,0))
      self.pad.append(point(-.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DTR',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.237,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.237,0,0))

class MTA_ICP(part):
   #
   # AMP 1445121-5
   # MTA .050 SMT 4-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: MISO
      #
      self.shape = translate(pad_MTA,-.1,-.1,0)
      self.pad.append(point(-.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'MISO',14))
      #
      # pin 2: GND
      #
      self.shape = add(self.shape,translate(pad_MTA,0,-.1,0))
      self.pad.append(point(0,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))
      #
      # pin 3: MOSI
      #
      self.shape = add(self.shape,translate(pad_MTA,.1,-.1,0))
      self.pad.append(point(.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'MOSI',14))
      #
      # pin 4: -RESET
      #
      self.shape = add(self.shape,translate(pad_MTA,.05,.1,0))
      self.pad.append(point(.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'-RESET',14))
      #
      # pin 5: SCK
      #
      self.shape = add(self.shape,translate(pad_MTA,-.05,.1,0))
      self.pad.append(point(-.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'SCK',14))
```

```
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.262,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.262,0,0))

class power_65mm(part):
   #
   # CUI PJ1-023-SMT
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: power
      #
      self.shape = cube(.433,.512,-.047,.047,0,0)
      self.pad.append(point(.467,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'P',14))
      #
      # pin 2: ground
      #
      self.shape = add(self.shape,cube(.285,.423,-.189,-.098,0,0))
      self.pad.append(point(.354,-.144,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 3: contact
      #
      self.shape = add(self.shape,cube(.325,.463,.098,.189,0,0))
      self.pad.append(point(.394,.144,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,cube(.108,.246,-.169,-.110,0,0))
      self.shape = add(self.shape,cube(.069,.207,.110,.169,0,0))

pad_stereo_2_5mm = cube(-.03,.03,-.05,.05,0,0)

class stereo_2_5mm(part):
   #
   # CUI SJ1-2533-SMT
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: base
      #
      self.shape = translate(pad_stereo_2_5mm,-.130,-.16,0)
      self.pad.append(point(-.130,-.149,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'base',14))
      #
      # pin 2: tip
      #
      self.shape = add(self.shape,translate(pad_stereo_2_5mm,.197,.15,0))
      self.pad.append(point(.197,.141,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'tip',14))
      #
      # pin 3: middle
      #
      self.shape = add(self.shape,translate(pad_stereo_2_5mm,-.012,-.16,0))
      self.pad.append(point(-.012,-.149,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'middle',14))

pad_Molex = cube(-.0155,.0155,-.0265,.0265,0,0)
pad_Molex_solder = cube(-.055,.055,-.065,.065,0,0)

class Molex_serial(part):
   #
   # Molex 53261-0471
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: Rx
      #
      self.shape = translate(pad_Molex,-.075,.064,0)
      self.pad.append(point(-.075,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Rx',14))
      #
      # pin 2: Tx
      #
      self.shape = add(self.shape,translate(pad_Molex,-.025,.064,0))
      self.pad.append(point(-.025,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Tx',14))
      #
      # pin 3: DTR
      #
      self.shape = add(self.shape,translate(pad_Molex,.025,.064,0))
      self.pad.append(point(.025,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DTR',14))
      #
      # pin 4: GND
      #
      self.shape = add(self.shape,translate(pad_Molex,.075,.064,0))
      self.pad.append(point(.075,.064,0))
```

```
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_Molex_solder,-.16,-.065,0))
      self.shape = add(self.shape,translate(pad_Molex_solder,.16,-.065,0))

#
# switches
#

pad_button_6mm = cube(-.04,.04,-.03,.03,0,0)

class button_6mm(part):
   #
   # Omron 6mm pushbutton
   # B3SN-3112P
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # left 1
      #
      self.shape = translate(pad_button_6mm,-.125,.08,0)
      self.pad.append(point(-.125,.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'L1',14))
      #
      # right 1
      #
      self.shape = add(self.shape,translate(pad_button_6mm,-.125,-.08,0))
      self.pad.append(point(-.125,-.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'R1',14))
      #
      # right 2
      #
      self.shape = add(self.shape,translate(pad_button_6mm,.125,-.08,0))
      self.pad.append(point(.125,-.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'R2',14))
      #
      # left 2
      #
      self.shape = add(self.shape,translate(pad_button_6mm,.125,.08,0))
      self.pad.append(point(.125,.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'L2',14))

#
# crystals and resonators
#

pad_XTAL = cube(-.016,.016,-.077,.077,0,0)

class XTAL(part):
   #
   # Panasonic EFOBM series
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # left
      #
      self.shape = translate(pad_XTAL,-.053,0,0)
      self.pad.append(point(-.053,0,0))
      #
      # ground
      #
      self.shape = add(self.shape,translate(pad_XTAL,0,0,0))
      self.pad.append(point(0,0,0))
      #
      # right
      #
      self.shape = add(self.shape,translate(pad_XTAL,.053,0,0))
      self.pad.append(point(.053,0,0))

#
# diodes, transistors, regulators
#

class D_1206(part):
   #
   # 1206 diode
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # anode
      #
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A',14))
      #
      # cathode
      #
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))
```

```
class LED_1206(part):
   #
   # 1206 LED
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # anode
      #
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A',14))
      #
      # cathode
      #
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))

pad_RGB = cube(-.021,.021,-.029,.029,0,0)

class LED_RGB(part):
   #
   # OSRAM LATBT686-QR+RS+KL-Z
   #
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      dx = .029
      dy = .059
      #
      # pin 1: red
      #
      self.shape = translate(pad_RGB,-dx,-dy,0)
      self.pad.append(point(-dx,-dy,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1R',14))
      #
      # pin 2: green
      #
      self.shape = add(self.shape,translate(pad_RGB,dx,-dy,0))
      self.pad.append(point(dx,-dy,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 3: blue
      #
      self.shape = add(self.shape,translate(pad_RGB,dx,dy,0))
      self.pad.append(point(dx,dy,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'B',14))
      #
      # pin 4: anode
      #
      self.shape = add(self.shape,translate(pad_RGB,-dx,dy,0))
      self.pad.append(point(-dx,dy,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A',14))

class phototransistor_1206(part):
   #
   # 1206 phototransistor
   # OPTEK 520,521,522
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # collector
      #
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))
      #
      # emitter
      #
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'E',14))

pad_SOD_123 = cube(-.02,.02,-.024,.024,0,0)

class D_SOD_123(part):
   #
   # SOD-123 diode
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # anode
      #
      self.shape = translate(pad_SOD_123,-.07,0,0)
      self.pad.append(point(-.07,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A',14))
      #
```

```
      # cathode
      #
      self.shape = add(self.shape,translate(pad_SOD_123,.07,0,0))
      self.pad.append(point(.07,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))

pad_SOT23 = cube(-.015,.015,-.02,.02,0,0)

class NMOSFET_SOT23(part):
   #
   # Fairchild NDS355AN
   #
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: gate
      #
      self.shape = translate(pad_SOT23,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 2: source
      #
      self.shape = add(self.shape,translate(pad_SOT23,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'S',14))
      #
      # pin 3: drain
      #
      self.shape = add(self.shape,translate(pad_SOT23,0,.045,0))
      self.pad.append(point(0,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'D',14))

class PMOSFET_SOT23(part):
   #
   # Fairchild NDS356AP
   #
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: gate
      #
      self.shape = translate(pad_SOT23,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 2: source
      #
      self.shape = add(self.shape,translate(pad_SOT23,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'S',14))
      #
      # pin 3: drain
      #
      self.shape = add(self.shape,translate(pad_SOT23,0,.045,0))
      self.pad.append(point(0,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'D',14))

class regulator_SOT23(part):
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: output
      #
      self.shape = translate(pad_SOT23,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: input
      #
      self.shape = add(self.shape,translate(pad_SOT23,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'in',14))
      #
      # pin 3: ground
      #
      self.shape = add(self.shape,translate(pad_SOT23,0,.045,0))
      self.pad.append(point(0,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'gnd',14))

pad_SOT223 = cube(-.02,.02,-.03,.03,0,0)
pad_SOT223_ground = cube(-.065,.065,-.03,.03,0,0)

class regulator_SOT223(part):
   def __init__(self,value=''):
      self.value = value
```

```
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: input
      #
      self.shape = translate(pad_SOT223,-.09,-.12,0)
      self.pad.append(point(-.09,-.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1I',14))
      #
      # pin 2: ground
      #
      self.shape = add(self.shape,translate(pad_SOT223,0,-.12,0))
      self.pad.append(point(0,-.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 3: output
      #
      self.shape = add(self.shape,translate(pad_SOT223,.09,-.12,0))
      self.pad.append(point(.09,-.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'O',14))
      #
      # pin 4: ground
      #
      self.shape = add(self.shape,translate(pad_SOT223_ground,0,.12,0))
      self.pad.append(point(0,.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))

pad_SM8 = cube(-.035,.035,-.016,.016,0,0)

class H_bridge_SM8(part):
   #
   # Zetex ZXMHC3A01T8
   #
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      d = .13
      #
      # pin 1: G3 (right N gate)
      #
      self.shape = translate(pad_SM8,-d,.09,0)
      self.pad.append(point(-d,.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GRN',14))
      #
      # pin 2: S2 S3 (N source)
      #
      self.shape = add(self.shape,translate(pad_SM8,-d,.03,0))
      self.pad.append(point(-d,.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'SN',14))
      #
      # pin 3: G2 (left N gate)
      #
      self.shape = add(self.shape,translate(pad_SM8,-d,-.03,0))
      self.pad.append(point(-d,-.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GLN',14))
      #
      # pin 4: G1 (left P gate)
      #
      self.shape = add(self.shape,translate(pad_SM8,-d,-.09,0))
      self.pad.append(point(-d,-.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GLP',14))
      #
      # pin 5: D1 D2 (left drain)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,-.09,0))
      self.pad.append(point(d,-.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DL',14))
      #
      # pin 6: S1 S4 (P source)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,-.03,0))
      self.pad.append(point(d,-.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'SP',14))
      #
      # pin 7: D3 D4 (right drain)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,.03,0))
      self.pad.append(point(d,.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DR',14))
      #
      # pin 8: G4 (right N gate)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,.09,0))
      self.pad.append(point(d,.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GRP',14))

#
# ICs
#

pad_SOT23_5 = cube(-.01,.01,-.02,.02,0,0)

class op_amp_SOT23_5(part):
   def __init__(self,value=''):
```

```
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: output
      #
      self.shape = translate(pad_SOT23_5,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'O',14))
      #
      # pin 2: V-
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,0,-.045,0))
      self.pad.append(point(0,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V-',14))
      #
      # pin 3: I+
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'I+',14))
      #
      # pin 4: I-
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,.0375,.045,0))
      self.pad.append(point(.0375,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'I-',14))
      #
      # pin 5: V+
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,-.0375,.045,0))
      self.pad.append(point(-.0375,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V+',14))

pad_SOICN = cube(-.035,.035,-.015,.015,0,0)

class op_amp_SOICN(part):
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: A out
      #
      self.shape = translate(pad_SOICN,-.12,.075,0)
      self.pad.append(point(-.12,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1 Ao',14))
      #
      # pin 2: A-
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.025,0))
      self.pad.append(point(-.12,.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A-',14))
      #
      # pin 3: A+
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.025,0))
      self.pad.append(point(-.12,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A+',14))
      #
      # pin 4: V-
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.075,0))
      self.pad.append(point(-.12,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V-',14))
      #
      # pin 5: B+
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.075,0))
      self.pad.append(point(.12,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'B+',14))
      #
      # pin 6: B-
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.025,0))
      self.pad.append(point(.12,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'B-',14))
      #
      # pin 7: B out
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.025,0))
      self.pad.append(point(.12,.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Bo',14))
      #
      # pin 8: V+
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.075,0))
      self.pad.append(point(.12,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V+',14))

pad_SOIC = cube(-.043,.043,-.015,.015,0,0)

class ATtiny45_SOIC(part):
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
```

```
        # pin 1: PB5/dW/ADC0/-RESET/PCINT5
        #
        self.shape = translate(pad_SOIC,-.14,.075,0)
        self.pad.append(point(-.14,.075,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
        #
        # pin 2: PB3/ADC3/-OC1B/CLKI/XTAL1/PCINT3
        #
        self.shape = add(self.shape,translate(pad_SOIC,-.14,.025,0))
        self.pad.append(point(-.14,.025,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB3',14))
        #
        # pin 3: PB4/ADC2/OC1B/CLKO/XTAL2/PCINT4
        #
        self.shape = add(self.shape,translate(pad_SOIC,-.14,-.025,0))
        self.pad.append(point(-.14,-.025,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB4',14))
        #
        # pin 4: GND
        #
        self.shape = add(self.shape,translate(pad_SOIC,-.14,-.075,0))
        self.pad.append(point(-.14,-.075,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))
        #
        # pin 5: PB0/MOSI/DI/SDA/AIN0/OC0A/-OC1A/AREF/PCINT0
        #
        self.shape = add(self.shape,translate(pad_SOIC,.14,-.075,0))
        self.pad.append(point(.14,-.075,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 6: PB1/MISO/DO/AIN1/OC0B/OC1A/PCINT1
        #
        self.shape = add(self.shape,translate(pad_SOIC,.14,-.025,0))
        self.pad.append(point(.14,-.025,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB1',14))
        #
        # pin 7: PB2/SCK/USCK/SCL/ADC1/T0/INT0/PCINT2
        #
        self.shape = add(self.shape,translate(pad_SOIC,.14,.025,0))
        self.pad.append(point(.14,.025,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB2',14))
        #
        # pin 8: VCC
        #
        self.shape = add(self.shape,translate(pad_SOIC,.14,.075,0))
        self.pad.append(point(.14,.075,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'VCC',14))

class ATtiny44_SOICN(part):
   def __init__(self,value=''):
        self.value = value
        self.pad = [point(0,0,0)]
        self.labels = []
        #
        # pin 1: VCC
        #
        self.shape = translate(pad_SOICN,-.12,.15,0)
        self.pad.append(point(-.12,.15,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
        #
        # pin 2: PB0/XTAL1/PCINT8
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 3: PB1/XTAL2/PCINT9
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.050,0))
        self.pad.append(point(-.12,.05,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB1',14))
        #
        # pin 4: PB3/dW/-RESET/PCINT11
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,0,0))
        self.pad.append(point(-.12,0,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB3',14))
        #
        # pin 5: PB2/CKOUT/OC0A/INT0/PCINT10
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,-.05,0))
        self.pad.append(point(-.12,-.05,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB2',14))
        #
        # pin 6: PA7/ADC7/OC0B/ICP/PCINT7
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,-.1,0))
        self.pad.append(point(-.12,-.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA7',14))
        #
        # pin 7: PA6/ADC6/MOSI/SDA/OC1A/PCINT6
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,-.15,0))
        self.pad.append(point(-.12,-.15,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA6',14))
        #
        # pin 8: PA5/ADC5/DO/MISO/OC1B/PCINT5
        #
        self.shape = add(self.shape,translate(pad_SOICN,.12,-.15,0))
        self.pad.append(point(.12,-.15,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA5',14))
```

```
        #
        # pin 9: PA4/ADC4/USCK/SCL/T1/PCINT4
        #
        self.shape = add(self.shape,translate(pad_SOICN,.12,-.1,0))
        self.pad.append(point(.12,-.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA4',14))
        #
        # pin 10: PA3/ADC3/T0/PCINT3
        #
        self.shape = add(self.shape,translate(pad_SOICN,.12,-.05,0))
        self.pad.append(point(.12,-.05,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA3',14))
        #
        # pin 11: PA2/ADC2/AIN1/PCINT2
        #
        self.shape = add(self.shape,translate(pad_SOICN,.12,0,0))
        self.pad.append(point(.12,0,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA2',14))
        #
        # pin 12: PA1/ADC1/AIN0/PCINT1
        #
        self.shape = add(self.shape,translate(pad_SOICN,.12,.050,0))
        self.pad.append(point(.12,.05,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA1',14))
        #
        # pin 13: PA0/ADC0/AREF/PCINT0
        #
        self.shape = add(self.shape,translate(pad_SOICN,.12,.1,0))
        self.pad.append(point(.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA0',14))
        #
        # pin 14: GND
        #
        self.shape = add(self.shape,translate(pad_SOICN,.12,.15,0))
        self.pad.append(point(.12,.15,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))

pad_TQFP = cube(-.043,.043,-.015,.015,0,0)

class ATmega88_TQFP(part):
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []

      #
      # pin 1: PD3/PCINT19/OC2B/INT1
      #
      self.shape = translate(pad_SOICN,-.12,.15,0)
      self.pad.append(point(-.12,.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: PD4/PCINT20/XCK/T0
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 3: GND
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 4: VCC
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 5: GND
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 6: VCC
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 7: PB6/PCINT6/XTAL1/TOSC1
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 8: PB7/PCINT7/XTAL2/TOSC2
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

      #
      # pin 9: PD5/PCINT21/OC0B/T1
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
```

```
# pin 10: PD6/PCINT22/OC0A/AIN0
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 11: PD7/PCINT23/AIN1
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 12: PB0/PCINT0/CLKO/ICP1
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 13: PB1/PCINT1/OC1A
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 14: PB2/PCINT2/-SS/OC1B
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 15: PB3/PCINT3/OC2A/MOSI
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 16: PB4/PCINT4/MISO
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

#
# pin 17: PB5/SCK/PCINT5
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 18: AVCC
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 19: ADC6
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 20: AREF
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 21: GND
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 22: ADC7
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 23: PC0/ADC0/PCINT8
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 24: PC1/ADC1/PCINT9
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

#
# pin 25: PC2/ADC2/PCINT10
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 26: PC2/ADC3/PCINT11
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
```

```
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 27: PC4/ADC4/SDA/PCINT12
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 28: PC5/ADC5/SCL/PCINT13
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 29: PC6/-RESET/PCINT14
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 30: PD0/RXD/PCINT16
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 31: PD1/TXD/PCINT17
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 32: PD2/INT0/PCINT18
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

#
# graphics
#

class CBA(part):
   def __init__(self,r=.02):
      self.value = ''
      self.pad = [point(0,0,0)]
      self.labels = []
      d = 3*r
      self.shape = circle(0,0,r)
      self.shape = add(self.shape,translate(circle(0,0,r),-d,d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),-d,0,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),-d,-d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),0,-d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),d,-d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),d,0,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),d,d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),0,d,0))

class hello(part):
   #
   # HELLO
   #
   def __init__(self,w,z):
      self.value = ''
      self.pad = [point(0,0,0)]
      self.labels = []
      # H
      self.shape = rectangle(-.01,.01,0,.1)
      self.shape = add(self.shape,rectangle(0,.05,.04,.06))
      self.shape = add(self.shape,rectangle(.04,.06,0,.1))
      # E
      self.shape = add(self.shape,rectangle(.08,.1,0,.1))
      self.shape = add(self.shape,rectangle(.1,.14,0,.02))
      self.shape = add(self.shape,rectangle(.1,.13,.04,.06))
      self.shape = add(self.shape,rectangle(.1,.14,.08,.1))
      # L
      self.shape = add(self.shape,rectangle(.16,.18,0,.1))
      self.shape = add(self.shape,rectangle(.18,.21,0,.02))
      # L
      self.shape = add(self.shape,rectangle(.23,.25,0,.1))
      self.shape = add(self.shape,rectangle(.25,.28,0,.02))
      # 0
      self.shape = add(self.shape,rectangle(.3,.32,0,.1))
      self.shape = add(self.shape,rectangle(.32,.345,0,.02))
      self.shape = add(self.shape,rectangle(.32,.345,.08,.1))
      self.shape = add(self.shape,rectangle(.345,.365,0,.1))

#
# define board
#

x0 = 1
y0 = 1
width = 1.32
height = .7
z = -.005
w = .018
mask = .004

pcb = PCB(x0,y0,width,height,mask)
```

```
IC1 = ATtiny45_SOIC('IC1\nt45')
pcb = IC1.add(pcb,x0+0.88,y0+.35,z,angle=-90)

R1 = R_1206('R1\n499')
pcb = R1.add(pcb,IC1.x-.22,IC1.y+.07,z)

pcb = wire(pcb,w,
   point(R1.pad[2].x,R1.pad[2].y-.025,z),
   IC1.pad[7])

R2 = R_1206('R2\n1k')
pcb = R2.add(pcb,R1.x,IC1.y-.07,z)

pcb = wire(pcb,w,
   point(R2.pad[2].x,R2.pad[2].y+.025,z),
   IC1.pad[3])

R3 = R_1206('R3\n1k')
pcb = R3.add(pcb,R2.x,R2.y-.17,z)

pcb = wire(pcb,w,
   R3.pad[2],
   IC1.pad[2])

LED1 = LED_RGB('LED1')
pcb = LED1.add(pcb,R2.x-.17,IC1.y,z)

pcb = wire(pcb,w,
   R3.pad[1],
   LED1.pad[1])

pcb = wire(pcb,w,
   LED1.pad[2],
   R2.pad[1])

pcb = wire(pcb,w,
   LED1.pad[3],
   R1.pad[1])

R4 = R_1206('R4\n10k')
pcb = R4.add(pcb,LED1.x-.12,IC1.y,z,angle=90)

pcb = wire(pcb,w,
   IC1.pad[8],
   R4.pad[1])

pcb = wire(pcb,w,
   LED1.pad[4],
   R4.pad[1])

pcb = wire(pcb,w,
   IC1.pad[1],
   point(R3.x,R3.y-.07,z),
   R4.pad[2])

J1 = MTA_power('J1')
pcb = J1.add(pcb,x0+.17,y0+.28,z,angle=-90)

IC2 = regulator_SOT23('IC2\n5V')
pcb = IC2.add(pcb,J1.x-.1,J1.y+.33,z,angle=180)

pcb = wire(pcb,w,
   J1.pad[2],
   IC2.pad[2])

C1 = C_1206('C1\n3.3uF')
pcb = C1.add(pcb,J1.x+.12,IC2.y-.02,z,angle=90)

pcb = wire(pcb,w,
   C1.pad[1],
   IC2.pad[1])

pcb = wire(pcb,w,
   C1.pad[2],
   point(C1.pad[2].x,IC1.y,z),
   J1.pad[1])

pcb = wire(pcb,w,
   IC2.pad[3],
   C1.pad[2])

pcb = wire(pcb,w,
   C1.pad[1],
   point(C1.pad[1].x,C1.y+.07,z),
   R4.pad[1])

J2 = MTA_ICP('J2')
pcb = J2.add(pcb,IC1.x+.28,IC1.y,z,angle=90)

pcb = wire(pcb,w,
   IC1.pad[6],
   point(IC1.pad[6].x,IC1.pad[6].y+.1,z),
   point(J2.pad[1].x-.03,J2.pad[1].y,z),
   J2.pad[1])

pcb = wire(pcb,w,
   J2.pad[2],
   IC1.pad[4])

pcb = wire(pcb,w,
```

```
    J2.pad[2],
    J1.pad[1])

pcb = wire(pcb,w,
    J2.pad[3],
    point(J2.x,J2.pad[5].y,z),
    IC1.pad[5])

pcb = wire(pcb,w,
    J2.pad[4],
    point(J2.pad[4].x,J2.pad[3].y-.05,z),
    point(J2.pad[3].x,R3.y-.07,z),
    point(R3.x,R3.y,z))

pcb = wire(pcb,w,
    IC1.pad[7],
    point(IC1.pad[7].x,IC1.pad[7].y+.15,z),
    point(J2.pad[1].x+.02,J2.pad[1].y+.05,z),
    J2.pad[5])

H1 = hello(w,z)
pcb = H1.add(pcb,R4.x+.06,y0+height-.13,z)

#
# assign board and exterior to cad.function for milling
#
# use single-sided FR2
# use 1/64 end-mill
# set tool diameter = .0156
# set xy, z speed = 4
# set # contours = -1
#

cad.function = add(pcb.board,pcb.exterior)

#
# uncomment to export traces
#

#cad.function = pcb.board

#
# uncomment to export exterior
#

#cad.function = pcb.exterior

#
# uncomment to export solder mask
#

#cad.function = pcb.mask

#
# uncomment to mill out board
#
# use 1/32 end-mill
# set tool diameter = .0312
# set xy, z speed = .5
# set # contours = 1
#

#cad.function = pcb.interior
#z = -.065

#
# define limits and parameters
#

d = 0.05
cad.xmin = x0-d # min x to render
cad.xmax = x0+width+d # max x to render
cad.ymin = y0-d # min y to render
cad.ymax = y0+height+d # max y to render
cad.zmin = z
cad.zmax = .050
dpi = 200 # low resolution for previewing
#dpi = 500 # high resolution for machining
nxy = int(dpi*(cad.xmax-cad.xmin))
cad.nx = int(dpi*(cad.xmax-cad.xmin)) # x points to render
cad.ny = int(dpi*(cad.ymax-cad.ymin)) # y points to render
cad.nz = 1
cad.inches_per_unit = 1.0 # use inch units
cad.view('xy') # 2D view
```

```
#
# hello.speaker.45.cad
#
# tiny45 speaker hello-world
#
# Neil Gershenfeld
# CBA MIT 8/1/07
#
# (c) Massachusetts Institute of Technology 2007
# Permission granted for experimental and personal use;
# license for commercial sale available from MIT.
#

#
# define shapes and transformation
#
# circle(x0, y0, r)
# cylinder(x0, y0, z0, z1, r)
# cone(x0, y0, z0, z1, r0)
# sphere(x0, y0, z0, r)
# torus(x0, y0, z0, r0, r1)
# rectangle(x0, x1, y0, y1)
# cube(x0, x1, y0, y1, z0, z1)
# triangle(x0, y0, x1, y1, x2, y2) (points in clockwise order)
# pyramid(x0, x1, y0, y1, z0, z1)
# function(Z_of_XY)
# functions(upper_Z_of_XY,lower_Z_of_XY)
# add(part1, part2)
# subtract(part1, part2)
# intersect(part1, part2)
# move(part,dx,dy)
# translate(part,dx,dy,dz)
# rotate(part, angle)
# rotate_x(part, angle)
# rotate_y(part, angle)
# rotate_z(part, angle)
# rotate_90(part)
# rotate_180(part)
# rotate_270(part)
# reflect_x(part)
# reflect_y(part)
# reflect_z(part)
# reflect_xy(part)
# reflect_xz(part)
# reflect_yz(part)
# scale_x(part, x0, sx)
# scale_y(part, y0, sy)
# scale_z(part, z0, sz)
# scale_xy(part, x0, y0, sxy)
# scale_xyz(part, x0, y0, z0, sxyz)
# coscale_x_y(part, x0, y0, y1, angle0, angle1, amplitude, offset)
# coscale_x_z(part, x0, z0, z1, angle0, angle1, amplitude, offset)
# coscale_xy_z(part, x0, y0, z0, z1, angle0, angle1, amplitude, offset)
# taper_x_y(part, x0, y0, y1, s0, s1)
# taper_x_z(part, x0, z0, z1, s0, s1)
# taper_xy_z(part, x0, y0, z0, z1, s0, s1)
# shear_x_y(part, y0, y1, dx0, dx1)
# shear_x_z(part, z0, z1, dx0, dx1)
# (more to come)

def circle(x0, y0, r):
   part = "(((X-x0)**2 + (Y-y0)**2) <= r**2)"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'r',str(r))
   return part

def cylinder(x0, y0, z0, z1, r):
   part = "(((X-x0)**2 + (Y-y0)**2 <= r**2) & (Z >= z0) & (Z <= z1))"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'r',str(r))
   return part

def cone(x0, y0, z0, z1, r0):
   part = cylinder(x0, y0, z0, z1, r0)
   part = taper_xy_z(part, x0, y0, z0, z1, 1.0, 0.0)
   return part

def sphere(x0, y0, z0, r):
   part = "(((X-x0)**2 + (Y-y0)**2 + (Z-z0)**2) <= r**2)"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'r',str(r))
   return part

def torus(x0, y0, z0, r0, r1):
   part = "(((r0 - sqrt((X-x0)**2 + (Y-y0)**2))**2 + (Z-z0)**2) <= r1**2)"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'r0',str(r0))
   part = replace(part,'r1',str(r1))
   return part

def rectangle(x0, x1, y0, y1):
   part = "((X >= x0) & (X <= x1) & (Y >= y0) & (Y <= y1))"
   part = replace(part,'x0',str(x0))
```

```
   part = replace(part,'x1',str(x1))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   return part

def cube(x0, x1, y0, y1, z0, z1):
   part = "((X >= x0) & (X <= x1) & (Y >= y0) & (Y <= y1) & (Z >= z0) & (Z <= z1))"
   part = replace(part,'x0',str(x0))
   part = replace(part,'x1',str(x1))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   return part

def triangle(x0, y0, x1, y1, x2, y2): # points in clockwise order
   part = "((((y1-y0)*(X-x0)-(x1-x0)*(Y-y0)) >= 0) & (((y2-y1)*(X-x1)-(x2-x1)*(Y-y1)) >= 0) & (((y0-y2)*(X-x2)-(x0-x2)*(Y-y2)) >= 0))"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'x1',str(x1))
   part = replace(part,'y1',str(y1))
   part = replace(part,'x2',str(x2))
   part = replace(part,'y2',str(y2))
   return part

def pyramid(x0, x1, y0, y1, z0, z1):
   part = cube(x0, x1, y0, y1, z0, z1)
   part = taper_xy_z(part, (x0+x1)/2., (y0+y1)/2., z0, z1, 1.0, 0.0)
   return part

def function(Z_of_XY):
   part = '(Z <= '+Z_of_XY+')'
   return part

def functions(upper_Z_of_XY,lower_Z_of_XY):
   part = '(Z <= '+upper_Z_of_XY+') & (Z >= '+lower_Z_of_XY+')'
   return part

def add(part1, part2):
   part = "part1 | part2"
   part = replace(part,'part1',part1)
   part = replace(part,'part2',part2)
   return part

def subtract(part1, part2):
   part = "(part1) & ~(part2)"
   part = replace(part,'part1',part1)
   part = replace(part,'part2',part2)
   return part

def intersect(part1, part2):
   part = "(part1) & (part2)"
   part = replace(part,'part1',part1)
   part = replace(part,'part2',part2)
   return part

def move(part,dx,dy):
   part = replace(part,'X','(X-'+str(dx)+')')
   part = replace(part,'Y','(Y-'+str(dy)+')')
   return part

def translate(part,dx,dy,dz):
   part = replace(part,'X','(X-'+str(dx)+')')
   part = replace(part,'Y','(Y-'+str(dy)+')')
   part = replace(part,'Z','(Z-'+str(dz)+')')
   return part

def rotate(part, angle):
   angle = angle*pi/180
   part = replace(part,'X','(cos(angle)*X+sin(angle)*y)')
   part = replace(part,'Y','(-sin(angle)*X+cos(angle)*y)')
   part = replace(part,'y','Y')
   part = replace(part,'angle',str(angle))
   return part

def rotate_x(part, angle):
   angle = angle*pi/180
   part = replace(part,'Y','(cos(angle)*Y+sin(angle)*z)')
   part = replace(part,'Z','(-sin(angle)*Y+cos(angle)*z)')
   part = replace(part,'z','Z')
   part = replace(part,'angle',str(angle))
   return part

def rotate_y(part, angle):
   angle = angle*pi/180
   part = replace(part,'X','(cos(angle)*X+sin(angle)*z)')
   part = replace(part,'Z','(-sin(angle)*X+cos(angle)*z)')
   part = replace(part,'z','Z')
   part = replace(part,'angle',str(angle))
   return part

def rotate_z(part, angle):
   angle = angle*pi/180
   part = replace(part,'X','(cos(angle)*X+sin(angle)*y)')
   part = replace(part,'Y','(-sin(angle)*X+cos(angle)*y)')
   part = replace(part,'y','Y')
   part = replace(part,'angle',str(angle))
   return part

def rotate_90(part):
   part = reflect_xy(part)
```

```
   part = reflect_y(part)
   return part

def rotate_180(part):
   part = reflect_xy(part)
   part = reflect_y(part)
   part = reflect_xy(part)
   part = reflect_y(part)
   return part

def rotate_270(part):
   part = reflect_xy(part)
   part = reflect_y(part)
   part = reflect_xy(part)
   part = reflect_y(part)
   part = reflect_xy(part)
   part = reflect_y(part)
   return part

def reflect_x(part):
   part = replace(part,'X','-X')
   return part

def reflect_y(part):
   part = replace(part,'Y','-Y')
   return part

def reflect_z(part):
   part = replace(part,'Z','-Z')
   return part

def reflect_xy(part):
   part = replace(part,'X','temp')
   part = replace(part,'Y','X')
   part = replace(part,'temp','Y')
   return part

def reflect_xz(part):
   part = replace(part,'X','temp')
   part = replace(part,'Z','X')
   part = replace(part,'temp','Z')
   return part

def reflect_yz(part):
   part = replace(part,'Y','temp')
   part = replace(part,'Z','Y')
   part = replace(part,'temp','Z')
   return part

def scale_x(part, x0, sx):
   part = replace(part,'X','(x0 + (X-x0)/sx)')
   part = replace(part,'x0',str(x0))
   part = replace(part,'sx',str(sx))
   return part

def scale_y(part, y0, sy):
   part = replace(part,'Y','(y0 + (Y-y0)/sy)')
   part = replace(part,'y0',str(y0))
   part = replace(part,'sy',str(sy))
   return part

def scale_z(part, z0, sz):
   part = replace(part,'Z','(z0 + (Z-z0)/sz)')
   part = replace(part,'z0',str(z0))
   part = replace(part,'sz',str(sz))
   return part

def scale_xy(part, x0, y0, sxy):
   part = replace(part,'X','(x0 + (X-x0)/sxy)')
   part = replace(part,'Y','(y0 + (Y-y0)/sxy)')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'sxy',str(sxy))
   return part

def scale_xyz(part, x0, y0, z0, sxyz):
   part = replace(part,'X','(x0 + (X-x0)/sxyz)')
   part = replace(part,'Y','(y0 + (Y-y0)/sxyz)')
   part = replace(part,'Z','(z0 + (Z-z0)/sxyz)')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'sxyz',str(sxyz))
   return part

def coscale_x_y(part, x0, y0, y1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
   part = replace(part,'X','(x0 + (X-x0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Y-y0)/(y1-y0))))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

def coscale_x_z(part, x0, z0, z1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
```

```
   part = replace(part,'X','(x0 + (X-x0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0))))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

def coscale_xy_z(part, x0, y0, z0, z1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
   part = replace(part,'X','(x0 + (X-x0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0))))')
   part = replace(part,'Y','(y0 + (Y-y0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0))))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

def taper_x_y(part, x0, y0, y1, s0, s1):
   part = replace(part,'X','(x0 + (X-x0)*(y1-y0)/(s1*(Y-y0) + s0*(y1-Y)))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'s0',str(s0))
   part = replace(part,'s1',str(s1))
   return part

def taper_x_z(part, x0, z0, z1, s0, s1):
   part = replace(part,'X','(x0 + (X-x0)*(z1-z0)/(s1*(Z-z0) + s0*(z1-Z)))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'s0',str(s0))
   part = replace(part,'s1',str(s1))
   return part

def taper_xy_z(part, x0, y0, z0, z1, s0, s1):
   part = replace(part,'X','(x0 + (X-x0)*(z1-z0)/(s1*(Z-z0) + s0*(z1-Z)))')
   part = replace(part,'Y','(y0 + (Y-y0)*(z1-z0)/(s1*(Z-z0) + s0*(z1-Z)))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'s0',str(s0))
   part = replace(part,'s1',str(s1))
   return part

def shear_x_y(part, y0, y1, dx0, dx1):
   part = replace(part,'X','(X - dx0 - (dx1-dx0)*(Y-y0)/(y1-y0))')
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'dx0',str(dx0))
   part = replace(part,'dx1',str(dx1))
   return part

def shear_x_z(part, z0, z1, dx0, dx1):
   part = replace(part,'X','(X - dx0 - (dx1-dx0)*(Z-z0)/(z1-z0))')
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'dx0',str(dx0))
   part = replace(part,'dx1',str(dx1))
   return part

def coshear_x_z(part, z0, z1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
   part = replace(part,'X','(X - offset - amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0)))')
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

#
# PCB classes and definitions
#

cad.labels = []

class PCB:
   def __init__(self,x0,y0,width,height,mask):
      self.board = "False"
      self.interior = rectangle(x0,x0+width,y0,y0+height)
      self.exterior = subtract("True",rectangle(x0,x0+width,y0,y0+height))
      self.mask = "False"
   def add(self,part):
      self.board = add(self.board,part)
      self.mask = add(self.mask,move(part,-mask,mask))
      self.mask = add(self.mask,move(part,-mask,-mask))
      self.mask = add(self.mask,move(part,mask,mask))
      self.mask = add(self.mask,move(part,mask,-mask))
      return self
```

```
class part:
   def add(self,pcb,x,y,z=0,angle=0):
      self.x = x
      self.y = y
      self.z = z
      self.angle = angle
      if (angle == 90):
         self.shape = rotate_90(self.shape)
      elif (angle == 180):
         self.shape = rotate_180(self.shape)
      elif ((angle == 270) | (angle == -90)):
         self.shape = rotate_270(self.shape)
      angle = pi*angle/180
      self.shape = translate(self.shape,x,y,z)
      for i in range(len(self.pad)):
         xnew = cos(angle)*self.pad[i].x + sin(angle)*self.pad[i].y
         ynew = -sin(angle)*self.pad[i].x + cos(angle)*self.pad[i].y
           self.pad[i].x = x + xnew
           self.pad[i].y = y + ynew
         self.pad[i].z += z
      cad.labels.append(cad_text(x,y,z,self.value,size=14))
      for i in range(len(self.labels)):
         xnew = cos(angle)*self.labels[i].x + sin(angle)*self.labels[i].y
         ynew = -sin(angle)*self.labels[i].x + cos(angle)*self.labels[i].y
           self.labels[i].x = x + xnew
           self.labels[i].y = y + ynew
         self.labels[i].z += z
           cad.labels.append(self.labels[i])
      pcb = pcb.add(self.shape)
      return pcb

def wire(pcb,width,*points):
   for i in range(1,len(points)):
      x0 = points[i-1].x
      y0 = points[i-1].y
      z0 = points[i-1].z
      x1 = points[i].x
      y1 = points[i].y
      z1 = points[i].z
      if (x0 < x1):
         pcb.board = add(pcb.board,cube(x0-width/2,x1+width/2,y0-width/2,y0+width/2,z0,z0))
      elif (x1 < x0):
         pcb.board = add(pcb.board,cube(x1-width/2,x0+width/2,y0-width/2,y0+width/2,z0,z0))
      if (y0 < y1):
         pcb.board = add(pcb.board,cube(x1-width/2,x1+width/2,y0-width/2,y1+width/2,z0,z0))
      elif (y1 < y0):
         pcb.board = add(pcb.board,cube(x1-width/2,x1+width/2,y1-width/2,y0+width/2,z0,z0))
   return pcb

#
# PCB library
#

#
# discretes
#

pad_0402 = cube(-.0175,.0175,-.014,.014,0,0)

class R_0402(part):
   #
   # 0402 resistor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_0402,-.0265,0,0)
      self.pad.append(point(-.0265,0,0))
      self.shape = add(self.shape,translate(pad_0402,.0265,0,0))
      self.pad.append(point(.0265,0,0))

pad_1206 = cube(-.032,.032,-.034,.034,0,0)

class R_1206(part):
   #
   # 1206 resistor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))

class C_1206(part):
   #
   # 1206 capacitor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))
```

```
pad_1210 = cube(-.032,.032,-.048,.048,0,0)

class L_1210(part):
   #
   # 1210 inductor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_1210,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.shape = add(self.shape,translate(pad_1210,.06,0,0))
      self.pad.append(point(.055,0,0))

pad_choke = cube(-.06,.06,-.06,.06,0,0)

class choke(part):
   #
   # Panasonic ELLCTV
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_choke,-.177,-.177,0)
      self.pad.append(point(-.177,-.177,0))
      self.shape = add(self.shape,translate(pad_choke,.177,.177,0))
      self.pad.append(point(.177,.177,0))

#
# connectors
#

pad_MTA = cube(-.021,.021,-.041,.041,0,0)
pad_MTA_solder = cube(-.071,.071,-.041,.041,0,0)

class MTA_2(part):
   #
   # AMP 1445121-2
   # MTA .050 SMT 2-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1
      #
      self.shape = translate(pad_MTA,-.025,-.1,0)
      self.pad.append(point(-.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,.1,0))
      self.pad.append(point(.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'2',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.187,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.187,0,0))

class MTA_power(part):
   #
   # AMP 1445121-2
   # MTA .050 SMT 2-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: Gnd
      #
      self.shape = translate(pad_MTA,-.025,-.1,0)
      self.pad.append(point(-.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: Vcc
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,.1,0))
      self.pad.append(point(.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Vcc',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.187,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.187,0,0))

class MTA_i0(part):
   #
   # AMP 1445121-3
   # MTA .050 SMT 3-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
```

```
      # pin 1: GND
      #
      self.shape = translate(pad_MTA,.05,.1,0)
      self.pad.append(point(.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1\nGND',14))
      #
      # pin 2: power
      #
      self.shape = add(self.shape,translate(pad_MTA,-.05,.1,0))
      self.pad.append(point(-.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V',14))
      #
      # pin 3: data
      #
      self.shape = add(self.shape,translate(pad_MTA,0,-.1,0))
      self.pad.append(point(0,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'data',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.212,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.212,0,0))

class MTA_serial(part):
   #
   # AMP 1445121-4
   # MTA .050 SMT 4-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: Gnd
      #
      self.shape = translate(pad_MTA,-.075,-.1,0)
      self.pad.append(point(-.075,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: Tx
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,-.1,0))
      self.pad.append(point(.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Tx',14))
      #
      # pin 3: Rx
      #
      self.shape = add(self.shape,translate(pad_MTA,.075,.1,0))
      self.pad.append(point(.075,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Rx',14))
      #
      # pin 4: DTR
      #
      self.shape = add(self.shape,translate(pad_MTA,-.025,.1,0))
      self.pad.append(point(-.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DTR',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.237,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.237,0,0))

class MTA_ICP(part):
   #
   # AMP 1445121-5
   # MTA .050 SMT 4-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: MISO
      #
      self.shape = translate(pad_MTA,-.1,-.1,0)
      self.pad.append(point(-.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'MISO',14))
      #
      # pin 2: GND
      #
      self.shape = add(self.shape,translate(pad_MTA,0,-.1,0))
      self.pad.append(point(0,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))
      #
      # pin 3: MOSI
      #
      self.shape = add(self.shape,translate(pad_MTA,.1,-.1,0))
      self.pad.append(point(.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'MOSI',14))
      #
      # pin 4: -RESET
      #
      self.shape = add(self.shape,translate(pad_MTA,.05,.1,0))
      self.pad.append(point(.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'-RESET',14))
      #
      # pin 5: SCK
      #
      self.shape = add(self.shape,translate(pad_MTA,-.05,.1,0))
      self.pad.append(point(-.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'SCK',14))
```

```
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.262,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.262,0,0))

class power_65mm(part):
   #
   # CUI PJ1-023-SMT
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: power
      #
      self.shape = cube(.433,.512,-.047,.047,0,0)
      self.pad.append(point(.467,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'P',14))
      #
      # pin 2: ground
      #
      self.shape = add(self.shape,cube(.285,.423,-.189,-.098,0,0))
      self.pad.append(point(.354,-.144,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 3: contact
      #
      self.shape = add(self.shape,cube(.325,.463,.098,.189,0,0))
      self.pad.append(point(.394,.144,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,cube(.108,.246,-.169,-.110,0,0))
      self.shape = add(self.shape,cube(.069,.207,.110,.169,0,0))

pad_stereo_2_5mm = cube(-.03,.03,-.05,.05,0,0)

class stereo_2_5mm(part):
   #
   # CUI SJ1-2533-SMT
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: base
      #
      self.shape = translate(pad_stereo_2_5mm,-.130,-.16,0)
      self.pad.append(point(-.130,-.149,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'base',14))
      #
      # pin 2: tip
      #
      self.shape = add(self.shape,translate(pad_stereo_2_5mm,.197,.15,0))
      self.pad.append(point(.197,.141,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'tip',14))
      #
      # pin 3: middle
      #
      self.shape = add(self.shape,translate(pad_stereo_2_5mm,-.012,-.16,0))
      self.pad.append(point(-.012,-.149,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'middle',14))

pad_Molex = cube(-.0155,.0155,-.0265,.0265,0,0)
pad_Molex_solder = cube(-.055,.055,-.065,.065,0,0)

class Molex_serial(part):
   #
   # Molex 53261-0471
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: Rx
      #
      self.shape = translate(pad_Molex,-.075,.064,0)
      self.pad.append(point(-.075,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Rx',14))
      #
      # pin 2: Tx
      #
      self.shape = add(self.shape,translate(pad_Molex,-.025,.064,0))
      self.pad.append(point(-.025,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Tx',14))
      #
      # pin 3: DTR
      #
      self.shape = add(self.shape,translate(pad_Molex,.025,.064,0))
      self.pad.append(point(.025,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DTR',14))
      #
      # pin 4: GND
      #
      self.shape = add(self.shape,translate(pad_Molex,.075,.064,0))
      self.pad.append(point(.075,.064,0))
```

```
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_Molex_solder,-.16,-.065,0))
      self.shape = add(self.shape,translate(pad_Molex_solder,.16,-.065,0))

#
# switches
#

pad_button_6mm = cube(-.04,.04,-.03,.03,0,0)

class button_6mm(part):
   #
   # Omron 6mm pushbutton
   # B3SN-3112P
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # left 1
      #
      self.shape = translate(pad_button_6mm,-.125,.08,0)
      self.pad.append(point(-.125,.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'L1',14))
      #
      # right 1
      #
      self.shape = add(self.shape,translate(pad_button_6mm,-.125,-.08,0))
      self.pad.append(point(-.125,-.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'R1',14))
      #
      # right 2
      #
      self.shape = add(self.shape,translate(pad_button_6mm,.125,-.08,0))
      self.pad.append(point(.125,-.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'R2',14))
      #
      # left 2
      #
      self.shape = add(self.shape,translate(pad_button_6mm,.125,.08,0))
      self.pad.append(point(.125,.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'L2',14))

#
# crystals and resonators
#

pad_XTAL = cube(-.016,.016,-.077,.077,0,0)

class XTAL(part):
   #
   # Panasonic EFOBM series
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # left
      #
      self.shape = translate(pad_XTAL,-.053,0,0)
      self.pad.append(point(-.053,0,0))
      #
      # ground
      #
      self.shape = add(self.shape,translate(pad_XTAL,0,0,0))
      self.pad.append(point(0,0,0))
      #
      # right
      #
      self.shape = add(self.shape,translate(pad_XTAL,.053,0,0))
      self.pad.append(point(.053,0,0))

#
# diodes, transistors, regulators
#

class D_1206(part):
   #
   # 1206 diode
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # anode
      #
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A',14))
      #
      # cathode
      #
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))
```

```
class LED_1206(part):
   #
   # 1206 LED
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # anode
      #
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A',14))
      #
      # cathode
      #
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))

class phototransistor_1206(part):
   #
   # 1206 phototransistor
   # OPTEK 520,521
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # collector
      #
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))
      #
      # emitter
      #
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'E',14))

pad_SOD_123 = cube(-.02,.02,-.024,.024,0,0)

class D_SOD_123(part):
   #
   # SOD-123 diode
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # anode
      #
      self.shape = translate(pad_SOD_123,-.07,0,0)
      self.pad.append(point(-.07,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A',14))
      #
      # cathode
      #
      self.shape = add(self.shape,translate(pad_SOD_123,.07,0,0))
      self.pad.append(point(.07,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))

pad_SOT23 = cube(-.015,.015,-.02,.02,0,0)

class NMOSFET_SOT23(part):
   #
   # Fairchild NDS355AN
   #
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: gate
      #
      self.shape = translate(pad_SOT23,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 2: source
      #
      self.shape = add(self.shape,translate(pad_SOT23,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'S',14))
      #
      # pin 3: drain
      #
      self.shape = add(self.shape,translate(pad_SOT23,0,.045,0))
      self.pad.append(point(0,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'D',14))

class PMOSFET_SOT23(part):
   #
```

```
   # Fairchild NDS356AP
   #
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: gate
      #
      self.shape = translate(pad_SOT23,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 2: source
      #
      self.shape = add(self.shape,translate(pad_SOT23,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'S',14))
      #
      # pin 3: drain
      #
      self.shape = add(self.shape,translate(pad_SOT23,0,.045,0))
      self.pad.append(point(0,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'D',14))

class regulator_SOT23(part):
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: output
      #
      self.shape = translate(pad_SOT23,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: input
      #
      self.shape = add(self.shape,translate(pad_SOT23,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'in',14))
      #
      # pin 3: ground
      #
      self.shape = add(self.shape,translate(pad_SOT23,0,.045,0))
      self.pad.append(point(0,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'gnd',14))

pad_SOT223 = cube(-.02,.02,-.03,.03,0,0)
pad_SOT223_ground = cube(-.065,.065,-.03,.03,0,0)

class regulator_SOT223(part):
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: input
      #
      self.shape = translate(pad_SOT223,-.09,-.12,0)
      self.pad.append(point(-.09,-.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1I',14))
      #
      # pin 2: ground
      #
      self.shape = add(self.shape,translate(pad_SOT223,0,-.12,0))
      self.pad.append(point(0,-.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 3: output
      #
      self.shape = add(self.shape,translate(pad_SOT223,.09,-.12,0))
      self.pad.append(point(.09,-.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'O',14))
      #
      # pin 4: ground
      #
      self.shape = add(self.shape,translate(pad_SOT223_ground,0,.12,0))
      self.pad.append(point(0,.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))

pad_SM8 = cube(-.035,.035,-.016,.016,0,0)

class H_bridge_SM8(part):
   #
   # Zetex ZXMHC3A01T8
   #
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
```

```
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      d = .13
      #
      # pin 1: G3 (right N gate)
      #
      self.shape = translate(pad_SM8,-d,.09,0)
      self.pad.append(point(-d,.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GRN',14))
      #
      # pin 2: S2 S3 (N source)
      #
      self.shape = add(self.shape,translate(pad_SM8,-d,.03,0))
      self.pad.append(point(-d,.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'SN',14))
      #
      # pin 3: G2 (left N gate)
      #
      self.shape = add(self.shape,translate(pad_SM8,-d,-.03,0))
      self.pad.append(point(-d,-.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GLN',14))
      #
      # pin 4: G1 (left P gate)
      #
      self.shape = add(self.shape,translate(pad_SM8,-d,-.09,0))
      self.pad.append(point(-d,-.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GLP',14))
      #
      # pin 5: D1 D2 (left drain)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,-.09,0))
      self.pad.append(point(d,-.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DL',14))
      #
      # pin 6: S1 S4 (P source)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,-.03,0))
      self.pad.append(point(d,-.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'SP',14))
      #
      # pin 7: D3 D4 (right drain)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,.03,0))
      self.pad.append(point(d,.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DR',14))
      #
      # pin 8: G4 (right N gate)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,.09,0))
      self.pad.append(point(d,.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GRP',14))

#
# ICs
#

pad_SOT23_5 = cube(-.01,.01,-.02,.02,0,0)

class op_amp_SOT23_5(part):
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: output
      #
      self.shape = translate(pad_SOT23_5,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'O',14))
      #
      # pin 2: V-
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,0,-.045,0))
      self.pad.append(point(0,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V-',14))
      #
      # pin 3: I+
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'I+',14))
      #
      # pin 4: I-
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,.0375,.045,0))
      self.pad.append(point(.0375,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'I-',14))
      #
      # pin 5: V+
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,-.0375,.045,0))
      self.pad.append(point(-.0375,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V+',14))

pad_SOICN = cube(-.035,.035,-.015,.015,0,0)

class op_amp_SOICN(part):
```

```
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: A out
      #
      self.shape = translate(pad_SOICN,-.12,.075,0)
      self.pad.append(point(-.12,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1 Ao',14))
      #
      # pin 2: A-
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.025,0))
      self.pad.append(point(-.12,.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A-',14))
      #
      # pin 3: A+
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.025,0))
      self.pad.append(point(-.12,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A+',14))
      #
      # pin 4: V-
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.075,0))
      self.pad.append(point(-.12,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V-',14))
      #
      # pin 5: B+
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.075,0))
      self.pad.append(point(.12,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'B+',14))
      #
      # pin 6: B-
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.025,0))
      self.pad.append(point(.12,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'B-',14))
      #
      # pin 7: B out
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.025,0))
      self.pad.append(point(.12,.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Bo',14))
      #
      # pin 8: V+
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.075,0))
      self.pad.append(point(.12,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V+',14))

pad_SOIC = cube(-.043,.043,-.015,.015,0,0)

class ATtiny45_SOIC(part):
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: PB5/dW/ADC0/-RESET/PCINT5
      #
      self.shape = translate(pad_SOIC,-.14,.075,0)
      self.pad.append(point(-.14,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: PB3/ADC3/-OC1B/CLKI/XTAL1/PCINT3
      #
      self.shape = add(self.shape,translate(pad_SOIC,-.14,.025,0))
      self.pad.append(point(-.14,.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB3',14))
      #
      # pin 3: PB4/ADC2/OC1B/CLKO/XTAL2/PCINT4
      #
      self.shape = add(self.shape,translate(pad_SOIC,-.14,-.025,0))
      self.pad.append(point(-.14,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB4',14))
      #
      # pin 4: GND
      #
      self.shape = add(self.shape,translate(pad_SOIC,-.14,-.075,0))
      self.pad.append(point(-.14,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))
      #
      # pin 5: PB0/MOSI/DI/SDA/AIN0/OC0A/-OC1A/AREF/PCINT0
      #
      self.shape = add(self.shape,translate(pad_SOIC,.14,-.075,0))
      self.pad.append(point(.14,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 6: PB1/MISO/DO/AIN1/OC0B/OC1A/PCINT1
      #
      self.shape = add(self.shape,translate(pad_SOIC,.14,-.025,0))
      self.pad.append(point(.14,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB1',14))
      #
      # pin 7: PB2/SCK/USCK/SCL/ADC1/T0/INT0/PCINT2
      #
      self.shape = add(self.shape,translate(pad_SOIC,.14,.025,0))
      self.pad.append(point(.14,.025,0))
```

```
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB2',14))
      #
      # pin 8: VCC
      #
      self.shape = add(self.shape,translate(pad_SOIC,.14,.075,0))
      self.pad.append(point(.14,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'VCC',14))

class ATtiny44_SOICN(part):
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: VCC
      #
      self.shape = translate(pad_SOICN,-.12,.15,0)
      self.pad.append(point(-.12,.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: PB0/XTAL1/PCINT8
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 3: PB1/XTAL2/PCINT9
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.050,0))
      self.pad.append(point(-.12,.05,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB1',14))
      #
      # pin 4: PB3/dW/-RESET/PCINT11
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,0,0))
      self.pad.append(point(-.12,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB3',14))
      #
      # pin 5: PB2/CKOUT/OC0A/INT0/PCINT10
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.05,0))
      self.pad.append(point(-.12,-.05,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB2',14))
      #
      # pin 6: PA7/ADC7/OC0B/ICP/PCINT7
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.1,0))
      self.pad.append(point(-.12,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA7',14))
      #
      # pin 7: PA6/ADC6/MOSI/SDA/OC1A/PCINT6
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.15,0))
      self.pad.append(point(-.12,-.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA6',14))
      #
      # pin 8: PA5/ADC5/DO/MISO/OC1B/PCINT5
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.15,0))
      self.pad.append(point(.12,-.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA5',14))
      #
      # pin 9: PA4/ADC4/USCK/SCL/T1/PCINT4
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.1,0))
      self.pad.append(point(.12,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA4',14))
      #
      # pin 10: PA3/ADC3/T0/PCINT3
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.05,0))
      self.pad.append(point(.12,-.05,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA3',14))
      #
      # pin 11: PA2/ADC2/AIN1/PCINT2
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,0,0))
      self.pad.append(point(.12,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA2',14))
      #
      # pin 12: PA1/ADC1/AIN0/PCINT1
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.050,0))
      self.pad.append(point(.12,.05,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA1',14))
      #
      # pin 13: PA0/ADC0/AREF/PCINT0
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.1,0))
      self.pad.append(point(.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA0',14))
      #
      # pin 14: GND
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.15,0))
      self.pad.append(point(.12,.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))

pad_TQFP = cube(-.043,.043,-.015,.015,0,0)

class ATmega88_TQFP(part):
```

```
def __init__(self,value=''):
   self.value = value
   self.pad = [point(0,0,0)]
   self.labels = []

   #
   # pin 1: PD3/PCINT19/OC2B/INT1
   #
   self.shape = translate(pad_SOICN,-.12,.15,0)
   self.pad.append(point(-.12,.15,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
   #
   # pin 2: PD4/PCINT20/XCK/T0
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 3: GND
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 4: VCC
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 5: GND
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 6: VCC
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 7: PB6/PCINT6/XTAL1/TOSC1
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 8: PB7/PCINT7/XTAL2/TOSC2
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

   #
   # pin 9: PD5/PCINT21/OC0B/T1
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 10: PD6/PCINT22/OC0A/AIN0
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 11: PD7/PCINT23/AIN1
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 12: PB0/PCINT0/CLKO/ICP1
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 13: PB1/PCINT1/OC1A
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 14: PB2/PCINT2/-SS/OC1B
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 15: PB3/PCINT3/OC2A/MOSI
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 16: PB4/PCINT4/MISO
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
```

```
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

        #
        # pin 17: PB5/SCK/PCINT5
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 18: AVCC
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 19: ADC6
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 20: AREF
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 21: GND
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 22: ADC7
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 23: PC0/ADC0/PCINT8
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 24: PC1/ADC1/PCINT9
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

        #
        # pin 25: PC2/ADC2/PCINT10
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 26: PC2/ADC3/PCINT11
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 27: PC4/ADC4/SDA/PCINT12
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 28: PC5/ADC5/SCL/PCINT13
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 29: PC6/-RESET/PCINT14
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 30: PD0/RXD/PCINT16
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 31: PD1/TXD/PCINT17
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 32: PD2/INT0/PCINT18
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

#
```

```
# graphics
#

class CBA(part):
   def __init__(self,r=.02):
      self.value = ''
      self.pad = [point(0,0,0)]
      self.labels = []
      d = 3*r
      self.shape = circle(0,0,r)
      self.shape = add(self.shape,translate(circle(0,0,r),-d,d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),-d,0,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),-d,-d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),0,-d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),d,-d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),d,0,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),d,d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),0,d,0))

class hello(part):
   #
   # HELLO
   #
   def __init__(self,w,z):
      self.value = ''
      self.pad = [point(0,0,0)]
      self.labels = []
      # H
      self.shape = rectangle(-.01,.01,0,.1)
      self.shape = add(self.shape,rectangle(0,.05,.04,.06))
      self.shape = add(self.shape,rectangle(.04,.06,0,.1))
      # E
      self.shape = add(self.shape,rectangle(.08,.1,0,.1))
      self.shape = add(self.shape,rectangle(.1,.14,0,.02))
      self.shape = add(self.shape,rectangle(.1,.13,.04,.06))
      self.shape = add(self.shape,rectangle(.1,.14,.08,.1))
      # L
      self.shape = add(self.shape,rectangle(.16,.18,0,.1))
      self.shape = add(self.shape,rectangle(.18,.21,0,.02))
      # L
      self.shape = add(self.shape,rectangle(.23,.25,0,.1))
      self.shape = add(self.shape,rectangle(.25,.28,0,.02))
      # 0
      self.shape = add(self.shape,rectangle(.3,.32,0,.1))
      self.shape = add(self.shape,rectangle(.32,.345,0,.02))
      self.shape = add(self.shape,rectangle(.32,.345,.08,.1))
      self.shape = add(self.shape,rectangle(.345,.365,0,.1))

#
# define board
#

x0 = 1
y0 = 1
width = 1.33
height = .79
z = -.005
w = .018
mask = .004

pcb = PCB(x0,y0,width,height,mask)

IC1 = ATtiny45_SOIC('IC1\nt45')
pcb = IC1.add(pcb,x0+.77,y0+.45,z,angle=-90)

R1 = R_1206('R1\n10k')
pcb = R1.add(pcb,IC1.pad[1].x-.1,IC1.y,z,angle=90)

pcb = wire(pcb,w,
   IC1.pad[1],
   R1.pad[2])

pcb = wire(pcb,w,
   IC1.pad[8],
   R1.pad[1])

J1 = MTA_power('J1')
pcb = J1.add(pcb,x0+.17,y0+.5,z,angle=-90)

IC2 = regulator_SOT23('IC2\n5V')
pcb = IC2.add(pcb,J1.x+.19,J1.y+.15,z,angle=180)

pcb = wire(pcb,w,
   J1.pad[1],
   IC2.pad[3])

pcb = wire(pcb,w,
   J1.pad[2],
   point(J1.x,J1.pad[2].y+.05,z),
   point(IC2.pad[2].x-.05,IC2.pad[2].y,z),
   IC2.pad[2])

C1 = C_1206('C1\n3.3uF')
pcb = C1.add(pcb,IC2.x+.13,IC2.y,z,angle=90)

pcb = wire(pcb,w,
   IC2.pad[1],
   C1.pad[1])

pcb = wire(pcb,w,
   IC2.pad[3],
```

```
    C1.pad[2])

pcb = wire(pcb,w,
    C1.pad[1],
    IC1.pad[8])

J2 = MTA_ICP('J2')
pcb = J2.add(pcb,IC1.x+.35,IC1.y-.02,z,angle=90)

pcb = wire(pcb,w,
    IC1.pad[6],
    point(IC1.pad[6].x,IC1.pad[6].y+.1,z),
    point(J2.pad[1].x-.03,J2.pad[1].y,z),
    J2.pad[1])

pcb = wire(pcb,w,
    J2.pad[3],
    point(J2.pad[3].x-.09,IC1.y+.03,z),
    IC1.pad[5])

pcb = wire(pcb,w,
    J2.pad[4],
    point(J2.pad[4].x,J2.pad[3].y-.05,z),
    point(IC1.pad[4].x+.05,IC1.y-.05,z),
    IC1.pad[1])

cpcb = wire(pcb,w,
    IC1.pad[7],
    point(IC1.pad[7].x,IC1.pad[7].y+.15,z),
    point(J2.pad[1].x+.02,J2.pad[1].y+.05,z),
    J2.pad[5])

J3 = MTA_2('J3')
pcb = J3.add(pcb,x0+.38,y0+.16,z,angle=180)

pcb = wire(pcb,w,
    J3.pad[2],
    J1.pad[2])

pcb = wire(pcb,w,
    J1.pad[1],
    point(J1.pad[1].x,J3.pad[1].y,z),
    point(J3.pad[2].x-.05,J3.y,z),
    point(J3.pad[1].x+.05,J3.pad[2].y,z),
    point(J2.pad[4].x+.08,J2.y,z),
    J2.pad[2])

pcb = wire(pcb,w,
    IC1.pad[4],
    point(IC1.pad[4].x,IC1.pad[4].y-.08,z),
    point(J3.pad[1].x+.05,J3.y,z))

T1 = NMOSFET_SOT23('T1: N')
pcb = T1.add(pcb,J3.x+.02,R1.y-.02,z,angle=180)

pcb = wire(pcb,w,
    J1.pad[1],
    T1.pad[2])

pcb = wire(pcb,w,
    J3.pad[1],
    T1.pad[3])

pcb = wire(pcb,w,
    T1.pad[1],
    point(R1.x-.09,R1.y,z),
    IC1.pad[6])


H1 = hello(w,z)
pcb = H1.add(pcb,J3.x+.3,J3.y-.06,z)

#
# assign board and exterior to cad.function for milling
#
# use single-sided FR2
# use 1/64 end-mill
# set tool diameter = .0156
# set xy, z speed = 4
# set # contours = -1
#

cad.function = add(pcb.board,pcb.exterior)

#
# uncomment to export traces
#

#cad.function = pcb.board

#
# uncomment to export exterior
#

#cad.function = pcb.exterior

#
# uncomment to export solder mask
#

#cad.function = pcb.mask
```

```
#
# uncomment to mill out board
#
# use 1/32 end-mill
# set tool diameter = .0312
# set xy, z speed = .5
# set # contours = 1
#

#cad.function = pcb.interior
#z = -.065

#
# define limits and parameters
#

d = 0.05
cad.xmin = x0-d # min x to render
cad.xmax = x0+width+d # max x to render
cad.ymin = y0-d # min y to render
cad.ymax = y0+height+d # max y to render
cad.zmin = z
cad.zmax = .050
dpi = 200 # low resolution for previewing
#dpi = 500 # high resolution for machining
nxy = int(dpi*(cad.xmax-cad.xmin))
cad.nx = int(dpi*(cad.xmax-cad.xmin)) # x points to render
cad.ny = int(dpi*(cad.ymax-cad.ymin)) # y points to render
cad.nz = 1
cad.inches_per_unit = 1.0 # use inch units
cad.view('xy') # 2D view
```

```
#
# hello.H-bridge.44.cad
#
# H-bridge hello-world
#
# Neil Gershenfeld
# CBA MIT 10/17/07
#
# (c) Massachusetts Institute of Technology 2007
# Permission granted for experimental and personal use;
# license for commercial sale available from MIT.
#

#
# define shapes and transformation
#
# circle(x0, y0, r)
# cylinder(x0, y0, z0, z1, r)
# cone(x0, y0, z0, z1, r0)
# sphere(x0, y0, z0, r)
# torus(x0, y0, z0, r0, r1)
# rectangle(x0, x1, y0, y1)
# cube(x0, x1, y0, y1, z0, z1)
# triangle(x0, y0, x1, y1, x2, y2) (points in clockwise order)
# pyramid(x0, x1, y0, y1, z0, z1)
# function(Z_of_XY)
# functions(upper_Z_of_XY,lower_Z_of_XY)
# add(part1, part2)
# subtract(part1, part2)
# intersect(part1, part2)
# move(part,dx,dy)
# translate(part,dx,dy,dz)
# rotate(part, angle)
# rotate_x(part, angle)
# rotate_y(part, angle)
# rotate_z(part, angle)
# rotate_90(part)
# rotate_180(part)
# rotate_270(part)
# reflect_x(part)
# reflect_y(part)
# reflect_z(part)
# reflect_xy(part)
# reflect_xz(part)
# reflect_yz(part)
# scale_x(part, x0, sx)
# scale_y(part, y0, sy)
# scale_z(part, z0, sz)
# scale_xy(part, x0, y0, sxy)
# scale_xyz(part, x0, y0, z0, sxyz)
# coscale_x_y(part, x0, y0, y1, angle0, angle1, amplitude, offset)
# coscale_x_z(part, x0, z0, z1, angle0, angle1, amplitude, offset)
# coscale_xy_z(part, x0, y0, z0, z1, angle0, angle1, amplitude, offset)
# taper_x_y(part, x0, y0, y1, s0, s1)
# taper_x_z(part, x0, z0, z1, s0, s1)
# taper_xy_z(part, x0, y0, z0, z1, s0, s1)
# shear_x_y(part, y0, y1, dx0, dx1)
# shear_x_z(part, z0, z1, dx0, dx1)
# (more to come)

def circle(x0, y0, r):
   part = "(((X-x0)**2 + (Y-y0)**2) <= r**2)"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'r',str(r))
   return part

def cylinder(x0, y0, z0, z1, r):
   part = "(((X-x0)**2 + (Y-y0)**2 <= r**2) & (Z >= z0) & (Z <= z1))"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'r',str(r))
   return part

def cone(x0, y0, z0, z1, r0):
   part = cylinder(x0, y0, z0, z1, r0)
   part = taper_xy_z(part, x0, y0, z0, z1, 1.0, 0.0)
   return part

def sphere(x0, y0, z0, r):
   part = "(((X-x0)**2 + (Y-y0)**2 + (Z-z0)**2) <= r**2)"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'r',str(r))
   return part

def torus(x0, y0, z0, r0, r1):
   part = "(((r0 - sqrt((X-x0)**2 + (Y-y0)**2))**2 + (Z-z0)**2) <= r1**2)"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'r0',str(r0))
   part = replace(part,'r1',str(r1))
   return part

def rectangle(x0, x1, y0, y1):
   part = "((X >= x0) & (X <= x1) & (Y >= y0) & (Y <= y1))"
   part = replace(part,'x0',str(x0))
```

```
   part = replace(part,'x1',str(x1))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   return part

def cube(x0, x1, y0, y1, z0, z1):
   part = "((X >= x0) & (X <= x1) & (Y >= y0) & (Y <= y1) & (Z >= z0) & (Z <= z1))"
   part = replace(part,'x0',str(x0))
   part = replace(part,'x1',str(x1))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   return part

def triangle(x0, y0, x1, y1, x2, y2): # points in clockwise order
   part = "((((y1-y0)*(X-x0)-(x1-x0)*(Y-y0)) >= 0) & (((y2-y1)*(X-x1)-(x2-x1)*(Y-y1)) >= 0) & (((y0-y2)*(X-x2)-(x0-x2)*(Y-y2)) >= 0))"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'x1',str(x1))
   part = replace(part,'y1',str(y1))
   part = replace(part,'x2',str(x2))
   part = replace(part,'y2',str(y2))
   return part

def pyramid(x0, x1, y0, y1, z0, z1):
   part = cube(x0, x1, y0, y1, z0, z1)
   part = taper_xy_z(part, (x0+x1)/2., (y0+y1)/2., z0, z1, 1.0, 0.0)
   return part

def function(Z_of_XY):
   part = '(Z <= '+Z_of_XY+')'
   return part

def functions(upper_Z_of_XY,lower_Z_of_XY):
   part = '(Z <= '+upper_Z_of_XY+') & (Z >= '+lower_Z_of_XY+')'
   return part

def add(part1, part2):
   part = "part1 | part2"
   part = replace(part,'part1',part1)
   part = replace(part,'part2',part2)
   return part

def subtract(part1, part2):
   part = "(part1) & ~(part2)"
   part = replace(part,'part1',part1)
   part = replace(part,'part2',part2)
   return part

def intersect(part1, part2):
   part = "(part1) & (part2)"
   part = replace(part,'part1',part1)
   part = replace(part,'part2',part2)
   return part

def move(part,dx,dy):
   part = replace(part,'X','(X-'+str(dx)+')')
   part = replace(part,'Y','(Y-'+str(dy)+')')
   return part

def translate(part,dx,dy,dz):
   part = replace(part,'X','(X-'+str(dx)+')')
   part = replace(part,'Y','(Y-'+str(dy)+')')
   part = replace(part,'Z','(Z-'+str(dz)+')')
   return part

def rotate(part, angle):
   angle = angle*pi/180
   part = replace(part,'X','(cos(angle)*X+sin(angle)*y)')
   part = replace(part,'Y','(-sin(angle)*X+cos(angle)*y)')
   part = replace(part,'y','Y')
   part = replace(part,'angle',str(angle))
   return part

def rotate_x(part, angle):
   angle = angle*pi/180
   part = replace(part,'Y','(cos(angle)*Y+sin(angle)*z)')
   part = replace(part,'Z','(-sin(angle)*Y+cos(angle)*z)')
   part = replace(part,'z','Z')
   part = replace(part,'angle',str(angle))
   return part

def rotate_y(part, angle):
   angle = angle*pi/180
   part = replace(part,'X','(cos(angle)*X+sin(angle)*z)')
   part = replace(part,'Z','(-sin(angle)*X+cos(angle)*z)')
   part = replace(part,'z','Z')
   part = replace(part,'angle',str(angle))
   return part

def rotate_z(part, angle):
   angle = angle*pi/180
   part = replace(part,'X','(cos(angle)*X+sin(angle)*y)')
   part = replace(part,'Y','(-sin(angle)*X+cos(angle)*y)')
   part = replace(part,'y','Y')
   part = replace(part,'angle',str(angle))
   return part

def rotate_90(part):
   part = reflect_xy(part)
```

```
   part = reflect_y(part)
   return part

def rotate_180(part):
   part = reflect_xy(part)
   part = reflect_y(part)
   part = reflect_xy(part)
   part = reflect_y(part)
   return part

def rotate_270(part):
   part = reflect_xy(part)
   part = reflect_y(part)
   part = reflect_xy(part)
   part = reflect_y(part)
   part = reflect_xy(part)
   part = reflect_y(part)
   return part

def reflect_x(part):
   part = replace(part,'X','-X')
   return part

def reflect_y(part):
   part = replace(part,'Y','-Y')
   return part

def reflect_z(part):
   part = replace(part,'Z','-Z')
   return part

def reflect_xy(part):
   part = replace(part,'X','temp')
   part = replace(part,'Y','X')
   part = replace(part,'temp','Y')
   return part

def reflect_xz(part):
   part = replace(part,'X','temp')
   part = replace(part,'Z','X')
   part = replace(part,'temp','Z')
   return part

def reflect_yz(part):
   part = replace(part,'Y','temp')
   part = replace(part,'Z','Y')
   part = replace(part,'temp','Z')
   return part

def scale_x(part, x0, sx):
   part = replace(part,'X','(x0 + (X-x0)/sx)')
   part = replace(part,'x0',str(x0))
   part = replace(part,'sx',str(sx))
   return part

def scale_y(part, y0, sy):
   part = replace(part,'Y','(y0 + (Y-y0)/sy)')
   part = replace(part,'y0',str(y0))
   part = replace(part,'sy',str(sy))
   return part

def scale_z(part, z0, sz):
   part = replace(part,'Z','(z0 + (Z-z0)/sz)')
   part = replace(part,'z0',str(z0))
   part = replace(part,'sz',str(sz))
   return part

def scale_xy(part, x0, y0, sxy):
   part = replace(part,'X','(x0 + (X-x0)/sxy)')
   part = replace(part,'Y','(y0 + (Y-y0)/sxy)')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'sxy',str(sxy))
   return part

def scale_xyz(part, x0, y0, z0, sxyz):
   part = replace(part,'X','(x0 + (X-x0)/sxyz)')
   part = replace(part,'Y','(y0 + (Y-y0)/sxyz)')
   part = replace(part,'Z','(z0 + (Z-z0)/sxyz)')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'sxyz',str(sxyz))
   return part

def coscale_x_y(part, x0, y0, y1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
   part = replace(part,'X','(x0 + (X-x0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Y-y0)/(y1-y0))))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

def coscale_x_z(part, x0, z0, z1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
```

```
   part = replace(part,'X','(x0 + (X-x0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0))))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

def coscale_xy_z(part, x0, y0, z0, z1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
   part = replace(part,'X','(x0 + (X-x0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0))))')
   part = replace(part,'Y','(y0 + (Y-y0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0))))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

def taper_x_y(part, x0, y0, y1, s0, s1):
   part = replace(part,'X','(x0 + (X-x0)*(y1-y0)/(s1*(Y-y0) + s0*(y1-Y)))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'s0',str(s0))
   part = replace(part,'s1',str(s1))
   return part

def taper_x_z(part, x0, z0, z1, s0, s1):
   part = replace(part,'X','(x0 + (X-x0)*(z1-z0)/(s1*(Z-z0) + s0*(z1-Z)))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'s0',str(s0))
   part = replace(part,'s1',str(s1))
   return part

def taper_xy_z(part, x0, y0, z0, z1, s0, s1):
   part = replace(part,'X','(x0 + (X-x0)*(z1-z0)/(s1*(Z-z0) + s0*(z1-Z)))')
   part = replace(part,'Y','(y0 + (Y-y0)*(z1-z0)/(s1*(Z-z0) + s0*(z1-Z)))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'s0',str(s0))
   part = replace(part,'s1',str(s1))
   return part

def shear_x_y(part, y0, y1, dx0, dx1):
   part = replace(part,'X','(X - dx0 - (dx1-dx0)*(Y-y0)/(y1-y0))')
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'dx0',str(dx0))
   part = replace(part,'dx1',str(dx1))
   return part

def shear_x_z(part, z0, z1, dx0, dx1):
   part = replace(part,'X','(X - dx0 - (dx1-dx0)*(Z-z0)/(z1-z0))')
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'dx0',str(dx0))
   part = replace(part,'dx1',str(dx1))
   return part

def coshear_x_z(part, z0, z1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
   part = replace(part,'X','(X - offset - amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0)))')
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

#
# PCB classes and definitions
#

cad.labels = []

class PCB:
   def __init__(self,x0,y0,width,height,mask):
      self.board = "False"
      self.interior = rectangle(x0,x0+width,y0,y0+height)
      self.exterior = subtract("True",rectangle(x0,x0+width,y0,y0+height))
      self.mask = "False"
   def add(self,part):
      self.board = add(self.board,part)
      self.mask = add(self.mask,move(part,-mask,mask))
      self.mask = add(self.mask,move(part,-mask,-mask))
      self.mask = add(self.mask,move(part,mask,mask))
      self.mask = add(self.mask,move(part,mask,-mask))
      return self
```

```
class part:
   def add(self,pcb,x,y,z=0,angle=0):
      self.x = x
      self.y = y
      self.z = z
      self.angle = angle
      if (angle == 90):
         self.shape = rotate_90(self.shape)
      elif (angle == 180):
         self.shape = rotate_180(self.shape)
      elif ((angle == 270) | (angle == -90)):
         self.shape = rotate_270(self.shape)
      angle = pi*angle/180
      self.shape = translate(self.shape,x,y,z)
      for i in range(len(self.pad)):
         xnew = cos(angle)*self.pad[i].x + sin(angle)*self.pad[i].y
         ynew = -sin(angle)*self.pad[i].x + cos(angle)*self.pad[i].y
           self.pad[i].x = x + xnew
           self.pad[i].y = y + ynew
         self.pad[i].z += z
      cad.labels.append(cad_text(x,y,z,self.value,size=14))
      for i in range(len(self.labels)):
         xnew = cos(angle)*self.labels[i].x + sin(angle)*self.labels[i].y
         ynew = -sin(angle)*self.labels[i].x + cos(angle)*self.labels[i].y
           self.labels[i].x = x + xnew
           self.labels[i].y = y + ynew
         self.labels[i].z += z
           cad.labels.append(self.labels[i])
      pcb = pcb.add(self.shape)
      return pcb

def wire(pcb,width,*points):
   for i in range(1,len(points)):
      x0 = points[i-1].x
      y0 = points[i-1].y
      z0 = points[i-1].z
      x1 = points[i].x
      y1 = points[i].y
      z1 = points[i].z
      if (x0 < x1):
         pcb.board = add(pcb.board,cube(x0-width/2,x1+width/2,y0-width/2,y0+width/2,z0,z0))
      elif (x1 < x0):
         pcb.board = add(pcb.board,cube(x1-width/2,x0+width/2,y0-width/2,y0+width/2,z0,z0))
      if (y0 < y1):
         pcb.board = add(pcb.board,cube(x1-width/2,x1+width/2,y0-width/2,y1+width/2,z0,z0))
      elif (y1 < y0):
         pcb.board = add(pcb.board,cube(x1-width/2,x1+width/2,y1-width/2,y0+width/2,z0,z0))
   return pcb

#
# PCB library
#

#
# discretes
#

pad_0402 = cube(-.0175,.0175,-.014,.014,0,0)

class R_0402(part):
   #
   # 0402 resistor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_0402,-.0265,0,0)
      self.pad.append(point(-.0265,0,0))
      self.shape = add(self.shape,translate(pad_0402,.0265,0,0))
      self.pad.append(point(.0265,0,0))

pad_1206 = cube(-.032,.032,-.034,.034,0,0)

class R_1206(part):
   #
   # 1206 resistor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.06,0,0))
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.06,0,0))

class C_1206(part):
   #
   # 1206 capacitor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.06,0,0))
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.06,0,0))
```

```
pad_1210 = cube(-.032,.032,-.048,.048,0,0)

class L_1210(part):
   #
   # 1210 inductor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_1210,-.06,0,0)
      self.pad.append(point(-.06,0,0))
      self.shape = add(self.shape,translate(pad_1210,.06,0,0))
      self.pad.append(point(.06,0,0))

pad_choke = cube(-.06,.06,-.06,.06,0,0)

class choke(part):
   #
   # Panasonic ELLCTV
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_choke,-.177,-.177,0)
      self.pad.append(point(-.177,-.177,0))
      self.shape = add(self.shape,translate(pad_choke,.177,.177,0))
      self.pad.append(point(.177,.177,0))

#
# connectors
#

pad_MTA = cube(-.021,.021,-.041,.041,0,0)
pad_MTA_solder = cube(-.071,.071,-.041,.041,0,0)

class MTA_2(part):
   #
   # AMP 1445121-2
   # MTA .050 SMT 2-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1
      #
      self.shape = translate(pad_MTA,-.025,-.1,0)
      self.pad.append(point(-.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,.1,0))
      self.pad.append(point(.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'2',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.187,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.187,0,0))

class MTA_power(part):
   #
   # AMP 1445121-2
   # MTA .050 SMT 2-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: Gnd
      #
      self.shape = translate(pad_MTA,-.025,-.1,0)
      self.pad.append(point(-.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: Vcc
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,.1,0))
      self.pad.append(point(.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Vcc',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.187,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.187,0,0))

class MTA_i0(part):
   #
   # AMP 1445121-3
   # MTA .050 SMT 3-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
```

```
      # pin 1: GND
      #
      self.shape = translate(pad_MTA,.05,.1,0)
      self.pad.append(point(.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1\nGND',14))
      #
      # pin 2: power
      #
      self.shape = add(self.shape,translate(pad_MTA,-.05,.1,0))
      self.pad.append(point(-.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V',14))
      #
      # pin 3: data
      #
      self.shape = add(self.shape,translate(pad_MTA,0,-.1,0))
      self.pad.append(point(0,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'data',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.212,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.212,0,0))

class MTA_serial(part):
   #
   # AMP 1445121-4
   # MTA .050 SMT 4-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: Gnd
      #
      self.shape = translate(pad_MTA,-.075,-.1,0)
      self.pad.append(point(-.075,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: Tx
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,-.1,0))
      self.pad.append(point(.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Tx',14))
      #
      # pin 3: Rx
      #
      self.shape = add(self.shape,translate(pad_MTA,.075,.1,0))
      self.pad.append(point(.075,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Rx',14))
      #
      # pin 4: DTR
      #
      self.shape = add(self.shape,translate(pad_MTA,-.025,.1,0))
      self.pad.append(point(-.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DTR',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.237,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.237,0,0))

class MTA_ICP(part):
   #
   # AMP 1445121-5
   # MTA .050 SMT 4-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: MISO
      #
      self.shape = translate(pad_MTA,-.1,-.1,0)
      self.pad.append(point(-.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'MISO',14))
      #
      # pin 2: GND
      #
      self.shape = add(self.shape,translate(pad_MTA,0,-.1,0))
      self.pad.append(point(0,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))
      #
      # pin 3: MOSI
      #
      self.shape = add(self.shape,translate(pad_MTA,.1,-.1,0))
      self.pad.append(point(.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'MOSI',14))
      #
      # pin 4: -RESET
      #
      self.shape = add(self.shape,translate(pad_MTA,.05,.1,0))
      self.pad.append(point(.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'-RESET',14))
      #
      # pin 5: SCK
      #
      self.shape = add(self.shape,translate(pad_MTA,-.05,.1,0))
      self.pad.append(point(-.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'SCK',14))
```

```
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.262,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.262,0,0))

class power_65mm(part):
   #
   # CUI PJ1-023-SMT
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: power
      #
      self.shape = cube(.433,.512,-.047,.047,0,0)
      self.pad.append(point(.467,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'P',14))
      #
      # pin 2: ground
      #
      self.shape = add(self.shape,cube(.285,.423,-.189,-.098,0,0))
      self.pad.append(point(.354,-.144,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 3: contact
      #
      self.shape = add(self.shape,cube(.325,.463,.098,.189,0,0))
      self.pad.append(point(.394,.144,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,cube(.108,.246,-.169,-.110,0,0))
      self.shape = add(self.shape,cube(.069,.207,.110,.169,0,0))

pad_stereo_2_5mm = cube(-.03,.03,-.05,.05,0,0)

class stereo_2_5mm(part):
   #
   # CUI SJ1-2533-SMT
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: base
      #
      self.shape = translate(pad_stereo_2_5mm,-.130,-.16,0)
      self.pad.append(point(-.130,-.149,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'base',14))
      #
      # pin 2: tip
      #
      self.shape = add(self.shape,translate(pad_stereo_2_5mm,.197,.15,0))
      self.pad.append(point(.197,.141,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'tip',14))
      #
      # pin 3: middle
      #
      self.shape = add(self.shape,translate(pad_stereo_2_5mm,-.012,-.16,0))
      self.pad.append(point(-.012,-.149,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'middle',14))

pad_Molex = cube(-.0155,.0155,-.0265,.0265,0,0)
pad_Molex_solder = cube(-.055,.055,-.065,.065,0,0)

class Molex_serial(part):
   #
   # Molex 53261-0471
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: Rx
      #
      self.shape = translate(pad_Molex,-.075,.064,0)
      self.pad.append(point(-.075,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Rx',14))
      #
      # pin 2: Tx
      #
      self.shape = add(self.shape,translate(pad_Molex,-.025,.064,0))
      self.pad.append(point(-.025,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Tx',14))
      #
      # pin 3: DTR
      #
      self.shape = add(self.shape,translate(pad_Molex,.025,.064,0))
      self.pad.append(point(.025,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DTR',14))
      #
      # pin 4: GND
      #
      self.shape = add(self.shape,translate(pad_Molex,.075,.064,0))
      self.pad.append(point(.075,.064,0))
```

```
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_Molex_solder,-.16,-.065,0))
      self.shape = add(self.shape,translate(pad_Molex_solder,.16,-.065,0))

#
# switches
#

pad_button_6mm = cube(-.04,.04,-.03,.03,0,0)

class button_6mm(part):
   #
   # Omron 6mm pushbutton
   # B3SN-3112P
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # left 1
      #
      self.shape = translate(pad_button_6mm,-.125,.08,0)
      self.pad.append(point(-.125,.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'L1',14))
      #
      # right 1
      #
      self.shape = add(self.shape,translate(pad_button_6mm,-.125,-.08,0))
      self.pad.append(point(-.125,-.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'R1',14))
      #
      # right 2
      #
      self.shape = add(self.shape,translate(pad_button_6mm,.125,-.08,0))
      self.pad.append(point(.125,-.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'R2',14))
      #
      # left 2
      #
      self.shape = add(self.shape,translate(pad_button_6mm,.125,.08,0))
      self.pad.append(point(.125,.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'L2',14))

#
# crystals and resonators
#

pad_XTAL = cube(-.016,.016,-.077,.077,0,0)

class XTAL(part):
   #
   # Panasonic EFOBM series
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # left
      #
      self.shape = translate(pad_XTAL,-.053,0,0)
      self.pad.append(point(-.053,0,0))
      #
      # ground
      #
      self.shape = add(self.shape,translate(pad_XTAL,0,0,0))
      self.pad.append(point(0,0,0))
      #
      # right
      #
      self.shape = add(self.shape,translate(pad_XTAL,.053,0,0))
      self.pad.append(point(.053,0,0))

#
# diodes, transistors, regulators
#

class D_1206(part):
   #
   # 1206 diode
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # anode
      #
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A',14))
      #
      # cathode
      #
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))
```

```
class LED_1206(part):
   #
   # 1206 LED
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # anode
      #
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A',14))
      #
      # cathode
      #
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))

class phototransistor_1206(part):
   #
   # 1206 phototransistor
   # OPTEK 520,521
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # collector
      #
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))
      #
      # emitter
      #
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'E',14))

pad_SOD_123 = cube(-.02,.02,-.024,.024,0,0)

class D_SOD_123(part):
   #
   # SOD-123 diode
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # anode
      #
      self.shape = translate(pad_SOD_123,-.07,0,0)
      self.pad.append(point(-.07,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A',14))
      #
      # cathode
      #
      self.shape = add(self.shape,translate(pad_SOD_123,.07,0,0))
      self.pad.append(point(.07,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))

pad_SOT23 = cube(-.015,.015,-.02,.02,0,0)

class NMOSFET_SOT23(part):
   #
   # Fairchild NDS355AN
   #
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: gate
      #
      self.shape = translate(pad_SOT23,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 2: source
      #
      self.shape = add(self.shape,translate(pad_SOT23,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'S',14))
      #
      # pin 3: drain
      #
      self.shape = add(self.shape,translate(pad_SOT23,0,.045,0))
      self.pad.append(point(0,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'D',14))

class PMOSFET_SOT23(part):
   #
```

```
   # Fairchild NDS356AP
   #
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: gate
      #
      self.shape = translate(pad_SOT23,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 2: source
      #
      self.shape = add(self.shape,translate(pad_SOT23,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'S',14))
      #
      # pin 3: drain
      #
      self.shape = add(self.shape,translate(pad_SOT23,0,.045,0))
      self.pad.append(point(0,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'D',14))

class regulator_SOT23(part):
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: output
      #
      self.shape = translate(pad_SOT23,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: input
      #
      self.shape = add(self.shape,translate(pad_SOT23,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'in',14))
      #
      # pin 3: ground
      #
      self.shape = add(self.shape,translate(pad_SOT23,0,.045,0))
      self.pad.append(point(0,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'gnd',14))

pad_SOT223 = cube(-.02,.02,-.03,.03,0,0)
pad_SOT223_ground = cube(-.065,.065,-.03,.03,0,0)

class regulator_SOT223(part):
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: input
      #
      self.shape = translate(pad_SOT223,-.09,-.12,0)
      self.pad.append(point(-.09,-.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1I',14))
      #
      # pin 2: ground
      #
      self.shape = add(self.shape,translate(pad_SOT223,0,-.12,0))
      self.pad.append(point(0,-.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 3: output
      #
      self.shape = add(self.shape,translate(pad_SOT223,.09,-.12,0))
      self.pad.append(point(.09,-.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'O',14))
      #
      # pin 4: ground
      #
      self.shape = add(self.shape,translate(pad_SOT223_ground,0,.12,0))
      self.pad.append(point(0,.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))

pad_SM8 = cube(-.035,.035,-.016,.016,0,0)

class H_bridge_SM8(part):
   #
   # Zetex ZXMHC3A01T8
   #
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
```

```
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      d = .13
      #
      # pin 1: G3 (right N gate)
      #
      self.shape = translate(pad_SM8,-d,.09,0)
      self.pad.append(point(-d,.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GRN',14))
      #
      # pin 2: S2 S3 (N source)
      #
      self.shape = add(self.shape,translate(pad_SM8,-d,.03,0))
      self.pad.append(point(-d,.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'SN',14))
      #
      # pin 3: G2 (left N gate)
      #
      self.shape = add(self.shape,translate(pad_SM8,-d,-.03,0))
      self.pad.append(point(-d,-.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GLN',14))
      #
      # pin 4: G1 (left P gate)
      #
      self.shape = add(self.shape,translate(pad_SM8,-d,-.09,0))
      self.pad.append(point(-d,-.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GLP',14))
      #
      # pin 5: D1 D2 (left drain)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,-.09,0))
      self.pad.append(point(d,-.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DL',14))
      #
      # pin 6: S1 S4 (P source)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,-.03,0))
      self.pad.append(point(d,-.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'SP',14))
      #
      # pin 7: D3 D4 (right drain)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,.03,0))
      self.pad.append(point(d,.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DR',14))
      #
      # pin 8: G4 (right N gate)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,.09,0))
      self.pad.append(point(d,.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GRP',14))

#
# ICs
#

pad_SOT23_5 = cube(-.01,.01,-.02,.02,0,0)

class op_amp_SOT23_5(part):
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: output
      #
      self.shape = translate(pad_SOT23_5,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'O',14))
      #
      # pin 2: V-
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,0,-.045,0))
      self.pad.append(point(0,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V-',14))
      #
      # pin 3: I+
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'I+',14))
      #
      # pin 4: I-
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,.0375,.045,0))
      self.pad.append(point(.0375,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'I-',14))
      #
      # pin 5: V+
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,-.0375,.045,0))
      self.pad.append(point(-.0375,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V+',14))

pad_SOICN = cube(-.035,.035,-.015,.015,0,0)

class op_amp_SOICN(part):
```

```
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: A out
      #
      self.shape = translate(pad_SOICN,-.12,.075,0)
      self.pad.append(point(-.12,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1 Ao',14))
      #
      # pin 2: A-
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.025,0))
      self.pad.append(point(-.12,.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A-',14))
      #
      # pin 3: A+
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.025,0))
      self.pad.append(point(-.12,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A+',14))
      #
      # pin 4: V-
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.075,0))
      self.pad.append(point(-.12,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V-',14))
      #
      # pin 5: B+
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.075,0))
      self.pad.append(point(.12,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'B+',14))
      #
      # pin 6: B-
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.025,0))
      self.pad.append(point(.12,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'B-',14))
      #
      # pin 7: B out
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.025,0))
      self.pad.append(point(.12,.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Bo',14))
      #
      # pin 8: V+
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.075,0))
      self.pad.append(point(.12,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V+',14))

pad_SOIC = cube(-.043,.043,-.015,.015,0,0)

class ATtiny45_SOIC(part):
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: PB5/dW/ADC0/-RESET/PCINT5
      #
      self.shape = translate(pad_SOIC,-.14,.075,0)
      self.pad.append(point(-.14,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: PB3/ADC3/-OC1B/CLKI/XTAL1/PCINT3
      #
      self.shape = add(self.shape,translate(pad_SOIC,-.14,.025,0))
      self.pad.append(point(-.14,.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB3',14))
      #
      # pin 3: PB4/ADC2/OC1B/CLKO/XTAL2/PCINT4
      #
      self.shape = add(self.shape,translate(pad_SOIC,-.14,-.025,0))
      self.pad.append(point(-.14,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB4',14))
      #
      # pin 4: GND
      #
      self.shape = add(self.shape,translate(pad_SOIC,-.14,-.075,0))
      self.pad.append(point(-.14,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))
      #
      # pin 5: PB0/MOSI/DI/SDA/AIN0/OC0A/-OC1A/AREF/PCINT0
      #
      self.shape = add(self.shape,translate(pad_SOIC,.14,-.075,0))
      self.pad.append(point(.14,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 6: PB1/MISO/DO/AIN1/OC0B/OC1A/PCINT1
      #
      self.shape = add(self.shape,translate(pad_SOIC,.14,-.025,0))
      self.pad.append(point(.14,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB1',14))
      #
      # pin 7: PB2/SCK/USCK/SCL/ADC1/T0/INT0/PCINT2
      #
      self.shape = add(self.shape,translate(pad_SOIC,.14,.025,0))
      self.pad.append(point(.14,.025,0))
```

```
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB2',14))
      #
      # pin 8: VCC
      #
      self.shape = add(self.shape,translate(pad_SOIC,.14,.075,0))
      self.pad.append(point(.14,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'VCC',14))

class ATtiny44_SOICN(part):
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: VCC
      #
      self.shape = translate(pad_SOICN,-.12,.15,0)
      self.pad.append(point(-.12,.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: PB0/XTAL1/PCINT8
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 3: PB1/XTAL2/PCINT9
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.050,0))
      self.pad.append(point(-.12,.05,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB1',14))
      #
      # pin 4: PB3/dW/-RESET/PCINT11
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,0,0))
      self.pad.append(point(-.12,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB3',14))
      #
      # pin 5: PB2/CKOUT/OC0A/INT0/PCINT10
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.05,0))
      self.pad.append(point(-.12,-.05,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB2',14))
      #
      # pin 6: PA7/ADC7/OC0B/ICP/PCINT7
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.1,0))
      self.pad.append(point(-.12,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA7',14))
      #
      # pin 7: PA6/ADC6/MOSI/SDA/OC1A/PCINT6
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.15,0))
      self.pad.append(point(-.12,-.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA6',14))
      #
      # pin 8: PA5/ADC5/DO/MISO/OC1B/PCINT5
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.15,0))
      self.pad.append(point(.12,-.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA5',14))
      #
      # pin 9: PA4/ADC4/USCK/SCL/T1/PCINT4
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.1,0))
      self.pad.append(point(.12,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA4',14))
      #
      # pin 10: PA3/ADC3/T0/PCINT3
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.05,0))
      self.pad.append(point(.12,-.05,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA3',14))
      #
      # pin 11: PA2/ADC2/AIN1/PCINT2
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,0,0))
      self.pad.append(point(.12,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA2',14))
      #
      # pin 12: PA1/ADC1/AIN0/PCINT1
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.050,0))
      self.pad.append(point(.12,.05,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA1',14))
      #
      # pin 13: PA0/ADC0/AREF/PCINT0
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.1,0))
      self.pad.append(point(.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA0',14))
      #
      # pin 14: GND
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.15,0))
      self.pad.append(point(.12,.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))

pad_TQFP = cube(-.043,.043,-.015,.015,0,0)

class ATmega88_TQFP(part):
```

```
def __init__(self,value=''):
   self.value = value
   self.pad = [point(0,0,0)]
   self.labels = []

   #
   # pin 1: PD3/PCINT19/OC2B/INT1
   #
   self.shape = translate(pad_SOICN,-.12,.15,0)
   self.pad.append(point(-.12,.15,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
   #
   # pin 2: PD4/PCINT20/XCK/T0
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 3: GND
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 4: VCC
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 5: GND
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 6: VCC
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 7: PB6/PCINT6/XTAL1/TOSC1
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 8: PB7/PCINT7/XTAL2/TOSC2
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

   #
   # pin 9: PD5/PCINT21/OC0B/T1
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 10: PD6/PCINT22/OC0A/AIN0
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 11: PD7/PCINT23/AIN1
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 12: PB0/PCINT0/CLKO/ICP1
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 13: PB1/PCINT1/OC1A
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 14: PB2/PCINT2/-SS/OC1B
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 15: PB3/PCINT3/OC2A/MOSI
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
   self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
   #
   # pin 16: PB4/PCINT4/MISO
   #
   self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
   self.pad.append(point(-.12,.1,0))
```

```
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

        #
        # pin 17: PB5/SCK/PCINT5
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 18: AVCC
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 19: ADC6
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 20: AREF
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 21: GND
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 22: ADC7
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 23: PC0/ADC0/PCINT8
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 24: PC1/ADC1/PCINT9
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

        #
        # pin 25: PC2/ADC2/PCINT10
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 26: PC2/ADC3/PCINT11
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 27: PC4/ADC4/SDA/PCINT12
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 28: PC5/ADC5/SCL/PCINT13
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 29: PC6/-RESET/PCINT14
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 30: PD0/RXD/PCINT16
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 31: PD1/TXD/PCINT17
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 32: PD2/INT0/PCINT18
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

#
```

```
# graphics
#

class CBA(part):
   def __init__(self,r=.02):
      self.value = ''
      self.pad = [point(0,0,0)]
      self.labels = []
      d = 3*r
      self.shape = circle(0,0,r)
      self.shape = add(self.shape,translate(circle(0,0,r),-d,d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),-d,0,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),-d,-d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),0,-d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),d,-d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),d,0,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),d,d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),0,d,0))

class hello(part):
   #
   # HELLO
   #
   def __init__(self,w,z):
      self.value = ''
      self.pad = [point(0,0,0)]
      self.labels = []
      # H
      self.shape = rectangle(-.01,.01,0,.1)
      self.shape = add(self.shape,rectangle(0,.05,.04,.06))
      self.shape = add(self.shape,rectangle(.04,.06,0,.1))
      # E
      self.shape = add(self.shape,rectangle(.08,.1,0,.1))
      self.shape = add(self.shape,rectangle(.1,.14,0,.02))
      self.shape = add(self.shape,rectangle(.1,.13,.04,.06))
      self.shape = add(self.shape,rectangle(.1,.14,.08,.1))
      # L
      self.shape = add(self.shape,rectangle(.16,.18,0,.1))
      self.shape = add(self.shape,rectangle(.18,.21,0,.02))
      # L
      self.shape = add(self.shape,rectangle(.23,.25,0,.1))
      self.shape = add(self.shape,rectangle(.25,.28,0,.02))
      # 0
      self.shape = add(self.shape,rectangle(.3,.32,0,.1))
      self.shape = add(self.shape,rectangle(.32,.345,0,.02))
      self.shape = add(self.shape,rectangle(.32,.345,.08,.1))
      self.shape = add(self.shape,rectangle(.345,.365,0,.1))

#
# define board
#

w = .016
width = 1.19
height = 1.09
x = 1
y = 1
z = -.005
d = .06
mask = .004

pcb = PCB(x,y,width,height,mask)

IC1 = ATtiny44_SOICN('IC1\nt44')
pcb = IC1.add(pcb,x+.49,y+.55,z)

J1 = MTA_ICP('J1')
pcb = J1.add(pcb,IC1.x,IC1.pad[7].y-.2,z,angle=180)

pcb = wire(pcb,w,
   IC1.pad[7],
   J1.pad[3])

pcb = wire(pcb,w,
   IC1.pad[4],
   J1.pad[4])

pcb = wire(pcb,w,
   IC1.pad[14],
   point(J1.pad[5].x,IC1.pad[10].y,z),
   J1.pad[2])

pcb = wire(pcb,w,
   IC1.pad[9],
   J1.pad[5])

pcb = wire(pcb,w,
   IC1.pad[8],
   J1.pad[1])

J2 = MTA_power('J2')
pcb = J2.add(pcb,IC1.x-.33,IC1.y,z,angle=-90)

IC2 = regulator_SOT23('IC2\n5V')
pcb = IC2.add(pcb,J2.x,J2.y+.38,z,angle=180)

pcb = wire(pcb,w,
   IC2.pad[2],
   J2.pad[2])

pcb = wire(pcb,w,
```

```
    IC2.pad[3],
    point(IC2.x,IC2.y+.12,z),
    IC1.pad[14])

C1 = C_1206('C1\n3.3uF')
pcb = C1.add(pcb,IC2.x+.13,IC2.y,z,angle=90)

pcb = wire(pcb,w,
    IC2.pad[1],
    C1.pad[1])

pcb = wire(pcb,w,
    C1.pad[1],
    IC1.pad[1])

pcb = wire(pcb,w,
    IC2.pad[3],
    C1.pad[2])

pcb = wire(pcb,w,
    J2.pad[1],
    C1.pad[2])

R1 = R_1206('R1\n10k')
pcb = R1.add(pcb,IC1.x,C1.y,z,angle=90)

pcb = wire(pcb,w,
    IC1.pad[4],
    R1.pad[2])

pcb = wire(pcb,w,
    C1.pad[1],
    R1.pad[1])

H1 = H_bridge_SM8('H1')
pcb = H1.add(pcb,IC1.x+.45,IC1.y-.02,z)

pcb = wire(pcb,w,
    IC1.pad[12],
    point(IC1.pad[12].x+.12,H1.pad[1].y,z),
    H1.pad[1])

pcb = wire(pcb,w,
    H1.pad[3],
    point(IC1.pad[11].x+.12,IC1.pad[11].y,z),
    IC1.pad[11])

J3 = MTA_2('J3')
pcb = J3.add(pcb,H1.x-.03,H1.y+.4,z,angle=0)

pcb = wire(pcb,w,
    H1.pad[7],
    point(H1.pad[7].x+.06,J3.pad[1].y-.03,z),
    J3.pad[1])

pcb = wire(pcb,w,
    H1.pad[5],
    point(H1.pad[5].x+.09,J3.pad[1].y+.03,z),
    point(J3.pad[2].x+.05,J3.pad[2].y,z),
    J3.pad[2])

T1 = NMOSFET_SOT23('T1\nN')
pcb = T1.add(pcb,H1.x-.2,H1.y+.23,z)

pcb = wire(pcb,w,
    IC1.pad[13],
    point(IC1.pad[13].x+.08,T1.pad[1].y,z),
    T1.pad[1])

pcb = wire(pcb,w,
    T1.pad[2],
    point(H1.x-.06,H1.pad[2].y,z),
    H1.pad[2])

pcb = wire(pcb,w,
    IC1.pad[14],
    point(IC1.pad[14].x,T1.y,z),
    T1.pad[2])

R3 = R_1206('R3\n1k')
pcb = R3.add(pcb,H1.x+.06,H1.y+.17,z)

pcb = wire(pcb,w,
    T1.pad[3],
    point(T1.x+.08,T1.y,z),
    R3.pad[2])

pcb = wire(pcb,w,
    H1.pad[8],
    R3.pad[2])

R4 = R_1206('R4\n1k')
pcb = R4.add(pcb,H1.x,H1.y-.19,z)

pcb = wire(pcb,w,
    R4.pad[1],
    H1.pad[4])

pcb = wire(pcb,w,
    H1.pad[6],
    R4.pad[2])
```

```
pcb = wire(pcb,w,
   R3.pad[1],
   R4.pad[2])

pcb = wire(pcb,w,
   J2.pad[2],
   point(J2.pad[2].x,J1.pad[4].y-.07,z),
   R4.pad[2])

T2 = NMOSFET_SOT23('T2\nN')
pcb = T2.add(pcb,R4.x-.04,R4.y-.15,z,angle=0)

pcb = wire(pcb,w,
   IC1.pad[10],
   point(IC1.pad[10].x+.08,J1.pad[1].y-.025,z),
   T2.pad[1])

pcb = wire(pcb,w,
   T2.pad[2],
   point(H1.x,H1.pad[2].y,z),
   H1.pad[2])

pcb = wire(pcb,w,
   R4.pad[1],
   T2.pad[3])

G1 = hello(w,z)
pcb = G1.add(pcb,x+width-.03,y+.03,z,angle=-90)

cad.function = add(pcb.board,pcb.exterior)

#
# uncomment to export traces
#

#cad.function = pcb.board

#
# uncomment to export exterior
#

#cad.function = pcb.exterior

#
# uncomment to export solder mask
#

#cad.function = pcb.mask

#
# uncomment to mill out board
#
# use 1/32 end-mill
# set tool diameter = .0312
# set xy, z speed = .5
# set # contours = 1
#

#cad.function = pcb.interior
#z = -.065

#
# define limits and parameters
#

cad.xmin = x-.1 # min x to render
cad.xmax = x+width+.1 # max x to render
cad.ymin = y-.1 # min y to render
cad.ymax = y+width+.1 # max y to render
cad.zmin = z
cad.zmax = .050
dpi = 200 # low resolution for previewing
#dpi = 500 # high resolution for machining
nxy = int(dpi*(cad.xmax-cad.xmin))
cad.nx = nxy # x points to render
cad.ny = nxy # y points to render
cad.nz = 1
cad.view('xy') # 2D view
```

```
#
# hello.video.44.cad
#
# NTSC video hello-world
#
# Neil Gershenfeld
# CBA MIT 11/17/07
#
# (c) Massachusetts Institute of Technology 2007
# Permission granted for experimental and personal use;
# license for commercial sale available from MIT.
#

#
# define shapes and transformation
#
# circle(x0, y0, r)
# cylinder(x0, y0, z0, z1, r)
# cone(x0, y0, z0, z1, r0)
# sphere(x0, y0, z0, r)
# torus(x0, y0, z0, r0, r1)
# rectangle(x0, x1, y0, y1)
# cube(x0, x1, y0, y1, z0, z1)
# triangle(x0, y0, x1, y1, x2, y2) (points in clockwise order)
# pyramid(x0, x1, y0, y1, z0, z1)
# function(Z_of_XY)
# functions(upper_Z_of_XY,lower_Z_of_XY)
# add(part1, part2)
# subtract(part1, part2)
# intersect(part1, part2)
# move(part,dx,dy)
# translate(part,dx,dy,dz)
# rotate(part, angle)
# rotate_x(part, angle)
# rotate_y(part, angle)
# rotate_z(part, angle)
# rotate_90(part)
# rotate_180(part)
# rotate_270(part)
# reflect_x(part)
# reflect_y(part)
# reflect_z(part)
# reflect_xy(part)
# reflect_xz(part)
# reflect_yz(part)
# scale_x(part, x0, sx)
# scale_y(part, y0, sy)
# scale_z(part, z0, sz)
# scale_xy(part, x0, y0, sxy)
# scale_xyz(part, x0, y0, z0, sxyz)
# coscale_x_y(part, x0, y0, y1, angle0, angle1, amplitude, offset)
# coscale_x_z(part, x0, z0, z1, angle0, angle1, amplitude, offset)
# coscale_xy_z(part, x0, y0, z0, z1, angle0, angle1, amplitude, offset)
# taper_x_y(part, x0, y0, y1, s0, s1)
# taper_x_z(part, x0, z0, z1, s0, s1)
# taper_xy_z(part, x0, y0, z0, z1, s0, s1)
# shear_x_y(part, y0, y1, dx0, dx1)
# shear_x_z(part, z0, z1, dx0, dx1)
# (more to come)

def circle(x0, y0, r):
   part = "(((X-x0)**2 + (Y-y0)**2) <= r**2)"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'r',str(r))
   return part

def cylinder(x0, y0, z0, z1, r):
   part = "(((X-x0)**2 + (Y-y0)**2 <= r**2) & (Z >= z0) & (Z <= z1))"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'r',str(r))
   return part

def cone(x0, y0, z0, z1, r0):
   part = cylinder(x0, y0, z0, z1, r0)
   part = taper_xy_z(part, x0, y0, z0, z1, 1.0, 0.0)
   return part

def sphere(x0, y0, z0, r):
   part = "(((X-x0)**2 + (Y-y0)**2 + (Z-z0)**2) <= r**2)"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'r',str(r))
   return part

def torus(x0, y0, z0, r0, r1):
   part = "(((r0 - sqrt((X-x0)**2 + (Y-y0)**2))**2 + (Z-z0)**2) <= r1**2)"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'r0',str(r0))
   part = replace(part,'r1',str(r1))
   return part

def rectangle(x0, x1, y0, y1):
   part = "((X >= x0) & (X <= x1) & (Y >= y0) & (Y <= y1))"
   part = replace(part,'x0',str(x0))
```

```
   part = replace(part,'x1',str(x1))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   return part

def cube(x0, x1, y0, y1, z0, z1):
   part = "((X >= x0) & (X <= x1) & (Y >= y0) & (Y <= y1) & (Z >= z0) & (Z <= z1))"
   part = replace(part,'x0',str(x0))
   part = replace(part,'x1',str(x1))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   return part

def triangle(x0, y0, x1, y1, x2, y2): # points in clockwise order
   part = "((((y1-y0)*(X-x0)-(x1-x0)*(Y-y0)) >= 0) & (((y2-y1)*(X-x1)-(x2-x1)*(Y-y1)) >= 0) & (((y0-y2)*(X-x2)-(x0-x2)*(Y-y2)) >= 0))"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'x1',str(x1))
   part = replace(part,'y1',str(y1))
   part = replace(part,'x2',str(x2))
   part = replace(part,'y2',str(y2))
   return part

def pyramid(x0, x1, y0, y1, z0, z1):
   part = cube(x0, x1, y0, y1, z0, z1)
   part = taper_xy_z(part, (x0+x1)/2., (y0+y1)/2., z0, z1, 1.0, 0.0)
   return part

def function(Z_of_XY):
   part = '(Z <= '+Z_of_XY+')'
   return part

def functions(upper_Z_of_XY,lower_Z_of_XY):
   part = '(Z <= '+upper_Z_of_XY+') & (Z >= '+lower_Z_of_XY+')'
   return part

def add(part1, part2):
   part = "part1 | part2"
   part = replace(part,'part1',part1)
   part = replace(part,'part2',part2)
   return part

def subtract(part1, part2):
   part = "(part1) & ~(part2)"
   part = replace(part,'part1',part1)
   part = replace(part,'part2',part2)
   return part

def intersect(part1, part2):
   part = "(part1) & (part2)"
   part = replace(part,'part1',part1)
   part = replace(part,'part2',part2)
   return part

def move(part,dx,dy):
   part = replace(part,'X','(X-'+str(dx)+')')
   part = replace(part,'Y','(Y-'+str(dy)+')')
   return part

def translate(part,dx,dy,dz):
   part = replace(part,'X','(X-'+str(dx)+')')
   part = replace(part,'Y','(Y-'+str(dy)+')')
   part = replace(part,'Z','(Z-'+str(dz)+')')
   return part

def rotate(part, angle):
   angle = angle*pi/180
   part = replace(part,'X','(cos(angle)*X+sin(angle)*y)')
   part = replace(part,'Y','(-sin(angle)*X+cos(angle)*y)')
   part = replace(part,'y','Y')
   part = replace(part,'angle',str(angle))
   return part

def rotate_x(part, angle):
   angle = angle*pi/180
   part = replace(part,'Y','(cos(angle)*Y+sin(angle)*z)')
   part = replace(part,'Z','(-sin(angle)*Y+cos(angle)*z)')
   part = replace(part,'z','Z')
   part = replace(part,'angle',str(angle))
   return part

def rotate_y(part, angle):
   angle = angle*pi/180
   part = replace(part,'X','(cos(angle)*X+sin(angle)*z)')
   part = replace(part,'Z','(-sin(angle)*X+cos(angle)*z)')
   part = replace(part,'z','Z')
   part = replace(part,'angle',str(angle))
   return part

def rotate_z(part, angle):
   angle = angle*pi/180
   part = replace(part,'X','(cos(angle)*X+sin(angle)*y)')
   part = replace(part,'Y','(-sin(angle)*X+cos(angle)*y)')
   part = replace(part,'y','Y')
   part = replace(part,'angle',str(angle))
   return part

def rotate_90(part):
   part = reflect_xy(part)
```

```
   part = reflect_y(part)
   return part

def rotate_180(part):
   part = reflect_xy(part)
   part = reflect_y(part)
   part = reflect_xy(part)
   part = reflect_y(part)
   return part

def rotate_270(part):
   part = reflect_xy(part)
   part = reflect_y(part)
   part = reflect_xy(part)
   part = reflect_y(part)
   part = reflect_xy(part)
   part = reflect_y(part)
   return part

def reflect_x(part):
   part = replace(part,'X','-X')
   return part

def reflect_y(part):
   part = replace(part,'Y','-Y')
   return part

def reflect_z(part):
   part = replace(part,'Z','-Z')
   return part

def reflect_xy(part):
   part = replace(part,'X','temp')
   part = replace(part,'Y','X')
   part = replace(part,'temp','Y')
   return part

def reflect_xz(part):
   part = replace(part,'X','temp')
   part = replace(part,'Z','X')
   part = replace(part,'temp','Z')
   return part

def reflect_yz(part):
   part = replace(part,'Y','temp')
   part = replace(part,'Z','Y')
   part = replace(part,'temp','Z')
   return part

def scale_x(part, x0, sx):
   part = replace(part,'X','(x0 + (X-x0)/sx)')
   part = replace(part,'x0',str(x0))
   part = replace(part,'sx',str(sx))
   return part

def scale_y(part, y0, sy):
   part = replace(part,'Y','(y0 + (Y-y0)/sy)')
   part = replace(part,'y0',str(y0))
   part = replace(part,'sy',str(sy))
   return part

def scale_z(part, z0, sz):
   part = replace(part,'Z','(z0 + (Z-z0)/sz)')
   part = replace(part,'z0',str(z0))
   part = replace(part,'sz',str(sz))
   return part

def scale_xy(part, x0, y0, sxy):
   part = replace(part,'X','(x0 + (X-x0)/sxy)')
   part = replace(part,'Y','(y0 + (Y-y0)/sxy)')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'sxy',str(sxy))
   return part

def scale_xyz(part, x0, y0, z0, sxyz):
   part = replace(part,'X','(x0 + (X-x0)/sxyz)')
   part = replace(part,'Y','(y0 + (Y-y0)/sxyz)')
   part = replace(part,'Z','(z0 + (Z-z0)/sxyz)')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'sxyz',str(sxyz))
   return part

def coscale_x_y(part, x0, y0, y1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
   part = replace(part,'X','(x0 + (X-x0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Y-y0)/(y1-y0))))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

def coscale_x_z(part, x0, z0, z1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
```

```
   part = replace(part,'X','(x0 + (X-x0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0))))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

def coscale_xy_z(part, x0, y0, z0, z1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
   part = replace(part,'X','(x0 + (X-x0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0))))')
   part = replace(part,'Y','(y0 + (Y-y0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0))))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

def taper_x_y(part, x0, y0, y1, s0, s1):
   part = replace(part,'X','(x0 + (X-x0)*(y1-y0)/(s1*(Y-y0) + s0*(y1-Y)))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'s0',str(s0))
   part = replace(part,'s1',str(s1))
   return part

def taper_x_z(part, x0, z0, z1, s0, s1):
   part = replace(part,'X','(x0 + (X-x0)*(z1-z0)/(s1*(Z-z0) + s0*(z1-Z)))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'s0',str(s0))
   part = replace(part,'s1',str(s1))
   return part

def taper_xy_z(part, x0, y0, z0, z1, s0, s1):
   part = replace(part,'X','(x0 + (X-x0)*(z1-z0)/(s1*(Z-z0) + s0*(z1-Z)))')
   part = replace(part,'Y','(y0 + (Y-y0)*(z1-z0)/(s1*(Z-z0) + s0*(z1-Z)))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'s0',str(s0))
   part = replace(part,'s1',str(s1))
   return part

def shear_x_y(part, y0, y1, dx0, dx1):
   part = replace(part,'X','(X - dx0 - (dx1-dx0)*(Y-y0)/(y1-y0))')
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'dx0',str(dx0))
   part = replace(part,'dx1',str(dx1))
   return part

def shear_x_z(part, z0, z1, dx0, dx1):
   part = replace(part,'X','(X - dx0 - (dx1-dx0)*(Z-z0)/(z1-z0))')
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'dx0',str(dx0))
   part = replace(part,'dx1',str(dx1))
   return part

def coshear_x_z(part, z0, z1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
   part = replace(part,'X','(X - offset - amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0)))')
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

#
# PCB classes and definitions
#

cad.labels = []

class PCB:
   def __init__(self,x0,y0,width,height,mask):
      self.board = "False"
      self.interior = rectangle(x0,x0+width,y0,y0+height)
      self.exterior = subtract("True",rectangle(x0,x0+width,y0,y0+height))
      self.mask = "False"
   def add(self,part):
      self.board = add(self.board,part)
      self.mask = add(self.mask,move(part,-mask,mask))
      self.mask = add(self.mask,move(part,-mask,-mask))
      self.mask = add(self.mask,move(part,mask,mask))
      self.mask = add(self.mask,move(part,mask,-mask))
      return self
```

```
class part:
   def add(self,pcb,x,y,z=0,angle=0):
      self.x = x
      self.y = y
      self.z = z
      self.angle = angle
      if (angle == 90):
         self.shape = rotate_90(self.shape)
      elif (angle == 180):
         self.shape = rotate_180(self.shape)
      elif ((angle == 270) | (angle == -90)):
         self.shape = rotate_270(self.shape)
      angle = pi*angle/180
      self.shape = translate(self.shape,x,y,z)
      for i in range(len(self.pad)):
         xnew = cos(angle)*self.pad[i].x + sin(angle)*self.pad[i].y
         ynew = -sin(angle)*self.pad[i].x + cos(angle)*self.pad[i].y
           self.pad[i].x = x + xnew
           self.pad[i].y = y + ynew
         self.pad[i].z += z
      cad.labels.append(cad_text(x,y,z,self.value,size=14))
      for i in range(len(self.labels)):
         xnew = cos(angle)*self.labels[i].x + sin(angle)*self.labels[i].y
         ynew = -sin(angle)*self.labels[i].x + cos(angle)*self.labels[i].y
           self.labels[i].x = x + xnew
           self.labels[i].y = y + ynew
         self.labels[i].z += z
           cad.labels.append(self.labels[i])
      pcb = pcb.add(self.shape)
      return pcb

def wire(pcb,width,*points):
   for i in range(1,len(points)):
      x0 = points[i-1].x
      y0 = points[i-1].y
      z0 = points[i-1].z
      x1 = points[i].x
      y1 = points[i].y
      z1 = points[i].z
      if (x0 < x1):
         pcb.board = add(pcb.board,cube(x0-width/2,x1+width/2,y0-width/2,y0+width/2,z0,z0))
      elif (x1 < x0):
         pcb.board = add(pcb.board,cube(x1-width/2,x0+width/2,y0-width/2,y0+width/2,z0,z0))
      if (y0 < y1):
         pcb.board = add(pcb.board,cube(x1-width/2,x1+width/2,y0-width/2,y1+width/2,z0,z0))
      elif (y1 < y0):
         pcb.board = add(pcb.board,cube(x1-width/2,x1+width/2,y1-width/2,y0+width/2,z0,z0))
   return pcb

#
# PCB library
#

#
# discretes
#

pad_0402 = cube(-.0175,.0175,-.014,.014,0,0)

class R_0402(part):
   #
   # 0402 resistor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_0402,-.0265,0,0)
      self.pad.append(point(-.0265,0,0))
      self.shape = add(self.shape,translate(pad_0402,.0265,0,0))
      self.pad.append(point(.0265,0,0))

pad_1206 = cube(-.032,.032,-.034,.034,0,0)

class R_1206(part):
   #
   # 1206 resistor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.06,0,0))
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.06,0,0))

class C_1206(part):
   #
   # 1206 capacitor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.06,0,0))
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.06,0,0))
```

```
pad_1210 = cube(-.032,.032,-.048,.048,0,0)

class L_1210(part):
   #
   # 1210 inductor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_1210,-.06,0,0)
      self.pad.append(point(-.06,0,0))
      self.shape = add(self.shape,translate(pad_1210,.06,0,0))
      self.pad.append(point(.06,0,0))

pad_choke = cube(-.06,.06,-.06,.06,0,0)

class choke(part):
   #
   # Panasonic ELLCTV
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_choke,-.177,-.177,0)
      self.pad.append(point(-.177,-.177,0))
      self.shape = add(self.shape,translate(pad_choke,.177,.177,0))
      self.pad.append(point(.177,.177,0))

#
# connectors
#

pad_MTA = cube(-.021,.021,-.041,.041,0,0)
pad_MTA_solder = cube(-.071,.071,-.041,.041,0,0)

class MTA_2(part):
   #
   # AMP 1445121-2
   # MTA .050 SMT 2-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1
      #
      self.shape = translate(pad_MTA,-.025,-.1,0)
      self.pad.append(point(-.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,.1,0))
      self.pad.append(point(.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'2',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.187,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.187,0,0))

class MTA_power(part):
   #
   # AMP 1445121-2
   # MTA .050 SMT 2-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: Gnd
      #
      self.shape = translate(pad_MTA,-.025,-.1,0)
      self.pad.append(point(-.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1\nGND',14))
      #
      # pin 2: Vcc
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,.1,0))
      self.pad.append(point(.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Vcc',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.187,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.187,0,0))

class MTA_i0(part):
   #
   # AMP 1445121-3
   # MTA .050 SMT 3-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
```

```
      # pin 1: GND
      #
      self.shape = translate(pad_MTA,.05,.1,0)
      self.pad.append(point(.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1\nGND',14))
      #
      # pin 2: power
      #
      self.shape = add(self.shape,translate(pad_MTA,-.05,.1,0))
      self.pad.append(point(-.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V',14))
      #
      # pin 3: data
      #
      self.shape = add(self.shape,translate(pad_MTA,0,-.1,0))
      self.pad.append(point(0,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'data',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.212,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.212,0,0))

class MTA_serial(part):
   #
   # AMP 1445121-4
   # MTA .050 SMT 4-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: Gnd
      #
      self.shape = translate(pad_MTA,-.075,-.1,0)
      self.pad.append(point(-.075,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: Tx
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,-.1,0))
      self.pad.append(point(.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Tx',14))
      #
      # pin 3: Rx
      #
      self.shape = add(self.shape,translate(pad_MTA,.075,.1,0))
      self.pad.append(point(.075,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Rx',14))
      #
      # pin 4: DTR
      #
      self.shape = add(self.shape,translate(pad_MTA,-.025,.1,0))
      self.pad.append(point(-.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DTR',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.237,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.237,0,0))

class MTA_PS2(part):
   #
   # AMP 1445121-4
   # MTA .050 SMT 4-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: GND
      #
      self.shape = translate(pad_MTA,-.075,-.1,0)
      self.pad.append(point(-.075,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1\nGND',14))
      #
      # pin 2: data
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,-.1,0))
      self.pad.append(point(.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'data',14))
      #
      # pin 3: clock
      #
      self.shape = add(self.shape,translate(pad_MTA,.075,.1,0))
      self.pad.append(point(.075,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'clock',14))
      #
      # pin 4: 5V
      #
      self.shape = add(self.shape,translate(pad_MTA,-.025,.1,0))
      self.pad.append(point(-.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'5V',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.237,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.237,0,0))
```

```
class MTA_ICP(part):
   #
   # AMP 1445121-5
   # MTA .050 SMT 4-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: MISO
      #
      self.shape = translate(pad_MTA,-.1,-.1,0)
      self.pad.append(point(-.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'MISO',14))
      #
      # pin 2: GND
      #
      self.shape = add(self.shape,translate(pad_MTA,0,-.1,0))
      self.pad.append(point(0,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))
      #
      # pin 3: MOSI
      #
      self.shape = add(self.shape,translate(pad_MTA,.1,-.1,0))
      self.pad.append(point(.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'MOSI',14))
      #
      # pin 4: -RESET
      #
      self.shape = add(self.shape,translate(pad_MTA,.05,.1,0))
      self.pad.append(point(.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'-RESET',14))
      #
      # pin 5: SCK
      #
      self.shape = add(self.shape,translate(pad_MTA,-.05,.1,0))
      self.pad.append(point(-.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'SCK',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.262,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.262,0,0))

class power_65mm(part):
   #
   # CUI PJ1-023-SMT
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: power
      #
      self.shape = cube(.433,.512,-.047,.047,0,0)
      self.pad.append(point(.467,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'P',14))
      #
      # pin 2: ground
      #
      self.shape = add(self.shape,cube(.285,.423,-.189,-.098,0,0))
      self.pad.append(point(.354,-.144,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 3: contact
      #
      self.shape = add(self.shape,cube(.325,.463,.098,.189,0,0))
      self.pad.append(point(.394,.144,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,cube(.108,.246,-.169,-.110,0,0))
      self.shape = add(self.shape,cube(.069,.207,.110,.169,0,0))

pad_stereo_2_5mm = cube(-.03,.03,-.05,.05,0,0)

class stereo_2_5mm(part):
   #
   # CUI SJ1-2533-SMT
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: base
      #
      self.shape = translate(pad_stereo_2_5mm,-.130,-.16,0)
      self.pad.append(point(-.130,-.149,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'base',14))
      #
      # pin 2: tip
      #
      self.shape = add(self.shape,translate(pad_stereo_2_5mm,.197,.15,0))
      self.pad.append(point(.197,.141,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'tip',14))
      #
      # pin 3: middle
```

```
      #
      self.shape = add(self.shape,translate(pad_stereo_2_5mm,-.012,-.16,0))
      self.pad.append(point(-.012,-.149,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'middle',14))

pad_Molex = cube(-.0155,.0155,-.0265,.0265,0,0)
pad_Molex_solder = cube(-.055,.055,-.065,.065,0,0)

class Molex_serial(part):
   #
   # Molex 53261-0471
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: Rx
      #
      self.shape = translate(pad_Molex,-.075,.064,0)
      self.pad.append(point(-.075,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Rx',14))
      #
      # pin 2: Tx
      #
      self.shape = add(self.shape,translate(pad_Molex,-.025,.064,0))
      self.pad.append(point(-.025,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Tx',14))
      #
      # pin 3: DTR
      #
      self.shape = add(self.shape,translate(pad_Molex,.025,.064,0))
      self.pad.append(point(.025,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DTR',14))
      #
      # pin 4: GND
      #
      self.shape = add(self.shape,translate(pad_Molex,.075,.064,0))
      self.pad.append(point(.075,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_Molex_solder,-.16,-.065,0))
      self.shape = add(self.shape,translate(pad_Molex_solder,.16,-.065,0))

#
# switches
#

pad_button_6mm = cube(-.04,.04,-.03,.03,0,0)

class button_6mm(part):
   #
   # Omron 6mm pushbutton
   # B3SN-3112P
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # left 1
      #
      self.shape = translate(pad_button_6mm,-.125,.08,0)
      self.pad.append(point(-.125,.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'L1',14))
      #
      # right 1
      #
      self.shape = add(self.shape,translate(pad_button_6mm,-.125,-.08,0))
      self.pad.append(point(-.125,-.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'R1',14))
      #
      # right 2
      #
      self.shape = add(self.shape,translate(pad_button_6mm,.125,-.08,0))
      self.pad.append(point(.125,-.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'R2',14))
      #
      # left 2
      #
      self.shape = add(self.shape,translate(pad_button_6mm,.125,.08,0))
      self.pad.append(point(.125,.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'L2',14))

#
# crystals and resonators
#

pad_XTAL = cube(-.016,.016,-.077,.077,0,0)

class XTAL(part):
   #
   # Panasonic EFOBM series
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
```

```
      # left
      #
      self.shape = translate(pad_XTAL,-.053,0,0)
      self.pad.append(point(-.053,0,0))
      #
      # ground
      #
      self.shape = add(self.shape,translate(pad_XTAL,0,0,0))
      self.pad.append(point(0,0,0))
      #
      # right
      #
      self.shape = add(self.shape,translate(pad_XTAL,.053,0,0))
      self.pad.append(point(.053,0,0))

#
# diodes, transistors, regulators
#

class D_1206(part):
   #
   # 1206 diode
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # anode
      #
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A',14))
      #
      # cathode
      #
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))

class LED_1206(part):
   #
   # 1206 LED
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # anode
      #
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A',14))
      #
      # cathode
      #
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))

class phototransistor_1206(part):
   #
   # 1206 phototransistor
   # OPTEK 520,521
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # collector
      #
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))
      #
      # emitter
      #
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'E',14))

pad_SOD_123 = cube(-.02,.02,-.024,.024,0,0)

class D_SOD_123(part):
   #
   # SOD-123 diode
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # anode
      #
      self.shape = translate(pad_SOD_123,-.07,0,0)
      self.pad.append(point(-.07,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A',14))
      #
      # cathode
```

```
      #
      self.shape = add(self.shape,translate(pad_SOD_123,.07,0,0))
      self.pad.append(point(.07,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))

pad_SOT23 = cube(-.015,.015,-.02,.02,0,0)

class NMOSFET_SOT23(part):
   #
   # Fairchild NDS355AN
   #
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: gate
      #
      self.shape = translate(pad_SOT23,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 2: source
      #
      self.shape = add(self.shape,translate(pad_SOT23,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'S',14))
      #
      # pin 3: drain
      #
      self.shape = add(self.shape,translate(pad_SOT23,0,.045,0))
      self.pad.append(point(0,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'D',14))

class PMOSFET_SOT23(part):
   #
   # Fairchild NDS356AP
   #
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: gate
      #
      self.shape = translate(pad_SOT23,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 2: source
      #
      self.shape = add(self.shape,translate(pad_SOT23,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'S',14))
      #
      # pin 3: drain
      #
      self.shape = add(self.shape,translate(pad_SOT23,0,.045,0))
      self.pad.append(point(0,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'D',14))

class regulator_SOT23(part):
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: output
      #
      self.shape = translate(pad_SOT23,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: input
      #
      self.shape = add(self.shape,translate(pad_SOT23,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'in',14))
      #
      # pin 3: ground
      #
      self.shape = add(self.shape,translate(pad_SOT23,0,.045,0))
      self.pad.append(point(0,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'gnd',14))

pad_SOT223 = cube(-.02,.02,-.03,.03,0,0)
pad_SOT223_ground = cube(-.065,.065,-.03,.03,0,0)

class regulator_SOT223(part):
   def __init__(self,value=''):
      self.value = value
      self.x = 0
```

```
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: input
      #
      self.shape = translate(pad_SOT223,-.09,-.12,0)
      self.pad.append(point(-.09,-.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1I',14))
      #
      # pin 2: ground
      #
      self.shape = add(self.shape,translate(pad_SOT223,0,-.12,0))
      self.pad.append(point(0,-.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 3: output
      #
      self.shape = add(self.shape,translate(pad_SOT223,.09,-.12,0))
      self.pad.append(point(.09,-.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'O',14))
      #
      # pin 4: ground
      #
      self.shape = add(self.shape,translate(pad_SOT223_ground,0,.12,0))
      self.pad.append(point(0,.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))

pad_SM8 = cube(-.035,.035,-.016,.016,0,0)

class H_bridge_SM8(part):
   #
   # Zetex ZXMHC3A01T8
   #
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      d = .13
      #
      # pin 1: G3 (right N gate)
      #
      self.shape = translate(pad_SM8,-d,.09,0)
      self.pad.append(point(-d,.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GRN',14))
      #
      # pin 2: S2 S3 (N source)
      #
      self.shape = add(self.shape,translate(pad_SM8,-d,.03,0))
      self.pad.append(point(-d,.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'SN',14))
      #
      # pin 3: G2 (left N gate)
      #
      self.shape = add(self.shape,translate(pad_SM8,-d,-.03,0))
      self.pad.append(point(-d,-.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GLN',14))
      #
      # pin 4: G1 (left P gate)
      #
      self.shape = add(self.shape,translate(pad_SM8,-d,-.09,0))
      self.pad.append(point(-d,-.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GLP',14))
      #
      # pin 5: D1 D2 (left drain)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,-.09,0))
      self.pad.append(point(d,-.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DL',14))
      #
      # pin 6: S1 S4 (P source)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,-.03,0))
      self.pad.append(point(d,-.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'SP',14))
      #
      # pin 7: D3 D4 (right drain)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,.03,0))
      self.pad.append(point(d,.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DR',14))
      #
      # pin 8: G4 (right N gate)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,.09,0))
      self.pad.append(point(d,.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GRP',14))

#
# ICs
#

pad_SOT23_5 = cube(-.01,.01,-.02,.02,0,0)

class op_amp_SOT23_5(part):
   def __init__(self,value=''):
      self.value = value
```

```
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: output
      #
      self.shape = translate(pad_SOT23_5,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'O',14))
      #
      # pin 2: V-
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,0,-.045,0))
      self.pad.append(point(0,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V-',14))
      #
      # pin 3: I+
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'I+',14))
      #
      # pin 4: I-
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,.0375,.045,0))
      self.pad.append(point(.0375,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'I-',14))
      #
      # pin 5: V+
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,-.0375,.045,0))
      self.pad.append(point(-.0375,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V+',14))

pad_SOICN = cube(-.035,.035,-.015,.015,0,0)

class op_amp_SOICN(part):
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: A out
      #
      self.shape = translate(pad_SOICN,-.12,.075,0)
      self.pad.append(point(-.12,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1 Ao',14))
      #
      # pin 2: A-
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.025,0))
      self.pad.append(point(-.12,.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A-',14))
      #
      # pin 3: A+
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.025,0))
      self.pad.append(point(-.12,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A+',14))
      #
      # pin 4: V-
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.075,0))
      self.pad.append(point(-.12,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V-',14))
      #
      # pin 5: B+
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.075,0))
      self.pad.append(point(.12,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'B+',14))
      #
      # pin 6: B-
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.025,0))
      self.pad.append(point(.12,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'B-',14))
      #
      # pin 7: B out
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.025,0))
      self.pad.append(point(.12,.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Bo',14))
      #
      # pin 8: V+
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.075,0))
      self.pad.append(point(.12,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V+',14))

pad_SOIC = cube(-.043,.043,-.015,.015,0,0)

class ATtiny45_SOIC(part):
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: PB5/dW/ADC0/-RESET/PCINT5
```

```
        #
        self.shape = translate(pad_SOIC,-.14,.075,0)
        self.pad.append(point(-.14,.075,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
        #
        # pin 2: PB3/ADC3/-OC1B/CLKI/XTAL1/PCINT3
        #
        self.shape = add(self.shape,translate(pad_SOIC,-.14,.025,0))
        self.pad.append(point(-.14,.025,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB3',14))
        #
        # pin 3: PB4/ADC2/OC1B/CLKO/XTAL2/PCINT4
        #
        self.shape = add(self.shape,translate(pad_SOIC,-.14,-.025,0))
        self.pad.append(point(-.14,-.025,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB4',14))
        #
        # pin 4: GND
        #
        self.shape = add(self.shape,translate(pad_SOIC,-.14,-.075,0))
        self.pad.append(point(-.14,-.075,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))
        #
        # pin 5: PB0/MOSI/DI/SDA/AIN0/OC0A/-OC1A/AREF/PCINT0
        #
        self.shape = add(self.shape,translate(pad_SOIC,.14,-.075,0))
        self.pad.append(point(.14,-.075,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 6: PB1/MISO/DO/AIN1/OC0B/OC1A/PCINT1
        #
        self.shape = add(self.shape,translate(pad_SOIC,.14,-.025,0))
        self.pad.append(point(.14,-.025,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB1',14))
        #
        # pin 7: PB2/SCK/USCK/SCL/ADC1/T0/INT0/PCINT2
        #
        self.shape = add(self.shape,translate(pad_SOIC,.14,.025,0))
        self.pad.append(point(.14,.025,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB2',14))
        #
        # pin 8: VCC
        #
        self.shape = add(self.shape,translate(pad_SOIC,.14,.075,0))
        self.pad.append(point(.14,.075,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'VCC',14))

class ATtiny44_SOICN(part):
   def __init__(self,value=''):
        self.value = value
        self.pad = [point(0,0,0)]
        self.labels = []
        #
        # pin 1: VCC
        #
        self.shape = translate(pad_SOICN,-.12,.15,0)
        self.pad.append(point(-.12,.15,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
        #
        # pin 2: PB0/XTAL1/PCINT8
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 3: PB1/XTAL2/PCINT9
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.050,0))
        self.pad.append(point(-.12,.05,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB1',14))
        #
        # pin 4: PB3/dW/-RESET/PCINT11
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,0,0))
        self.pad.append(point(-.12,0,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB3',14))
        #
        # pin 5: PB2/CKOUT/OC0A/INT0/PCINT10
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,-.05,0))
        self.pad.append(point(-.12,-.05,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB2',14))
        #
        # pin 6: PA7/ADC7/OC0B/ICP/PCINT7
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,-.1,0))
        self.pad.append(point(-.12,-.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA7',14))
        #
        # pin 7: PA6/ADC6/MOSI/SDA/OC1A/PCINT6
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,-.15,0))
        self.pad.append(point(-.12,-.15,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA6',14))
        #
        # pin 8: PA5/ADC5/DO/MISO/OC1B/PCINT5
        #
        self.shape = add(self.shape,translate(pad_SOICN,.12,-.15,0))
        self.pad.append(point(.12,-.15,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA5',14))
        #
```

```
      # pin 9: PA4/ADC4/USCK/SCL/T1/PCINT4
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.1,0))
      self.pad.append(point(.12,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA4',14))
      #
      # pin 10: PA3/ADC3/T0/PCINT3
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.05,0))
      self.pad.append(point(.12,-.05,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA3',14))
      #
      # pin 11: PA2/ADC2/AIN1/PCINT2
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,0,0))
      self.pad.append(point(.12,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA2',14))
      #
      # pin 12: PA1/ADC1/AIN0/PCINT1
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.050,0))
      self.pad.append(point(.12,.05,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA1',14))
      #
      # pin 13: PA0/ADC0/AREF/PCINT0
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.1,0))
      self.pad.append(point(.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA0',14))
      #
      # pin 14: GND
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.15,0))
      self.pad.append(point(.12,.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))

pad_TQFP = cube(-.043,.043,-.015,.015,0,0)

class ATmega88_TQFP(part):
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []

      #
      # pin 1: PD3/PCINT19/OC2B/INT1
      #
      self.shape = translate(pad_SOICN,-.12,.15,0)
      self.pad.append(point(-.12,.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: PD4/PCINT20/XCK/T0
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 3: GND
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 4: VCC
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 5: GND
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 6: VCC
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 7: PB6/PCINT6/XTAL1/TOSC1
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 8: PB7/PCINT7/XTAL2/TOSC2
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

      #
      # pin 9: PD5/PCINT21/OC0B/T1
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 10: PD6/PCINT22/OC0A/AIN0
```

```
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 11: PD7/PCINT23/AIN1
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 12: PB0/PCINT0/CLKO/ICP1
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 13: PB1/PCINT1/OC1A
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 14: PB2/PCINT2/-SS/OC1B
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 15: PB3/PCINT3/OC2A/MOSI
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 16: PB4/PCINT4/MISO
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

#
# pin 17: PB5/SCK/PCINT5
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 18: AVCC
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 19: ADC6
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 20: AREF
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 21: GND
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 22: ADC7
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 23: PC0/ADC0/PCINT8
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 24: PC1/ADC1/PCINT9
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

#
# pin 25: PC2/ADC2/PCINT10
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 26: PC2/ADC3/PCINT11
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
```

```
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 27: PC4/ADC4/SDA/PCINT12
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 28: PC5/ADC5/SCL/PCINT13
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 29: PC6/-RESET/PCINT14
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 30: PD0/RXD/PCINT16
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 31: PD1/TXD/PCINT17
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 32: PD2/INT0/PCINT18
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

#
# graphics
#

class CBA(part):
   def __init__(self,r=.02):
      self.value = ''
      self.pad = [point(0,0,0)]
      self.labels = []
      d = 3*r
      self.shape = circle(0,0,r)
      self.shape = add(self.shape,translate(circle(0,0,r),-d,d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),-d,0,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),-d,-d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),0,-d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),d,-d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),d,0,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),d,d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),0,d,0))

class hello(part):
   #
   # HELLO
   #
   def __init__(self,w,z):
      self.value = ''
      self.pad = [point(0,0,0)]
      self.labels = []
      # H
      self.shape = rectangle(-.01,.01,0,.1)
      self.shape = add(self.shape,rectangle(0,.05,.04,.06))
      self.shape = add(self.shape,rectangle(.04,.06,0,.1))
      # E
      self.shape = add(self.shape,rectangle(.08,.1,0,.1))
      self.shape = add(self.shape,rectangle(.1,.14,0,.02))
      self.shape = add(self.shape,rectangle(.1,.13,.04,.06))
      self.shape = add(self.shape,rectangle(.1,.14,.08,.1))
      # L
      self.shape = add(self.shape,rectangle(.16,.18,0,.1))
      self.shape = add(self.shape,rectangle(.18,.21,0,.02))
      # L
      self.shape = add(self.shape,rectangle(.23,.25,0,.1))
      self.shape = add(self.shape,rectangle(.25,.28,0,.02))
      # 0
      self.shape = add(self.shape,rectangle(.3,.32,0,.1))
      self.shape = add(self.shape,rectangle(.32,.345,0,.02))
      self.shape = add(self.shape,rectangle(.32,.345,.08,.1))
      self.shape = add(self.shape,rectangle(.345,.365,0,.1))

#
# define board
#

w = .016
width = 1.0
height = .98
x = 1
y = 1
z = -.005
d = .06
mask = .004

pcb = PCB(x,y,width,height,mask)
```

```
IC1 = ATtiny44_SOICN('IC1\nt44')
pcb = IC1.add(pcb,x+.65,y+.505,z)

XTAL1 = XTAL('XTAL1\n20 MHz')
pcb = XTAL1.add(pcb,IC1.pad[2].x-.2,IC1.pad[2].y-.025,z,angle=90)

pcb = wire(pcb,w,
   IC1.pad[3],
   point(XTAL1.pad[3].x+.12,XTAL1.pad[3].y,z),
   XTAL1.pad[3])

pcb = wire(pcb,w,
   IC1.pad[2],
   point(XTAL1.pad[1].x+.12,XTAL1.pad[1].y,z),
   XTAL1.pad[1])

J1 = MTA_ICP('J1\nICP')
pcb = J1.add(pcb,IC1.x,IC1.pad[7].y-.2,z,angle=180)

pcb = wire(pcb,w,
   IC1.pad[7],
   J1.pad[3])

pcb = wire(pcb,w,
   IC1.pad[4],
   J1.pad[4])

pcb = wire(pcb,w,
   IC1.pad[14],
   point(J1.pad[5].x,IC1.pad[10].y,z),
   J1.pad[2])

pcb = wire(pcb,w,
   IC1.pad[9],
   J1.pad[5])

pcb = wire(pcb,w,
   IC1.pad[8],
   J1.pad[1])

J2 = MTA_power('J2\npower')
pcb = J2.add(pcb,IC1.x-.37,IC1.pad[1].y+.17,z,angle=0)

pcb = wire(pcb,w,
   XTAL1.pad[2],
   point(J2.pad[1].x-.1,J2.pad[1].y,z),
   J2.pad[1])

IC2 = regulator_SOT23('IC2\n5V')
pcb = IC2.add(pcb,J2.x+.38,J2.y+.02,z,angle=180)

pcb = wire(pcb,w,
   IC2.pad[3],
   point(IC2.pad[3].x-.09,J2.pad[1].y+.03,z),
   J2.pad[1])

pcb = wire(pcb,w,
   IC1.pad[1],
   point(J1.pad[4].x+.01,IC1.pad[1].y+.09,z),
   IC2.pad[1])

pcb = wire(pcb,w,
   J2.pad[2],
   IC2.pad[2])

R1 = R_1206('R1\n10k')
pcb = R1.add(pcb,IC2.x+.13,IC2.y-.03,z,angle=90)

pcb = wire(pcb,w,
   IC2.pad[1],
   R1.pad[1])

pcb = wire(pcb,w,
   IC1.pad[4],
   point(IC1.x,IC1.pad[14].y+.05,z),
   point(R1.pad[2].x-.025,R1.pad[2].y,z))

C1 = C_1206('C1\n3.3uF')
pcb = C1.add(pcb,R1.x+.11,R1.y,z,angle=90)

pcb = wire(pcb,w,
   C1.pad[1],
   R1.pad[1])

pcb = wire(pcb,w,
   IC1.pad[14],
   C1.pad[2])

pcb = wire(pcb,w,
   C1.pad[2],
   point(C1.x+.07,C1.pad[1].y+.07,z),
   IC2.pad[3])

J3 = MTA_2('J3\nvideo')
pcb = J3.add(pcb,IC1.x-.49,IC1.y-.23,z,angle=-90)

pcb = wire(pcb,w,
   J2.pad[1],
   point(J3.pad[2].x,J3.pad[2].y+.05,z),
   point(J3.x,J3.pad[1].y,z),
```

```
    J3.pad[1])

R2 = R_1206('R2\n499')
pcb = R2.add(pcb,IC1.pad[5].x-.16,IC1.pad[5].y-.025,z,angle=0)

pcb = wire(pcb,w,
    IC1.pad[6],
    R2.pad[2])

R3 = R_1206('R3\n1k')
pcb = R3.add(pcb,R2.x,R2.y-.1,z,angle=0)

pcb = wire(pcb,w,
    IC1.pad[7],
    R3.pad[2])

pcb = wire(pcb,w,
    R2.pad[1],
    R3.pad[1])

pcb = wire(pcb,w,
    J3.pad[2],
    point(J3.pad[2].x,J3.pad[1].y-.05,z),
    point(J3.pad[1].x,J1.pad[4].y,z),
    point(J1.pad[3].x-.05,J1.pad[3].y,z),
    point(R3.x,R3.y,z),
    R3.pad[1])

H1 = hello(w,z)
pcb = H1.add(pcb,x+width-.04,y+.25,z,angle=-90)

cad.function = add(pcb.board,pcb.exterior)

#
# uncomment to export traces
#

#cad.function = pcb.board

#
# uncomment to export exterior
#

#cad.function = pcb.exterior

#
# uncomment to export solder mask
#

#cad.function = pcb.mask

#
# uncomment to mill out board
#
# use 1/32 end-mill
# set tool diameter = .0312
# set xy, z speed = .5
# set # contours = 1
#

#cad.function = pcb.interior
#z = -.065

#
# define limits and parameters
#

cad.xmin = x-.1 # min x to render
cad.xmax = x+width+.1 # max x to render
cad.ymin = y-.1 # min y to render
cad.ymax = y+width+.1 # max y to render
cad.zmin = z
cad.zmax = .050
dpi = 200 # low resolution for previewing
#dpi = 500 # high resolution for machining
nxy = int(dpi*(cad.xmax-cad.xmin))
cad.nx = nxy # x points to render
cad.ny = nxy # y points to render
cad.nz = 1
cad.view('xy') # 2D view
```

```
#
# hello.LCD.44.cad
#
# 16x2 LCD hello-world
#
# Neil Gershenfeld
# CBA MIT 10/18/07
#
# (c) Massachusetts Institute of Technology 2007
# Permission granted for experimental and personal use;
# license for commercial sale available from MIT.
#

#
# define shapes and transformation
#
# circle(x0, y0, r)
# cylinder(x0, y0, z0, z1, r)
# cone(x0, y0, z0, z1, r0)
# sphere(x0, y0, z0, r)
# torus(x0, y0, z0, r0, r1)
# rectangle(x0, x1, y0, y1)
# cube(x0, x1, y0, y1, z0, z1)
# triangle(x0, y0, x1, y1, x2, y2) (points in clockwise order)
# pyramid(x0, x1, y0, y1, z0, z1)
# function(Z_of_XY)
# functions(upper_Z_of_XY,lower_Z_of_XY)
# add(part1, part2)
# subtract(part1, part2)
# intersect(part1, part2)
# move(part,dx,dy)
# translate(part,dx,dy,dz)
# rotate(part, angle)
# rotate_x(part, angle)
# rotate_y(part, angle)
# rotate_z(part, angle)
# rotate_90(part)
# rotate_180(part)
# rotate_270(part)
# reflect_x(part)
# reflect_y(part)
# reflect_z(part)
# reflect_xy(part)
# reflect_xz(part)
# reflect_yz(part)
# scale_x(part, x0, sx)
# scale_y(part, y0, sy)
# scale_z(part, z0, sz)
# scale_xy(part, x0, y0, sxy)
# scale_xyz(part, x0, y0, z0, sxyz)
# coscale_x_y(part, x0, y0, y1, angle0, angle1, amplitude, offset)
# coscale_x_z(part, x0, z0, z1, angle0, angle1, amplitude, offset)
# coscale_xy_z(part, x0, y0, z0, z1, angle0, angle1, amplitude, offset)
# taper_x_y(part, x0, y0, y1, s0, s1)
# taper_x_z(part, x0, z0, z1, s0, s1)
# taper_xy_z(part, x0, y0, z0, z1, s0, s1)
# shear_x_y(part, y0, y1, dx0, dx1)
# shear_x_z(part, z0, z1, dx0, dx1)
# (more to come)

def circle(x0, y0, r):
   part = "(((X-x0)**2 + (Y-y0)**2) <= r**2)"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'r',str(r))
   return part

def cylinder(x0, y0, z0, z1, r):
   part = "(((X-x0)**2 + (Y-y0)**2 <= r**2) & (Z >= z0) & (Z <= z1))"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'r',str(r))
   return part

def cone(x0, y0, z0, z1, r0):
   part = cylinder(x0, y0, z0, z1, r0)
   part = taper_xy_z(part, x0, y0, z0, z1, 1.0, 0.0)
   return part

def sphere(x0, y0, z0, r):
   part = "(((X-x0)**2 + (Y-y0)**2 + (Z-z0)**2) <= r**2)"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'r',str(r))
   return part

def torus(x0, y0, z0, r0, r1):
   part = "(((r0 - sqrt((X-x0)**2 + (Y-y0)**2))**2 + (Z-z0)**2) <= r1**2)"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'r0',str(r0))
   part = replace(part,'r1',str(r1))
   return part

def rectangle(x0, x1, y0, y1):
   part = "((X >= x0) & (X <= x1) & (Y >= y0) & (Y <= y1))"
   part = replace(part,'x0',str(x0))
```

```
   part = replace(part,'x1',str(x1))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   return part

def cube(x0, x1, y0, y1, z0, z1):
   part = "((X >= x0) & (X <= x1) & (Y >= y0) & (Y <= y1) & (Z >= z0) & (Z <= z1))"
   part = replace(part,'x0',str(x0))
   part = replace(part,'x1',str(x1))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   return part

def triangle(x0, y0, x1, y1, x2, y2): # points in clockwise order
   part = "((((y1-y0)*(X-x0)-(x1-x0)*(Y-y0)) >= 0) & (((y2-y1)*(X-x1)-(x2-x1)*(Y-y1)) >= 0) & (((y0-y2)*(X-x2)-(x0-x2)*(Y-y2)) >= 0))"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'x1',str(x1))
   part = replace(part,'y1',str(y1))
   part = replace(part,'x2',str(x2))
   part = replace(part,'y2',str(y2))
   return part

def pyramid(x0, x1, y0, y1, z0, z1):
   part = cube(x0, x1, y0, y1, z0, z1)
   part = taper_xy_z(part, (x0+x1)/2., (y0+y1)/2., z0, z1, 1.0, 0.0)
   return part

def function(Z_of_XY):
   part = '(Z <= '+Z_of_XY+')'
   return part

def functions(upper_Z_of_XY,lower_Z_of_XY):
   part = '(Z <= '+upper_Z_of_XY+') & (Z >= '+lower_Z_of_XY+')'
   return part

def add(part1, part2):
   part = "part1 | part2"
   part = replace(part,'part1',part1)
   part = replace(part,'part2',part2)
   return part

def subtract(part1, part2):
   part = "(part1) & ~(part2)"
   part = replace(part,'part1',part1)
   part = replace(part,'part2',part2)
   return part

def intersect(part1, part2):
   part = "(part1) & (part2)"
   part = replace(part,'part1',part1)
   part = replace(part,'part2',part2)
   return part

def move(part,dx,dy):
   part = replace(part,'X','(X-'+str(dx)+')')
   part = replace(part,'Y','(Y-'+str(dy)+')')
   return part

def translate(part,dx,dy,dz):
   part = replace(part,'X','(X-'+str(dx)+')')
   part = replace(part,'Y','(Y-'+str(dy)+')')
   part = replace(part,'Z','(Z-'+str(dz)+')')
   return part

def rotate(part, angle):
   angle = angle*pi/180
   part = replace(part,'X','(cos(angle)*X+sin(angle)*y)')
   part = replace(part,'Y','(-sin(angle)*X+cos(angle)*y)')
   part = replace(part,'y','Y')
   part = replace(part,'angle',str(angle))
   return part

def rotate_x(part, angle):
   angle = angle*pi/180
   part = replace(part,'Y','(cos(angle)*Y+sin(angle)*z)')
   part = replace(part,'Z','(-sin(angle)*Y+cos(angle)*z)')
   part = replace(part,'z','Z')
   part = replace(part,'angle',str(angle))
   return part

def rotate_y(part, angle):
   angle = angle*pi/180
   part = replace(part,'X','(cos(angle)*X+sin(angle)*z)')
   part = replace(part,'Z','(-sin(angle)*X+cos(angle)*z)')
   part = replace(part,'z','Z')
   part = replace(part,'angle',str(angle))
   return part

def rotate_z(part, angle):
   angle = angle*pi/180
   part = replace(part,'X','(cos(angle)*X+sin(angle)*y)')
   part = replace(part,'Y','(-sin(angle)*X+cos(angle)*y)')
   part = replace(part,'y','Y')
   part = replace(part,'angle',str(angle))
   return part

def rotate_90(part):
   part = reflect_xy(part)
```

```
   part = reflect_y(part)
   return part

def rotate_180(part):
   part = reflect_xy(part)
   part = reflect_y(part)
   part = reflect_xy(part)
   part = reflect_y(part)
   return part

def rotate_270(part):
   part = reflect_xy(part)
   part = reflect_y(part)
   part = reflect_xy(part)
   part = reflect_y(part)
   part = reflect_xy(part)
   part = reflect_y(part)
   return part

def reflect_x(part):
   part = replace(part,'X','-X')
   return part

def reflect_y(part):
   part = replace(part,'Y','-Y')
   return part

def reflect_z(part):
   part = replace(part,'Z','-Z')
   return part

def reflect_xy(part):
   part = replace(part,'X','temp')
   part = replace(part,'Y','X')
   part = replace(part,'temp','Y')
   return part

def reflect_xz(part):
   part = replace(part,'X','temp')
   part = replace(part,'Z','X')
   part = replace(part,'temp','Z')
   return part

def reflect_yz(part):
   part = replace(part,'Y','temp')
   part = replace(part,'Z','Y')
   part = replace(part,'temp','Z')
   return part

def scale_x(part, x0, sx):
   part = replace(part,'X','(x0 + (X-x0)/sx)')
   part = replace(part,'x0',str(x0))
   part = replace(part,'sx',str(sx))
   return part

def scale_y(part, y0, sy):
   part = replace(part,'Y','(y0 + (Y-y0)/sy)')
   part = replace(part,'y0',str(y0))
   part = replace(part,'sy',str(sy))
   return part

def scale_z(part, z0, sz):
   part = replace(part,'Z','(z0 + (Z-z0)/sz)')
   part = replace(part,'z0',str(z0))
   part = replace(part,'sz',str(sz))
   return part

def scale_xy(part, x0, y0, sxy):
   part = replace(part,'X','(x0 + (X-x0)/sxy)')
   part = replace(part,'Y','(y0 + (Y-y0)/sxy)')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'sxy',str(sxy))
   return part

def scale_xyz(part, x0, y0, z0, sxyz):
   part = replace(part,'X','(x0 + (X-x0)/sxyz)')
   part = replace(part,'Y','(y0 + (Y-y0)/sxyz)')
   part = replace(part,'Z','(z0 + (Z-z0)/sxyz)')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'sxyz',str(sxyz))
   return part

def coscale_x_y(part, x0, y0, y1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
   part = replace(part,'X','(x0 + (X-x0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Y-y0)/(y1-y0))))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

def coscale_x_z(part, x0, z0, z1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
```

```
   part = replace(part,'X','(x0 + (X-x0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0))))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

def coscale_xy_z(part, x0, y0, z0, z1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
   part = replace(part,'X','(x0 + (X-x0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0))))')
   part = replace(part,'Y','(y0 + (Y-y0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0))))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

def taper_x_y(part, x0, y0, y1, s0, s1):
   part = replace(part,'X','(x0 + (X-x0)*(y1-y0)/(s1*(Y-y0) + s0*(y1-Y)))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'s0',str(s0))
   part = replace(part,'s1',str(s1))
   return part

def taper_x_z(part, x0, z0, z1, s0, s1):
   part = replace(part,'X','(x0 + (X-x0)*(z1-z0)/(s1*(Z-z0) + s0*(z1-Z)))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'s0',str(s0))
   part = replace(part,'s1',str(s1))
   return part

def taper_xy_z(part, x0, y0, z0, z1, s0, s1):
   part = replace(part,'X','(x0 + (X-x0)*(z1-z0)/(s1*(Z-z0) + s0*(z1-Z)))')
   part = replace(part,'Y','(y0 + (Y-y0)*(z1-z0)/(s1*(Z-z0) + s0*(z1-Z)))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'s0',str(s0))
   part = replace(part,'s1',str(s1))
   return part

def shear_x_y(part, y0, y1, dx0, dx1):
   part = replace(part,'X','(X - dx0 - (dx1-dx0)*(Y-y0)/(y1-y0))')
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'dx0',str(dx0))
   part = replace(part,'dx1',str(dx1))
   return part

def shear_x_z(part, z0, z1, dx0, dx1):
   part = replace(part,'X','(X - dx0 - (dx1-dx0)*(Z-z0)/(z1-z0))')
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'dx0',str(dx0))
   part = replace(part,'dx1',str(dx1))
   return part

def coshear_x_z(part, z0, z1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
   part = replace(part,'X','(X - offset - amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0)))')
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

#
# PCB classes and definitions
#

cad.labels = []

class PCB:
   def __init__(self,x0,y0,width,height,mask):
      self.board = "False"
      self.interior = rectangle(x0,x0+width,y0,y0+height)
      self.exterior = subtract("True",rectangle(x0,x0+width,y0,y0+height))
      self.mask = "False"
   def add(self,part):
      self.board = add(self.board,part)
      self.mask = add(self.mask,move(part,-mask,mask))
      self.mask = add(self.mask,move(part,-mask,-mask))
      self.mask = add(self.mask,move(part,mask,mask))
      self.mask = add(self.mask,move(part,mask,-mask))
      return self
```

```
class part:
   def add(self,pcb,x,y,z=0,angle=0):
      self.x = x
      self.y = y
      self.z = z
      self.angle = angle
      if (angle == 90):
         self.shape = rotate_90(self.shape)
      elif (angle == 180):
         self.shape = rotate_180(self.shape)
      elif ((angle == 270) | (angle == -90)):
         self.shape = rotate_270(self.shape)
      angle = pi*angle/180
      self.shape = translate(self.shape,x,y,z)
      for i in range(len(self.pad)):
         xnew = cos(angle)*self.pad[i].x + sin(angle)*self.pad[i].y
         ynew = -sin(angle)*self.pad[i].x + cos(angle)*self.pad[i].y
           self.pad[i].x = x + xnew
           self.pad[i].y = y + ynew
         self.pad[i].z += z
      cad.labels.append(cad_text(x,y,z,self.value,size=14))
      for i in range(len(self.labels)):
         xnew = cos(angle)*self.labels[i].x + sin(angle)*self.labels[i].y
         ynew = -sin(angle)*self.labels[i].x + cos(angle)*self.labels[i].y
           self.labels[i].x = x + xnew
           self.labels[i].y = y + ynew
         self.labels[i].z += z
           cad.labels.append(self.labels[i])
      pcb = pcb.add(self.shape)
      return pcb

def wire(pcb,width,*points):
   for i in range(1,len(points)):
      x0 = points[i-1].x
      y0 = points[i-1].y
      z0 = points[i-1].z
      x1 = points[i].x
      y1 = points[i].y
      z1 = points[i].z
      if (x0 < x1):
         pcb.board = add(pcb.board,cube(x0-width/2,x1+width/2,y0-width/2,y0+width/2,z0,z0))
      elif (x1 < x0):
         pcb.board = add(pcb.board,cube(x1-width/2,x0+width/2,y0-width/2,y0+width/2,z0,z0))
      if (y0 < y1):
         pcb.board = add(pcb.board,cube(x1-width/2,x1+width/2,y0-width/2,y1+width/2,z0,z0))
      elif (y1 < y0):
         pcb.board = add(pcb.board,cube(x1-width/2,x1+width/2,y1-width/2,y0+width/2,z0,z0))
   return pcb

#
# PCB library
#

#
# discretes
#

pad_0402 = cube(-.0175,.0175,-.014,.014,0,0)

class R_0402(part):
   #
   # 0402 resistor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_0402,-.0265,0,0)
      self.pad.append(point(-.0265,0,0))
      self.shape = add(self.shape,translate(pad_0402,.0265,0,0))
      self.pad.append(point(.0265,0,0))

pad_1206 = cube(-.032,.032,-.034,.034,0,0)

class R_1206(part):
   #
   # 1206 resistor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.06,0,0))
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.06,0,0))

class C_1206(part):
   #
   # 1206 capacitor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.06,0,0))
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.06,0,0))
```

```
pad_1210 = cube(-.032,.032,-.048,.048,0,0)

class L_1210(part):
   #
   # 1210 inductor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_1210,-.06,0,0)
      self.pad.append(point(-.06,0,0))
      self.shape = add(self.shape,translate(pad_1210,.06,0,0))
      self.pad.append(point(.06,0,0))

pad_choke = cube(-.06,.06,-.06,.06,0,0)

class choke(part):
   #
   # Panasonic ELLCTV
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_choke,-.177,-.177,0)
      self.pad.append(point(-.177,-.177,0))
      self.shape = add(self.shape,translate(pad_choke,.177,.177,0))
      self.pad.append(point(.177,.177,0))

#
# connectors
#

pad_MTA = cube(-.021,.021,-.041,.041,0,0)
pad_MTA_solder = cube(-.071,.071,-.041,.041,0,0)

class MTA_2(part):
   #
   # AMP 1445121-2
   # MTA .050 SMT 2-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1
      #
      self.shape = translate(pad_MTA,-.025,-.1,0)
      self.pad.append(point(-.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,.1,0))
      self.pad.append(point(.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'2',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.187,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.187,0,0))

class MTA_power(part):
   #
   # AMP 1445121-2
   # MTA .050 SMT 2-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: Gnd
      #
      self.shape = translate(pad_MTA,-.025,-.1,0)
      self.pad.append(point(-.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: Vcc
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,.1,0))
      self.pad.append(point(.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Vcc',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.187,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.187,0,0))

class MTA_i0(part):
   #
   # AMP 1445121-3
   # MTA .050 SMT 3-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
```

```
      # pin 1: GND
      #
      self.shape = translate(pad_MTA,.05,.1,0)
      self.pad.append(point(.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1\nGND',14))
      #
      # pin 2: power
      #
      self.shape = add(self.shape,translate(pad_MTA,-.05,.1,0))
      self.pad.append(point(-.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V',14))
      #
      # pin 3: data
      #
      self.shape = add(self.shape,translate(pad_MTA,0,-.1,0))
      self.pad.append(point(0,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'data',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.212,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.212,0,0))

class MTA_4(part):
   #
   # AMP 1445121-4
   # MTA .050 SMT 4-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1
      #
      self.shape = translate(pad_MTA,-.075,-.1,0)
      self.pad.append(point(-.075,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,-.1,0))
      self.pad.append(point(.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'2',14))
      #
      # pin 3
      #
      self.shape = add(self.shape,translate(pad_MTA,.075,.1,0))
      self.pad.append(point(.075,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'3',14))
      #
      # pin 4
      #
      self.shape = add(self.shape,translate(pad_MTA,-.025,.1,0))
      self.pad.append(point(-.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'4',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.237,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.237,0,0))

class MTA_LCD_data(part):
   #
   # AMP 1445121-4
   # MTA .050 SMT 4-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1
      #
      self.shape = translate(pad_MTA,-.075,-.1,0)
      self.pad.append(point(-.075,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DB7\n14',14))
      #
      # pin 2
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,-.1,0))
      self.pad.append(point(.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DB5\n12',14))
      #
      # pin 3
      #
      self.shape = add(self.shape,translate(pad_MTA,.075,.1,0))
      self.pad.append(point(.075,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DB4\n11',14))
      #
      # pin 4
      #
      self.shape = add(self.shape,translate(pad_MTA,-.025,.1,0))
      self.pad.append(point(-.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DB6\n13',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.237,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.237,0,0))
```

```
class MTA_serial(part):
   #
   # AMP 1445121-4
   # MTA .050 SMT 4-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: Gnd
      #
      self.shape = translate(pad_MTA,-.075,-.1,0)
      self.pad.append(point(-.075,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: Tx
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,-.1,0))
      self.pad.append(point(.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Tx',14))
      #
      # pin 3: Rx
      #
      self.shape = add(self.shape,translate(pad_MTA,.075,.1,0))
      self.pad.append(point(.075,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Rx',14))
      #
      # pin 4: DTR
      #
      self.shape = add(self.shape,translate(pad_MTA,-.025,.1,0))
      self.pad.append(point(-.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DTR',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.237,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.237,0,0))

class MTA_5(part):
   #
   # AMP 1445121-5
   # MTA .050 SMT 5-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1
      #
      self.shape = translate(pad_MTA,-.1,-.1,0)
      self.pad.append(point(-.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2
      #
      self.shape = add(self.shape,translate(pad_MTA,0,-.1,0))
      self.pad.append(point(0,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'2',14))
      #
      # pin 3
      #
      self.shape = add(self.shape,translate(pad_MTA,.1,-.1,0))
      self.pad.append(point(.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'3',14))
      #
      # pin 4
      #
      self.shape = add(self.shape,translate(pad_MTA,.05,.1,0))
      self.pad.append(point(.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'4',14))
      #
      # pin 5
      #
      self.shape = add(self.shape,translate(pad_MTA,-.05,.1,0))
      self.pad.append(point(-.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'5',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.262,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.262,0,0))

class MTA_LCD_ctrl(part):
   #
   # AMP 1445121-5
   # MTA .050 SMT 5-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1
      #
      self.shape = translate(pad_MTA,-.1,-.1,0)
      self.pad.append(point(-.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'E\n6',14))
      #
      # pin 2
```

```
      #
      self.shape = add(self.shape,translate(pad_MTA,0,-.1,0))
      self.pad.append(point(0,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V0\n3',14))
      #
      # pin 3
      #
      self.shape = add(self.shape,translate(pad_MTA,.1,-.1,0))
      self.pad.append(point(.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND,R/W\n1,5',14))
      #
      # pin 4
      #
      self.shape = add(self.shape,translate(pad_MTA,.05,.1,0))
      self.pad.append(point(.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'VCC\n2',14))
      #
      # pin 5
      #
      self.shape = add(self.shape,translate(pad_MTA,-.05,.1,0))
      self.pad.append(point(-.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'RS\n4',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.262,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.262,0,0))

class MTA_ICP(part):
   #
   # AMP 1445121-5
   # MTA .050 SMT 4-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: MISO
      #
      self.shape = translate(pad_MTA,-.1,-.1,0)
      self.pad.append(point(-.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'MISO',14))
      #
      # pin 2: GND
      #
      self.shape = add(self.shape,translate(pad_MTA,0,-.1,0))
      self.pad.append(point(0,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))
      #
      # pin 3: MOSI
      #
      self.shape = add(self.shape,translate(pad_MTA,.1,-.1,0))
      self.pad.append(point(.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'MOSI',14))
      #
      # pin 4: -RESET
      #
      self.shape = add(self.shape,translate(pad_MTA,.05,.1,0))
      self.pad.append(point(.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'-RESET',14))
      #
      # pin 5: SCK
      #
      self.shape = add(self.shape,translate(pad_MTA,-.05,.1,0))
      self.pad.append(point(-.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'SCK',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.262,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.262,0,0))

class power_65mm(part):
   #
   # CUI PJ1-023-SMT
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: power
      #
      self.shape = cube(.433,.512,-.047,.047,0,0)
      self.pad.append(point(.467,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'P',14))
      #
      # pin 2: ground
      #
      self.shape = add(self.shape,cube(.285,.423,-.189,-.098,0,0))
      self.pad.append(point(.354,-.144,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 3: contact
      #
      self.shape = add(self.shape,cube(.325,.463,.098,.189,0,0))
      self.pad.append(point(.394,.144,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))
      #
      # solder pads
```

```
      #
      self.shape = add(self.shape,cube(.108,.246,-.169,-.110,0,0))
      self.shape = add(self.shape,cube(.069,.207,.110,.169,0,0))

pad_stereo_2_5mm = cube(-.03,.03,-.05,.05,0,0)

class stereo_2_5mm(part):
   #
   # CUI SJ1-2533-SMT
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: base
      #
      self.shape = translate(pad_stereo_2_5mm,-.130,-.16,0)
      self.pad.append(point(-.130,-.149,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'base',14))
      #
      # pin 2: tip
      #
      self.shape = add(self.shape,translate(pad_stereo_2_5mm,.197,.15,0))
      self.pad.append(point(.197,.141,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'tip',14))
      #
      # pin 3: middle
      #
      self.shape = add(self.shape,translate(pad_stereo_2_5mm,-.012,-.16,0))
      self.pad.append(point(-.012,-.149,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'middle',14))

pad_Molex = cube(-.0155,.0155,-.0265,.0265,0,0)
pad_Molex_solder = cube(-.055,.055,-.065,.065,0,0)

class Molex_serial(part):
   #
   # Molex 53261-0471
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: Rx
      #
      self.shape = translate(pad_Molex,-.075,.064,0)
      self.pad.append(point(-.075,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Rx',14))
      #
      # pin 2: Tx
      #
      self.shape = add(self.shape,translate(pad_Molex,-.025,.064,0))
      self.pad.append(point(-.025,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Tx',14))
      #
      # pin 3: DTR
      #
      self.shape = add(self.shape,translate(pad_Molex,.025,.064,0))
      self.pad.append(point(.025,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DTR',14))
      #
      # pin 4: GND
      #
      self.shape = add(self.shape,translate(pad_Molex,.075,.064,0))
      self.pad.append(point(.075,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_Molex_solder,-.16,-.065,0))
      self.shape = add(self.shape,translate(pad_Molex_solder,.16,-.065,0))

#
# switches
#

pad_button_6mm = cube(-.04,.04,-.03,.03,0,0)

class button_6mm(part):
   #
   # Omron 6mm pushbutton
   # B3SN-3112P
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # left 1
      #
      self.shape = translate(pad_button_6mm,-.125,.08,0)
      self.pad.append(point(-.125,.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'L1',14))
      #
      # right 1
      #
      self.shape = add(self.shape,translate(pad_button_6mm,-.125,-.08,0))
      self.pad.append(point(-.125,-.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'R1',14))
```

```
      #
      # right 2
      #
      self.shape = add(self.shape,translate(pad_button_6mm,.125,-.08,0))
      self.pad.append(point(.125,-.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'R2',14))
      #
      # left 2
      #
      self.shape = add(self.shape,translate(pad_button_6mm,.125,.08,0))
      self.pad.append(point(.125,.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'L2',14))

#
# crystals and resonators
#

pad_XTAL = cube(-.016,.016,-.077,.077,0,0)

class XTAL(part):
   #
   # Panasonic EFOBM series
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # left
      #
      self.shape = translate(pad_XTAL,-.053,0,0)
      self.pad.append(point(-.053,0,0))
      #
      # ground
      #
      self.shape = add(self.shape,translate(pad_XTAL,0,0,0))
      self.pad.append(point(0,0,0))
      #
      # right
      #
      self.shape = add(self.shape,translate(pad_XTAL,.053,0,0))
      self.pad.append(point(.053,0,0))

#
# diodes, transistors, regulators
#

class D_1206(part):
   #
   # 1206 diode
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # anode
      #
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A',14))
      #
      # cathode
      #
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))

class LED_1206(part):
   #
   # 1206 LED
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # anode
      #
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A',14))
      #
      # cathode
      #
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))

class phototransistor_1206(part):
   #
   # 1206 phototransistor
   # OPTEK 520,521
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # collector
      #
```

```
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))
      #
      # emitter
      #
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'E',14))

pad_SOD_123 = cube(-.02,.02,-.024,.024,0,0)

class D_SOD_123(part):
   #
   # SOD-123 diode
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # anode
      #
      self.shape = translate(pad_SOD_123,-.07,0,0)
      self.pad.append(point(-.07,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A',14))
      #
      # cathode
      #
      self.shape = add(self.shape,translate(pad_SOD_123,.07,0,0))
      self.pad.append(point(.07,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))

pad_SOT23 = cube(-.015,.015,-.02,.02,0,0)

class NMOSFET_SOT23(part):
   #
   # Fairchild NDS355AN
   #
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: gate
      #
      self.shape = translate(pad_SOT23,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 2: source
      #
      self.shape = add(self.shape,translate(pad_SOT23,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'S',14))
      #
      # pin 3: drain
      #
      self.shape = add(self.shape,translate(pad_SOT23,0,.045,0))
      self.pad.append(point(0,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'D',14))

class PMOSFET_SOT23(part):
   #
   # Fairchild NDS356AP
   #
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: gate
      #
      self.shape = translate(pad_SOT23,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 2: source
      #
      self.shape = add(self.shape,translate(pad_SOT23,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'S',14))
      #
      # pin 3: drain
      #
      self.shape = add(self.shape,translate(pad_SOT23,0,.045,0))
      self.pad.append(point(0,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'D',14))

class regulator_SOT23(part):
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
```

```
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: output
      #
      self.shape = translate(pad_SOT23,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: input
      #
      self.shape = add(self.shape,translate(pad_SOT23,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'in',14))
      #
      # pin 3: ground
      #
      self.shape = add(self.shape,translate(pad_SOT23,0,.045,0))
      self.pad.append(point(0,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'gnd',14))

pad_SOT223 = cube(-.02,.02,-.03,.03,0,0)
pad_SOT223_ground = cube(-.065,.065,-.03,.03,0,0)

class regulator_SOT223(part):
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: input
      #
      self.shape = translate(pad_SOT223,-.09,-.12,0)
      self.pad.append(point(-.09,-.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1I',14))
      #
      # pin 2: ground
      #
      self.shape = add(self.shape,translate(pad_SOT223,0,-.12,0))
      self.pad.append(point(0,-.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 3: output
      #
      self.shape = add(self.shape,translate(pad_SOT223,.09,-.12,0))
      self.pad.append(point(.09,-.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'O',14))
      #
      # pin 4: ground
      #
      self.shape = add(self.shape,translate(pad_SOT223_ground,0,.12,0))
      self.pad.append(point(0,.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))

pad_SM8 = cube(-.035,.035,-.016,.016,0,0)

class H_bridge_SM8(part):
   #
   # Zetex ZXMHC3A01T8
   #
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      d = .13
      #
      # pin 1: G3 (right N gate)
      #
      self.shape = translate(pad_SM8,-d,.09,0)
      self.pad.append(point(-d,.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GRN',14))
      #
      # pin 2: S2 S3 (N source)
      #
      self.shape = add(self.shape,translate(pad_SM8,-d,.03,0))
      self.pad.append(point(-d,.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'SN',14))
      #
      # pin 3: G2 (left N gate)
      #
      self.shape = add(self.shape,translate(pad_SM8,-d,-.03,0))
      self.pad.append(point(-d,-.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GLN',14))
      #
      # pin 4: G1 (left P gate)
      #
      self.shape = add(self.shape,translate(pad_SM8,-d,-.09,0))
      self.pad.append(point(-d,-.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GLP',14))
      #
      # pin 5: D1 D2 (left drain)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,-.09,0))
      self.pad.append(point(d,-.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DL',14))
```

```
      #
      # pin 6: S1 S4 (P source)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,-.03,0))
      self.pad.append(point(d,-.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'SP',14))
      #
      # pin 7: D3 D4 (right drain)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,.03,0))
      self.pad.append(point(d,.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DR',14))
      #
      # pin 8: G4 (right N gate)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,.09,0))
      self.pad.append(point(d,.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GRP',14))

#
# ICs
#

pad_SOT23_5 = cube(-.01,.01,-.02,.02,0,0)

class op_amp_SOT23_5(part):
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: output
      #
      self.shape = translate(pad_SOT23_5,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'O',14))
      #
      # pin 2: V-
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,0,-.045,0))
      self.pad.append(point(0,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V-',14))
      #
      # pin 3: I+
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'I+',14))
      #
      # pin 4: I-
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,.0375,.045,0))
      self.pad.append(point(.0375,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'I-',14))
      #
      # pin 5: V+
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,-.0375,.045,0))
      self.pad.append(point(-.0375,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V+',14))

pad_SOICN = cube(-.035,.035,-.015,.015,0,0)

class op_amp_SOICN(part):
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: A out
      #
      self.shape = translate(pad_SOICN,-.12,.075,0)
      self.pad.append(point(-.12,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1 Ao',14))
      #
      # pin 2: A-
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.025,0))
      self.pad.append(point(-.12,.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A-',14))
      #
      # pin 3: A+
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.025,0))
      self.pad.append(point(-.12,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A+',14))
      #
      # pin 4: V-
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.075,0))
      self.pad.append(point(-.12,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V-',14))
      #
      # pin 5: B+
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.075,0))
      self.pad.append(point(.12,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'B+',14))
```

```
      #
      # pin 6: B-
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.025,0))
      self.pad.append(point(.12,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'B-',14))
      #
      # pin 7: B out
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.025,0))
      self.pad.append(point(.12,.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Bo',14))
      #
      # pin 8: V+
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.075,0))
      self.pad.append(point(.12,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V+',14))

pad_SOIC = cube(-.043,.043,-.015,.015,0,0)

class ATtiny45_SOIC(part):
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: PB5/dW/ADC0/-RESET/PCINT5
      #
      self.shape = translate(pad_SOIC,-.14,.075,0)
      self.pad.append(point(-.14,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: PB3/ADC3/-OC1B/CLKI/XTAL1/PCINT3
      #
      self.shape = add(self.shape,translate(pad_SOIC,-.14,.025,0))
      self.pad.append(point(-.14,.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB3',14))
      #
      # pin 3: PB4/ADC2/OC1B/CLKO/XTAL2/PCINT4
      #
      self.shape = add(self.shape,translate(pad_SOIC,-.14,-.025,0))
      self.pad.append(point(-.14,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB4',14))
      #
      # pin 4: GND
      #
      self.shape = add(self.shape,translate(pad_SOIC,-.14,-.075,0))
      self.pad.append(point(-.14,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))
      #
      # pin 5: PB0/MOSI/DI/SDA/AIN0/OC0A/-OC1A/AREF/PCINT0
      #
      self.shape = add(self.shape,translate(pad_SOIC,.14,-.075,0))
      self.pad.append(point(.14,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 6: PB1/MISO/DO/AIN1/OC0B/OC1A/PCINT1
      #
      self.shape = add(self.shape,translate(pad_SOIC,.14,-.025,0))
      self.pad.append(point(.14,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB1',14))
      #
      # pin 7: PB2/SCK/USCK/SCL/ADC1/T0/INT0/PCINT2
      #
      self.shape = add(self.shape,translate(pad_SOIC,.14,.025,0))
      self.pad.append(point(.14,.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB2',14))
      #
      # pin 8: VCC
      #
      self.shape = add(self.shape,translate(pad_SOIC,.14,.075,0))
      self.pad.append(point(.14,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'VCC',14))

class ATtiny44_SOICN(part):
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: VCC
      #
      self.shape = translate(pad_SOICN,-.12,.15,0)
      self.pad.append(point(-.12,.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: PB0/XTAL1/PCINT8
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 3: PB1/XTAL2/PCINT9
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.050,0))
      self.pad.append(point(-.12,.05,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB1',14))
      #
      # pin 4: PB3/dW/-RESET/PCINT11
      #
```

```
      self.shape = add(self.shape,translate(pad_SOICN,-.12,0,0))
      self.pad.append(point(-.12,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB3',14))
      #
      # pin 5: PB2/CKOUT/OC0A/INT0/PCINT10
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.05,0))
      self.pad.append(point(-.12,-.05,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB2',14))
      #
      # pin 6: PA7/ADC7/OC0B/ICP/PCINT7
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.1,0))
      self.pad.append(point(-.12,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA7',14))
      #
      # pin 7: PA6/ADC6/MOSI/SDA/OC1A/PCINT6
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.15,0))
      self.pad.append(point(-.12,-.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA6',14))
      #
      # pin 8: PA5/ADC5/DO/MISO/OC1B/PCINT5
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.15,0))
      self.pad.append(point(.12,-.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA5',14))
      #
      # pin 9: PA4/ADC4/USCK/SCL/T1/PCINT4
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.1,0))
      self.pad.append(point(.12,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA4',14))
      #
      # pin 10: PA3/ADC3/T0/PCINT3
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.05,0))
      self.pad.append(point(.12,-.05,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA3',14))
      #
      # pin 11: PA2/ADC2/AIN1/PCINT2
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,0,0))
      self.pad.append(point(.12,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA2',14))
      #
      # pin 12: PA1/ADC1/AIN0/PCINT1
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.050,0))
      self.pad.append(point(.12,.05,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA1',14))
      #
      # pin 13: PA0/ADC0/AREF/PCINT0
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.1,0))
      self.pad.append(point(.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA0',14))
      #
      # pin 14: GND
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.15,0))
      self.pad.append(point(.12,.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))

pad_TQFP = cube(-.043,.043,-.015,.015,0,0)

class ATmega88_TQFP(part):
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []

      #
      # pin 1: PD3/PCINT19/OC2B/INT1
      #
      self.shape = translate(pad_SOICN,-.12,.15,0)
      self.pad.append(point(-.12,.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: PD4/PCINT20/XCK/T0
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 3: GND
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 4: VCC
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 5: GND
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
```

```
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 6: VCC
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 7: PB6/PCINT6/XTAL1/TOSC1
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 8: PB7/PCINT7/XTAL2/TOSC2
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

#
# pin 9: PD5/PCINT21/OC0B/T1
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 10: PD6/PCINT22/OC0A/AIN0
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 11: PD7/PCINT23/AIN1
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 12: PB0/PCINT0/CLKO/ICP1
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 13: PB1/PCINT1/OC1A
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 14: PB2/PCINT2/-SS/OC1B
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 15: PB3/PCINT3/OC2A/MOSI
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 16: PB4/PCINT4/MISO
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

#
# pin 17: PB5/SCK/PCINT5
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 18: AVCC
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 19: ADC6
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 20: AREF
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 21: GND
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 22: ADC7
```

```
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 23: PC0/ADC0/PCINT8
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 24: PC1/ADC1/PCINT9
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

      #
      # pin 25: PC2/ADC2/PCINT10
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 26: PC2/ADC3/PCINT11
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 27: PC4/ADC4/SDA/PCINT12
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 28: PC5/ADC5/SCL/PCINT13
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 29: PC6/-RESET/PCINT14
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 30: PD0/RXD/PCINT16
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 31: PD1/TXD/PCINT17
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 32: PD2/INT0/PCINT18
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

#
# graphics
#

class CBA(part):
   def __init__(self,r=.02):
      self.value = ''
      self.pad = [point(0,0,0)]
      self.labels = []
      d = 3*r
      self.shape = circle(0,0,r)
      self.shape = add(self.shape,translate(circle(0,0,r),-d,d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),-d,0,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),-d,-d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),0,-d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),d,-d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),d,0,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),d,d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),0,d,0))

class hello(part):
   #
   # HELLO
   #
   def __init__(self,w,z):
      self.value = ''
      self.pad = [point(0,0,0)]
      self.labels = []
      # H
      self.shape = rectangle(-.01,.01,0,.1)
      self.shape = add(self.shape,rectangle(0,.05,.04,.06))
      self.shape = add(self.shape,rectangle(.04,.06,0,.1))
      # E
      self.shape = add(self.shape,rectangle(.08,.1,0,.1))
      self.shape = add(self.shape,rectangle(.1,.14,0,.02))
```

```
        self.shape = add(self.shape,rectangle(.1,.13,.04,.06))
        self.shape = add(self.shape,rectangle(.1,.14,.08,.1))
        # L
        self.shape = add(self.shape,rectangle(.16,.18,0,.1))
        self.shape = add(self.shape,rectangle(.18,.21,0,.02))
        # L
        self.shape = add(self.shape,rectangle(.23,.25,0,.1))
        self.shape = add(self.shape,rectangle(.25,.28,0,.02))
        # 0
        self.shape = add(self.shape,rectangle(.3,.32,0,.1))
        self.shape = add(self.shape,rectangle(.32,.345,0,.02))
        self.shape = add(self.shape,rectangle(.32,.345,.08,.1))
        self.shape = add(self.shape,rectangle(.345,.365,0,.1))

#
# define board
#

w = .016
width = 1.02
height = 1.18
x = 1
y = 1
z = -.005
d = .06
mask = .004

pcb = PCB(x,y,width,height,mask)

IC1 = ATtiny44_SOICN('IC1\nt44')
pcb = IC1.add(pcb,x+.35,y+.67,z)

J1 = MTA_ICP('J1\nICP')
pcb = J1.add(pcb,IC1.x,IC1.pad[7].y-.19,z,angle=180)

pcb = wire(pcb,w,
   IC1.pad[7],
   J1.pad[3])

pcb = wire(pcb,w,
   IC1.pad[4],
   J1.pad[4])

pcb = wire(pcb,w,
   IC1.pad[14],
   J1.pad[2])

pcb = wire(pcb,w,
   IC1.pad[9],
   J1.pad[5])

pcb = wire(pcb,w,
   IC1.pad[8],
   J1.pad[1])

J2 = MTA_power('J2\npower')
pcb = J2.add(pcb,IC1.x-.03,IC1.pad[1].y+.2,z,angle=0)

pcb = wire(pcb,w,
   IC1.pad[14],
   J2.pad[1])

IC2 = regulator_SOT23('IC2\n5V')
pcb = IC2.add(pcb,IC1.x-.28,IC1.pad[1].y,z,angle=180)

pcb = wire(pcb,w,
   IC1.pad[1],
   IC2.pad[1])

pcb = wire(pcb,w,
   J2.pad[2],
   IC2.pad[2])

pcb = wire(pcb,.014,
   J2.pad[1],
   IC2.pad[3])

C1 = C_1206('C1\n3.3uF')
pcb = C1.add(pcb,IC2.x,IC2.y-.22,z,angle=90)

pcb = wire(pcb,w,
   C1.pad[1],
   IC2.pad[3])

pcb = wire(pcb,w,
   C1.pad[2],
   point(C1.x+.07,IC1.pad[1].y,z),
   IC1.pad[1])

R1 = R_1206('R1\n10k')
pcb = R1.add(pcb,J1.x-.22,J1.y-.21,z)

pcb = wire(pcb,w,
   C1.pad[2],
   point(C1.x,J1.pad[3].y,z),
   point(J1.pad[3].x-.05,J1.pad[4].y+.02,z),
   R1.pad[1])

pcb = wire(pcb,w,
   point(J1.pad[4].x,J1.pad[4].y-.02,z),
   R1.pad[2])
```

```
J3 = MTA_LCD_ctrl('J3\nLCD\nctrl')
pcb = J3.add(pcb,x+width-.16,y+.35,z,angle=90)

pcb = wire(pcb,w,
   J3.pad[1],
   point(IC1.pad[8].x+.1,IC1.pad[8].y,z),
   IC1.pad[8])

pcb = wire(pcb,w,
   J3.pad[5],
   point(J3.pad[5].x,J3.pad[1].y+.05,z),
   point(IC1.pad[9].x+.14,IC1.pad[9].y,z),
   IC1.pad[9])

J4 = MTA_LCD_data('J4\nLCD\ndata')
pcb = J4.add(pcb,J3.x-.13,J3.y+.5,z,angle=90)

pcb = wire(pcb,w,
   IC1.pad[13],
   point(IC1.pad[13].x+.08,J4.pad[1].y,z),
   J4.pad[1])

pcb = wire(pcb,w,
   IC1.pad[12],
   point(J4.pad[2].x-.03,J4.pad[2].y,z))

pcb = wire(pcb,w,
   IC1.pad[10],
   point(J4.pad[2].x+.03,J4.pad[3].y-.05,z),
   J4.pad[3])

pcb = wire(pcb,w,
   IC1.pad[11],
   point(J4.pad[2].x,J4.pad[3].y,z),
   point(J4.x,J4.pad[4].y,z),
   J4.pad[4])

R2 = R_1206('R2\n100k')
pcb = R2.add(pcb,J1.x,R1.y,z)

pcb = wire(pcb,w,
   R2.pad[1],
   point(R2.pad[1].x,R2.y-.095,z),
   R1.pad[1])

pcb = wire(pcb,w,
   R2.pad[1],
   point(R2.pad[1].x,R2.y-.095,z),
   point(J3.pad[3].x,J3.pad[3].y-.05,z),
   J3.pad[4])

R3 = R_1206('R3\n1k')
pcb = R3.add(pcb,J1.x+.22,R1.y,z)

pcb = wire(pcb,w,
   R3.pad[1],
   R2.pad[2])

pcb = wire(pcb,w,
   R3.pad[1],
   point(J1.pad[1].x+.05,J3.pad[5].y,z),
   J3.pad[2])

pcb = wire(pcb,w,
   point(J3.pad[3].x,J3.pad[3].y-.01,z),
   R3.pad[2])

pcb = wire(pcb,w,
   J1.pad[2],
   point(J1.pad[2].x,R2.y-.06,z),
   R3.pad[2])

H1 = hello(w,z)
pcb = H1.add(pcb,x+width-.02,y+height-.4,z,angle=-90)

cad.function = add(pcb.board,pcb.exterior)

#
# uncomment to export traces
#

#cad.function = pcb.board

#
# uncomment to export exterior
#

#cad.function = pcb.exterior

#
# uncomment to export solder mask
#

#cad.function = pcb.mask

#
# uncomment to mill out board
#
# use 1/32 end-mill
# set tool diameter = .0312
```

```
# set xy, z speed = .5
# set # contours = 1
#

#cad.function = pcb.interior
#z = -.065

#
# define limits and parameters
#

cad.xmin = x-.1 # min x to render
cad.xmax = x+width+.1 # max x to render
cad.ymin = y-.1 # min y to render
cad.ymax = y+height+.1 # max y to render
cad.zmin = z
cad.zmax = .050
dpi = 200 # low resolution for previewing
#dpi = 500 # high resolution for machining
nxy = int(dpi*(cad.xmax-cad.xmin))
cad.nx = nxy # x points to render
cad.ny = nxy # y points to render
cad.nz = 1
cad.view('xy') # 2D view
```

```
#
# hello.stepper.44.cad
#
# stepper motor hello-world
#
# Neil Gershenfeld
# CBA MIT 10/18/07
#
# (c) Massachusetts Institute of Technology 2007
# Permission granted for experimental and personal use;
# license for commercial sale available from MIT.
#

#
# define shapes and transformation
#
# circle(x0, y0, r)
# cylinder(x0, y0, z0, z1, r)
# cone(x0, y0, z0, z1, r0)
# sphere(x0, y0, z0, r)
# torus(x0, y0, z0, r0, r1)
# rectangle(x0, x1, y0, y1)
# cube(x0, x1, y0, y1, z0, z1)
# triangle(x0, y0, x1, y1, x2, y2) (points in clockwise order)
# pyramid(x0, x1, y0, y1, z0, z1)
# function(Z_of_XY)
# functions(upper_Z_of_XY,lower_Z_of_XY)
# add(part1, part2)
# subtract(part1, part2)
# intersect(part1, part2)
# move(part,dx,dy)
# translate(part,dx,dy,dz)
# rotate(part, angle)
# rotate_x(part, angle)
# rotate_y(part, angle)
# rotate_z(part, angle)
# rotate_90(part)
# rotate_180(part)
# rotate_270(part)
# reflect_x(part)
# reflect_y(part)
# reflect_z(part)
# reflect_xy(part)
# reflect_xz(part)
# reflect_yz(part)
# scale_x(part, x0, sx)
# scale_y(part, y0, sy)
# scale_z(part, z0, sz)
# scale_xy(part, x0, y0, sxy)
# scale_xyz(part, x0, y0, z0, sxyz)
# coscale_x_y(part, x0, y0, y1, angle0, angle1, amplitude, offset)
# coscale_x_z(part, x0, z0, z1, angle0, angle1, amplitude, offset)
# coscale_xy_z(part, x0, y0, z0, z1, angle0, angle1, amplitude, offset)
# taper_x_y(part, x0, y0, y1, s0, s1)
# taper_x_z(part, x0, z0, z1, s0, s1)
# taper_xy_z(part, x0, y0, z0, z1, s0, s1)
# shear_x_y(part, y0, y1, dx0, dx1)
# shear_x_z(part, z0, z1, dx0, dx1)
# (more to come)

def circle(x0, y0, r):
   part = "(((X-x0)**2 + (Y-y0)**2) <= r**2)"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'r',str(r))
   return part

def cylinder(x0, y0, z0, z1, r):
   part = "(((X-x0)**2 + (Y-y0)**2 <= r**2) & (Z >= z0) & (Z <= z1))"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'r',str(r))
   return part

def cone(x0, y0, z0, z1, r0):
   part = cylinder(x0, y0, z0, z1, r0)
   part = taper_xy_z(part, x0, y0, z0, z1, 1.0, 0.0)
   return part

def sphere(x0, y0, z0, r):
   part = "(((X-x0)**2 + (Y-y0)**2 + (Z-z0)**2) <= r**2)"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'r',str(r))
   return part

def torus(x0, y0, z0, r0, r1):
   part = "(((r0 - sqrt((X-x0)**2 + (Y-y0)**2))**2 + (Z-z0)**2) <= r1**2)"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'r0',str(r0))
   part = replace(part,'r1',str(r1))
   return part

def rectangle(x0, x1, y0, y1):
   part = "((X >= x0) & (X <= x1) & (Y >= y0) & (Y <= y1))"
   part = replace(part,'x0',str(x0))
```

```
   part = replace(part,'x1',str(x1))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   return part

def cube(x0, x1, y0, y1, z0, z1):
   part = "((X >= x0) & (X <= x1) & (Y >= y0) & (Y <= y1) & (Z >= z0) & (Z <= z1))"
   part = replace(part,'x0',str(x0))
   part = replace(part,'x1',str(x1))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   return part

def triangle(x0, y0, x1, y1, x2, y2): # points in clockwise order
   part = "((((y1-y0)*(X-x0)-(x1-x0)*(Y-y0)) >= 0) & (((y2-y1)*(X-x1)-(x2-x1)*(Y-y1)) >= 0) & (((y0-y2)*(X-x2)-(x0-x2)*(Y-y2)) >= 0))"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'x1',str(x1))
   part = replace(part,'y1',str(y1))
   part = replace(part,'x2',str(x2))
   part = replace(part,'y2',str(y2))
   return part

def pyramid(x0, x1, y0, y1, z0, z1):
   part = cube(x0, x1, y0, y1, z0, z1)
   part = taper_xy_z(part, (x0+x1)/2., (y0+y1)/2., z0, z1, 1.0, 0.0)
   return part

def function(Z_of_XY):
   part = '(Z <= '+Z_of_XY+')'
   return part

def functions(upper_Z_of_XY,lower_Z_of_XY):
   part = '(Z <= '+upper_Z_of_XY+') & (Z >= '+lower_Z_of_XY+')'
   return part

def add(part1, part2):
   part = "part1 | part2"
   part = replace(part,'part1',part1)
   part = replace(part,'part2',part2)
   return part

def subtract(part1, part2):
   part = "(part1) & ~(part2)"
   part = replace(part,'part1',part1)
   part = replace(part,'part2',part2)
   return part

def intersect(part1, part2):
   part = "(part1) & (part2)"
   part = replace(part,'part1',part1)
   part = replace(part,'part2',part2)
   return part

def move(part,dx,dy):
   part = replace(part,'X','(X-'+str(dx)+')')
   part = replace(part,'Y','(Y-'+str(dy)+')')
   return part

def translate(part,dx,dy,dz):
   part = replace(part,'X','(X-'+str(dx)+')')
   part = replace(part,'Y','(Y-'+str(dy)+')')
   part = replace(part,'Z','(Z-'+str(dz)+')')
   return part

def rotate(part, angle):
   angle = angle*pi/180
   part = replace(part,'X','(cos(angle)*X+sin(angle)*y)')
   part = replace(part,'Y','(-sin(angle)*X+cos(angle)*y)')
   part = replace(part,'y','Y')
   part = replace(part,'angle',str(angle))
   return part

def rotate_x(part, angle):
   angle = angle*pi/180
   part = replace(part,'Y','(cos(angle)*Y+sin(angle)*z)')
   part = replace(part,'Z','(-sin(angle)*Y+cos(angle)*z)')
   part = replace(part,'z','Z')
   part = replace(part,'angle',str(angle))
   return part

def rotate_y(part, angle):
   angle = angle*pi/180
   part = replace(part,'X','(cos(angle)*X+sin(angle)*z)')
   part = replace(part,'Z','(-sin(angle)*X+cos(angle)*z)')
   part = replace(part,'z','Z')
   part = replace(part,'angle',str(angle))
   return part

def rotate_z(part, angle):
   angle = angle*pi/180
   part = replace(part,'X','(cos(angle)*X+sin(angle)*y)')
   part = replace(part,'Y','(-sin(angle)*X+cos(angle)*y)')
   part = replace(part,'y','Y')
   part = replace(part,'angle',str(angle))
   return part

def rotate_90(part):
   part = reflect_xy(part)
```

```
   part = reflect_y(part)
   return part

def rotate_180(part):
   part = reflect_xy(part)
   part = reflect_y(part)
   part = reflect_xy(part)
   part = reflect_y(part)
   return part

def rotate_270(part):
   part = reflect_xy(part)
   part = reflect_y(part)
   part = reflect_xy(part)
   part = reflect_y(part)
   part = reflect_xy(part)
   part = reflect_y(part)
   return part

def reflect_x(part):
   part = replace(part,'X','-X')
   return part

def reflect_y(part):
   part = replace(part,'Y','-Y')
   return part

def reflect_z(part):
   part = replace(part,'Z','-Z')
   return part

def reflect_xy(part):
   part = replace(part,'X','temp')
   part = replace(part,'Y','X')
   part = replace(part,'temp','Y')
   return part

def reflect_xz(part):
   part = replace(part,'X','temp')
   part = replace(part,'Z','X')
   part = replace(part,'temp','Z')
   return part

def reflect_yz(part):
   part = replace(part,'Y','temp')
   part = replace(part,'Z','Y')
   part = replace(part,'temp','Z')
   return part

def scale_x(part, x0, sx):
   part = replace(part,'X','(x0 + (X-x0)/sx)')
   part = replace(part,'x0',str(x0))
   part = replace(part,'sx',str(sx))
   return part

def scale_y(part, y0, sy):
   part = replace(part,'Y','(y0 + (Y-y0)/sy)')
   part = replace(part,'y0',str(y0))
   part = replace(part,'sy',str(sy))
   return part

def scale_z(part, z0, sz):
   part = replace(part,'Z','(z0 + (Z-z0)/sz)')
   part = replace(part,'z0',str(z0))
   part = replace(part,'sz',str(sz))
   return part

def scale_xy(part, x0, y0, sxy):
   part = replace(part,'X','(x0 + (X-x0)/sxy)')
   part = replace(part,'Y','(y0 + (Y-y0)/sxy)')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'sxy',str(sxy))
   return part

def scale_xyz(part, x0, y0, z0, sxyz):
   part = replace(part,'X','(x0 + (X-x0)/sxyz)')
   part = replace(part,'Y','(y0 + (Y-y0)/sxyz)')
   part = replace(part,'Z','(z0 + (Z-z0)/sxyz)')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'sxyz',str(sxyz))
   return part

def coscale_x_y(part, x0, y0, y1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
   part = replace(part,'X','(x0 + (X-x0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Y-y0)/(y1-y0))))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

def coscale_x_z(part, x0, z0, z1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
```

```
   part = replace(part,'X','(x0 + (X-x0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0))))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

def coscale_xy_z(part, x0, y0, z0, z1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
   part = replace(part,'X','(x0 + (X-x0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0))))')
   part = replace(part,'Y','(y0 + (Y-y0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0))))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

def taper_x_y(part, x0, y0, y1, s0, s1):
   part = replace(part,'X','(x0 + (X-x0)*(y1-y0)/(s1*(Y-y0) + s0*(y1-Y)))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'s0',str(s0))
   part = replace(part,'s1',str(s1))
   return part

def taper_x_z(part, x0, z0, z1, s0, s1):
   part = replace(part,'X','(x0 + (X-x0)*(z1-z0)/(s1*(Z-z0) + s0*(z1-Z)))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'s0',str(s0))
   part = replace(part,'s1',str(s1))
   return part

def taper_xy_z(part, x0, y0, z0, z1, s0, s1):
   part = replace(part,'X','(x0 + (X-x0)*(z1-z0)/(s1*(Z-z0) + s0*(z1-Z)))')
   part = replace(part,'Y','(y0 + (Y-y0)*(z1-z0)/(s1*(Z-z0) + s0*(z1-Z)))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'s0',str(s0))
   part = replace(part,'s1',str(s1))
   return part

def shear_x_y(part, y0, y1, dx0, dx1):
   part = replace(part,'X','(X - dx0 - (dx1-dx0)*(Y-y0)/(y1-y0))')
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'dx0',str(dx0))
   part = replace(part,'dx1',str(dx1))
   return part

def shear_x_z(part, z0, z1, dx0, dx1):
   part = replace(part,'X','(X - dx0 - (dx1-dx0)*(Z-z0)/(z1-z0))')
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'dx0',str(dx0))
   part = replace(part,'dx1',str(dx1))
   return part

def coshear_x_z(part, z0, z1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
   part = replace(part,'X','(X - offset - amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0)))')
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

#
# PCB classes and definitions
#

cad.labels = []

class PCB:
   def __init__(self,x0,y0,width,height,mask):
      self.board = "False"
      self.interior = rectangle(x0,x0+width,y0,y0+height)
      self.exterior = subtract("True",rectangle(x0,x0+width,y0,y0+height))
      self.mask = "False"
   def add(self,part):
      self.board = add(self.board,part)
      self.mask = add(self.mask,move(part,-mask,mask))
      self.mask = add(self.mask,move(part,-mask,-mask))
      self.mask = add(self.mask,move(part,mask,mask))
      self.mask = add(self.mask,move(part,mask,-mask))
      return self
```

```
class part:
   def add(self,pcb,x,y,z=0,angle=0):
      self.x = x
      self.y = y
      self.z = z
      self.angle = angle
      if (angle == 90):
         self.shape = rotate_90(self.shape)
      elif (angle == 180):
         self.shape = rotate_180(self.shape)
      elif ((angle == 270) | (angle == -90)):
         self.shape = rotate_270(self.shape)
      angle = pi*angle/180
      self.shape = translate(self.shape,x,y,z)
      for i in range(len(self.pad)):
         xnew = cos(angle)*self.pad[i].x + sin(angle)*self.pad[i].y
         ynew = -sin(angle)*self.pad[i].x + cos(angle)*self.pad[i].y
           self.pad[i].x = x + xnew
           self.pad[i].y = y + ynew
         self.pad[i].z += z
      cad.labels.append(cad_text(x,y,z,self.value,size=14))
      for i in range(len(self.labels)):
         xnew = cos(angle)*self.labels[i].x + sin(angle)*self.labels[i].y
         ynew = -sin(angle)*self.labels[i].x + cos(angle)*self.labels[i].y
           self.labels[i].x = x + xnew
           self.labels[i].y = y + ynew
         self.labels[i].z += z
           cad.labels.append(self.labels[i])
      pcb = pcb.add(self.shape)
      return pcb

def wire(pcb,width,*points):
   for i in range(1,len(points)):
      x0 = points[i-1].x
      y0 = points[i-1].y
      z0 = points[i-1].z
      x1 = points[i].x
      y1 = points[i].y
      z1 = points[i].z
      if (x0 < x1):
         pcb.board = add(pcb.board,cube(x0-width/2,x1+width/2,y0-width/2,y0+width/2,z0,z0))
      elif (x1 < x0):
         pcb.board = add(pcb.board,cube(x1-width/2,x0+width/2,y0-width/2,y0+width/2,z0,z0))
      if (y0 < y1):
         pcb.board = add(pcb.board,cube(x1-width/2,x1+width/2,y0-width/2,y1+width/2,z0,z0))
      elif (y1 < y0):
         pcb.board = add(pcb.board,cube(x1-width/2,x1+width/2,y1-width/2,y0+width/2,z0,z0))
   return pcb

#
# PCB library
#

#
# discretes
#

pad_0402 = cube(-.0175,.0175,-.014,.014,0,0)

class R_0402(part):
   #
   # 0402 resistor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_0402,-.0265,0,0)
      self.pad.append(point(-.0265,0,0))
      self.shape = add(self.shape,translate(pad_0402,.0265,0,0))
      self.pad.append(point(.0265,0,0))

pad_1206 = cube(-.032,.032,-.034,.034,0,0)

class R_1206(part):
   #
   # 1206 resistor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.06,0,0))
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.06,0,0))

class C_1206(part):
   #
   # 1206 capacitor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.06,0,0))
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.06,0,0))
```

```
pad_1210 = cube(-.032,.032,-.048,.048,0,0)

class L_1210(part):
   #
   # 1210 inductor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_1210,-.06,0,0)
      self.pad.append(point(-.06,0,0))
      self.shape = add(self.shape,translate(pad_1210,.06,0,0))
      self.pad.append(point(.06,0,0))

pad_choke = cube(-.06,.06,-.06,.06,0,0)

class choke(part):
   #
   # Panasonic ELLCTV
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_choke,-.177,-.177,0)
      self.pad.append(point(-.177,-.177,0))
      self.shape = add(self.shape,translate(pad_choke,.177,.177,0))
      self.pad.append(point(.177,.177,0))

#
# connectors
#

pad_MTA = cube(-.021,.021,-.041,.041,0,0)
pad_MTA_solder = cube(-.071,.071,-.041,.041,0,0)

class MTA_2(part):
   #
   # AMP 1445121-2
   # MTA .050 SMT 2-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1
      #
      self.shape = translate(pad_MTA,-.025,-.1,0)
      self.pad.append(point(-.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,.1,0))
      self.pad.append(point(.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'2',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.187,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.187,0,0))

class MTA_power(part):
   #
   # AMP 1445121-2
   # MTA .050 SMT 2-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: Gnd
      #
      self.shape = translate(pad_MTA,-.025,-.1,0)
      self.pad.append(point(-.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: Vcc
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,.1,0))
      self.pad.append(point(.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Vcc',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.187,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.187,0,0))

class MTA_i0(part):
   #
   # AMP 1445121-3
   # MTA .050 SMT 3-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
```

```
      # pin 1: GND
      #
      self.shape = translate(pad_MTA,.05,.1,0)
      self.pad.append(point(.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1\nGND',14))
      #
      # pin 2: power
      #
      self.shape = add(self.shape,translate(pad_MTA,-.05,.1,0))
      self.pad.append(point(-.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V',14))
      #
      # pin 3: data
      #
      self.shape = add(self.shape,translate(pad_MTA,0,-.1,0))
      self.pad.append(point(0,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'data',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.212,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.212,0,0))

class MTA_4(part):
   #
   # AMP 1445121-4
   # MTA .050 SMT 4-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1
      #
      self.shape = translate(pad_MTA,-.075,-.1,0)
      self.pad.append(point(-.075,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,-.1,0))
      self.pad.append(point(.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'2',14))
      #
      # pin 3
      #
      self.shape = add(self.shape,translate(pad_MTA,.075,.1,0))
      self.pad.append(point(.075,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'3',14))
      #
      # pin 4
      #
      self.shape = add(self.shape,translate(pad_MTA,-.025,.1,0))
      self.pad.append(point(-.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'4',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.237,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.237,0,0))

class MTA_LCD_data(part):
   #
   # AMP 1445121-4
   # MTA .050 SMT 4-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1
      #
      self.shape = translate(pad_MTA,-.075,-.1,0)
      self.pad.append(point(-.075,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DB7\n14',14))
      #
      # pin 2
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,-.1,0))
      self.pad.append(point(.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DB5\n12',14))
      #
      # pin 3
      #
      self.shape = add(self.shape,translate(pad_MTA,.075,.1,0))
      self.pad.append(point(.075,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DB4\n11',14))
      #
      # pin 4
      #
      self.shape = add(self.shape,translate(pad_MTA,-.025,.1,0))
      self.pad.append(point(-.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DB6\n13',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.237,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.237,0,0))
```

```
class MTA_serial(part):
   #
   # AMP 1445121-4
   # MTA .050 SMT 4-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: Gnd
      #
      self.shape = translate(pad_MTA,-.075,-.1,0)
      self.pad.append(point(-.075,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: Tx
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,-.1,0))
      self.pad.append(point(.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Tx',14))
      #
      # pin 3: Rx
      #
      self.shape = add(self.shape,translate(pad_MTA,.075,.1,0))
      self.pad.append(point(.075,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Rx',14))
      #
      # pin 4: DTR
      #
      self.shape = add(self.shape,translate(pad_MTA,-.025,.1,0))
      self.pad.append(point(-.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DTR',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.237,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.237,0,0))

class MTA_5(part):
   #
   # AMP 1445121-5
   # MTA .050 SMT 5-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1
      #
      self.shape = translate(pad_MTA,-.1,-.1,0)
      self.pad.append(point(-.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2
      #
      self.shape = add(self.shape,translate(pad_MTA,0,-.1,0))
      self.pad.append(point(0,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'2',14))
      #
      # pin 3
      #
      self.shape = add(self.shape,translate(pad_MTA,.1,-.1,0))
      self.pad.append(point(.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'3',14))
      #
      # pin 4
      #
      self.shape = add(self.shape,translate(pad_MTA,.05,.1,0))
      self.pad.append(point(.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'4',14))
      #
      # pin 5
      #
      self.shape = add(self.shape,translate(pad_MTA,-.05,.1,0))
      self.pad.append(point(-.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'5',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.262,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.262,0,0))

class MTA_stepper(part):
   #
   # AMP 1445121-5
   # MTA .050 SMT 5-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1
      #
      self.shape = translate(pad_MTA,-.1,-.1,0)
      self.pad.append(point(-.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'red,green',14))
      #
      # pin 2
```

```
      #
      self.shape = add(self.shape,translate(pad_MTA,0,-.1,0))
      self.pad.append(point(0,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'black',14))
      #
      # pin 3
      #
      self.shape = add(self.shape,translate(pad_MTA,.1,-.1,0))
      self.pad.append(point(.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'orange',14))
      #
      # pin 4
      #
      self.shape = add(self.shape,translate(pad_MTA,.05,.1,0))
      self.pad.append(point(.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'yellow',14))
      #
      # pin 5
      #
      self.shape = add(self.shape,translate(pad_MTA,-.05,.1,0))
      self.pad.append(point(-.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'brown',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.262,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.262,0,0))

class MTA_LCD_ctrl(part):
   #
   # AMP 1445121-5
   # MTA .050 SMT 5-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1
      #
      self.shape = translate(pad_MTA,-.1,-.1,0)
      self.pad.append(point(-.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'E\n6',14))
      #
      # pin 2
      #
      self.shape = add(self.shape,translate(pad_MTA,0,-.1,0))
      self.pad.append(point(0,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V0\n3',14))
      #
      # pin 3
      #
      self.shape = add(self.shape,translate(pad_MTA,.1,-.1,0))
      self.pad.append(point(.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND,R/W\n1,5',14))
      #
      # pin 4
      #
      self.shape = add(self.shape,translate(pad_MTA,.05,.1,0))
      self.pad.append(point(.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'VCC\n2',14))
      #
      # pin 5
      #
      self.shape = add(self.shape,translate(pad_MTA,-.05,.1,0))
      self.pad.append(point(-.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'RS\n4',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.262,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.262,0,0))

class MTA_ICP(part):
   #
   # AMP 1445121-5
   # MTA .050 SMT 4-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: MISO
      #
      self.shape = translate(pad_MTA,-.1,-.1,0)
      self.pad.append(point(-.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'MISO',14))
      #
      # pin 2: GND
      #
      self.shape = add(self.shape,translate(pad_MTA,0,-.1,0))
      self.pad.append(point(0,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))
      #
      # pin 3: MOSI
      #
      self.shape = add(self.shape,translate(pad_MTA,.1,-.1,0))
      self.pad.append(point(.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'MOSI',14))
      #
```

```
      # pin 4: -RESET
      #
      self.shape = add(self.shape,translate(pad_MTA,.05,.1,0))
      self.pad.append(point(.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'-RESET',14))
      #
      # pin 5: SCK
      #
      self.shape = add(self.shape,translate(pad_MTA,-.05,.1,0))
      self.pad.append(point(-.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'SCK',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.262,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.262,0,0))

class power_65mm(part):
   #
   # CUI PJ1-023-SMT
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: power
      #
      self.shape = cube(.433,.512,-.047,.047,0,0)
      self.pad.append(point(.467,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'P',14))
      #
      # pin 2: ground
      #
      self.shape = add(self.shape,cube(.285,.423,-.189,-.098,0,0))
      self.pad.append(point(.354,-.144,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 3: contact
      #
      self.shape = add(self.shape,cube(.325,.463,.098,.189,0,0))
      self.pad.append(point(.394,.144,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,cube(.108,.246,-.169,-.110,0,0))
      self.shape = add(self.shape,cube(.069,.207,.110,.169,0,0))

pad_stereo_2_5mm = cube(-.03,.03,-.05,.05,0,0)

class stereo_2_5mm(part):
   #
   # CUI SJ1-2533-SMT
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: base
      #
      self.shape = translate(pad_stereo_2_5mm,-.130,-.16,0)
      self.pad.append(point(-.130,-.149,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'base',14))
      #
      # pin 2: tip
      #
      self.shape = add(self.shape,translate(pad_stereo_2_5mm,.197,.15,0))
      self.pad.append(point(.197,.141,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'tip',14))
      #
      # pin 3: middle
      #
      self.shape = add(self.shape,translate(pad_stereo_2_5mm,-.012,-.16,0))
      self.pad.append(point(-.012,-.149,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'middle',14))

pad_Molex = cube(-.0155,.0155,-.0265,.0265,0,0)
pad_Molex_solder = cube(-.055,.055,-.065,.065,0,0)

class Molex_serial(part):
   #
   # Molex 53261-0471
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: Rx
      #
      self.shape = translate(pad_Molex,-.075,.064,0)
      self.pad.append(point(-.075,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Rx',14))
      #
      # pin 2: Tx
      #
      self.shape = add(self.shape,translate(pad_Molex,-.025,.064,0))
      self.pad.append(point(-.025,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Tx',14))
```

```
      #
      # pin 3: DTR
      #
      self.shape = add(self.shape,translate(pad_Molex,.025,.064,0))
      self.pad.append(point(.025,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DTR',14))
      #
      # pin 4: GND
      #
      self.shape = add(self.shape,translate(pad_Molex,.075,.064,0))
      self.pad.append(point(.075,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_Molex_solder,-.16,-.065,0))
      self.shape = add(self.shape,translate(pad_Molex_solder,.16,-.065,0))

#
# switches
#

pad_button_6mm = cube(-.04,.04,-.03,.03,0,0)

class button_6mm(part):
   #
   # Omron 6mm pushbutton
   # B3SN-3112P
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # left 1
      #
      self.shape = translate(pad_button_6mm,-.125,.08,0)
      self.pad.append(point(-.125,.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'L1',14))
      #
      # right 1
      #
      self.shape = add(self.shape,translate(pad_button_6mm,-.125,-.08,0))
      self.pad.append(point(-.125,-.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'R1',14))
      #
      # right 2
      #
      self.shape = add(self.shape,translate(pad_button_6mm,.125,-.08,0))
      self.pad.append(point(.125,-.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'R2',14))
      #
      # left 2
      #
      self.shape = add(self.shape,translate(pad_button_6mm,.125,.08,0))
      self.pad.append(point(.125,.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'L2',14))

#
# crystals and resonators
#

pad_XTAL = cube(-.016,.016,-.077,.077,0,0)

class XTAL(part):
   #
   # Panasonic EFOBM series
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # left
      #
      self.shape = translate(pad_XTAL,-.053,0,0)
      self.pad.append(point(-.053,0,0))
      #
      # ground
      #
      self.shape = add(self.shape,translate(pad_XTAL,0,0,0))
      self.pad.append(point(0,0,0))
      #
      # right
      #
      self.shape = add(self.shape,translate(pad_XTAL,.053,0,0))
      self.pad.append(point(.053,0,0))

#
# diodes, transistors, regulators
#

class D_1206(part):
   #
   # 1206 diode
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
```

```
      # anode
      #
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A',14))
      #
      # cathode
      #
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))

class LED_1206(part):
   #
   # 1206 LED
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # anode
      #
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A',14))
      #
      # cathode
      #
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))

class phototransistor_1206(part):
   #
   # 1206 phototransistor
   # OPTEK 520,521
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # collector
      #
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))
      #
      # emitter
      #
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'E',14))

pad_SOD_123 = cube(-.02,.02,-.024,.024,0,0)

class D_SOD_123(part):
   #
   # SOD-123 diode
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # anode
      #
      self.shape = translate(pad_SOD_123,-.07,0,0)
      self.pad.append(point(-.07,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A',14))
      #
      # cathode
      #
      self.shape = add(self.shape,translate(pad_SOD_123,.07,0,0))
      self.pad.append(point(.07,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))

pad_SOT23 = cube(-.015,.015,-.02,.02,0,0)

class NMOSFET_SOT23(part):
   #
   # Fairchild NDS355AN
   #
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: gate
      #
      self.shape = translate(pad_SOT23,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 2: source
      #
      self.shape = add(self.shape,translate(pad_SOT23,.0375,-.045,0))
```

```
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'S',14))
      #
      # pin 3: drain
      #
      self.shape = add(self.shape,translate(pad_SOT23,0,.045,0))
      self.pad.append(point(0,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'D',14))

class PMOSFET_SOT23(part):
   #
   # Fairchild NDS356AP
   #
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: gate
      #
      self.shape = translate(pad_SOT23,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 2: source
      #
      self.shape = add(self.shape,translate(pad_SOT23,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'S',14))
      #
      # pin 3: drain
      #
      self.shape = add(self.shape,translate(pad_SOT23,0,.045,0))
      self.pad.append(point(0,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'D',14))

class regulator_SOT23(part):
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: output
      #
      self.shape = translate(pad_SOT23,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: input
      #
      self.shape = add(self.shape,translate(pad_SOT23,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'in',14))
      #
      # pin 3: ground
      #
      self.shape = add(self.shape,translate(pad_SOT23,0,.045,0))
      self.pad.append(point(0,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'gnd',14))

pad_SOT223 = cube(-.02,.02,-.03,.03,0,0)
pad_SOT223_ground = cube(-.065,.065,-.03,.03,0,0)

class regulator_SOT223(part):
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: input
      #
      self.shape = translate(pad_SOT223,-.09,-.12,0)
      self.pad.append(point(-.09,-.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1I',14))
      #
      # pin 2: ground
      #
      self.shape = add(self.shape,translate(pad_SOT223,0,-.12,0))
      self.pad.append(point(0,-.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 3: output
      #
      self.shape = add(self.shape,translate(pad_SOT223,.09,-.12,0))
      self.pad.append(point(.09,-.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'O',14))
      #
      # pin 4: ground
      #
      self.shape = add(self.shape,translate(pad_SOT223_ground,0,.12,0))
      self.pad.append(point(0,.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
```

```
pad_SM8 = cube(-.035,.035,-.016,.016,0,0)

class H_bridge_SM8(part):
   #
   # Zetex ZXMHC3A01T8
   #
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      d = .13
      #
      # pin 1: G3 (right N gate)
      #
      self.shape = translate(pad_SM8,-d,.09,0)
      self.pad.append(point(-d,.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GRN',14))
      #
      # pin 2: S2 S3 (N source)
      #
      self.shape = add(self.shape,translate(pad_SM8,-d,.03,0))
      self.pad.append(point(-d,.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'SN',14))
      #
      # pin 3: G2 (left N gate)
      #
      self.shape = add(self.shape,translate(pad_SM8,-d,-.03,0))
      self.pad.append(point(-d,-.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GLN',14))
      #
      # pin 4: G1 (left P gate)
      #
      self.shape = add(self.shape,translate(pad_SM8,-d,-.09,0))
      self.pad.append(point(-d,-.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GLP',14))
      #
      # pin 5: D1 D2 (left drain)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,-.09,0))
      self.pad.append(point(d,-.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DL',14))
      #
      # pin 6: S1 S4 (P source)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,-.03,0))
      self.pad.append(point(d,-.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'SP',14))
      #
      # pin 7: D3 D4 (right drain)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,.03,0))
      self.pad.append(point(d,.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DR',14))
      #
      # pin 8: G4 (right N gate)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,.09,0))
      self.pad.append(point(d,.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GRP',14))

#
# ICs
#

pad_SOT23_5 = cube(-.01,.01,-.02,.02,0,0)

class op_amp_SOT23_5(part):
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: output
      #
      self.shape = translate(pad_SOT23_5,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'O',14))
      #
      # pin 2: V-
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,0,-.045,0))
      self.pad.append(point(0,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V-',14))
      #
      # pin 3: I+
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'I+',14))
      #
      # pin 4: I-
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,.0375,.045,0))
      self.pad.append(point(.0375,.045,0))
```

```
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'I-',14))
      #
      # pin 5: V+
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,-.0375,.045,0))
      self.pad.append(point(-.0375,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V+',14))

pad_SOICN = cube(-.035,.035,-.015,.015,0,0)

class op_amp_SOICN(part):
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: A out
      #
      self.shape = translate(pad_SOICN,-.12,.075,0)
      self.pad.append(point(-.12,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1 Ao',14))
      #
      # pin 2: A-
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.025,0))
      self.pad.append(point(-.12,.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A-',14))
      #
      # pin 3: A+
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.025,0))
      self.pad.append(point(-.12,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A+',14))
      #
      # pin 4: V-
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.075,0))
      self.pad.append(point(-.12,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V-',14))
      #
      # pin 5: B+
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.075,0))
      self.pad.append(point(.12,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'B+',14))
      #
      # pin 6: B-
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.025,0))
      self.pad.append(point(.12,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'B-',14))
      #
      # pin 7: B out
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.025,0))
      self.pad.append(point(.12,.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Bo',14))
      #
      # pin 8: V+
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.075,0))
      self.pad.append(point(.12,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V+',14))

pad_SOIC = cube(-.043,.043,-.015,.015,0,0)

class ATtiny45_SOIC(part):
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: PB5/dW/ADC0/-RESET/PCINT5
      #
      self.shape = translate(pad_SOIC,-.14,.075,0)
      self.pad.append(point(-.14,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: PB3/ADC3/-OC1B/CLKI/XTAL1/PCINT3
      #
      self.shape = add(self.shape,translate(pad_SOIC,-.14,.025,0))
      self.pad.append(point(-.14,.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB3',14))
      #
      # pin 3: PB4/ADC2/OC1B/CLKO/XTAL2/PCINT4
      #
      self.shape = add(self.shape,translate(pad_SOIC,-.14,-.025,0))
      self.pad.append(point(-.14,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB4',14))
      #
      # pin 4: GND
      #
      self.shape = add(self.shape,translate(pad_SOIC,-.14,-.075,0))
      self.pad.append(point(-.14,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))
      #
      # pin 5: PB0/MOSI/DI/SDA/AIN0/OC0A/-OC1A/AREF/PCINT0
      #
      self.shape = add(self.shape,translate(pad_SOIC,.14,-.075,0))
      self.pad.append(point(.14,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
```

```
        #
        # pin 6: PB1/MISO/DO/AIN1/OC0B/OC1A/PCINT1
        #
        self.shape = add(self.shape,translate(pad_SOIC,.14,-.025,0))
        self.pad.append(point(.14,-.025,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB1',14))
        #
        # pin 7: PB2/SCK/USCK/SCL/ADC1/T0/INT0/PCINT2
        #
        self.shape = add(self.shape,translate(pad_SOIC,.14,.025,0))
        self.pad.append(point(.14,.025,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB2',14))
        #
        # pin 8: VCC
        #
        self.shape = add(self.shape,translate(pad_SOIC,.14,.075,0))
        self.pad.append(point(.14,.075,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'VCC',14))

class ATtiny44_SOICN(part):
   def __init__(self,value=''):
        self.value = value
        self.pad = [point(0,0,0)]
        self.labels = []
        #
        # pin 1: VCC
        #
        self.shape = translate(pad_SOICN,-.12,.15,0)
        self.pad.append(point(-.12,.15,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
        #
        # pin 2: PB0/XTAL1/PCINT8
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 3: PB1/XTAL2/PCINT9
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.050,0))
        self.pad.append(point(-.12,.05,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB1',14))
        #
        # pin 4: PB3/dW/-RESET/PCINT11
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,0,0))
        self.pad.append(point(-.12,0,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB3',14))
        #
        # pin 5: PB2/CKOUT/OC0A/INT0/PCINT10
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,-.05,0))
        self.pad.append(point(-.12,-.05,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB2',14))
        #
        # pin 6: PA7/ADC7/OC0B/ICP/PCINT7
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,-.1,0))
        self.pad.append(point(-.12,-.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA7',14))
        #
        # pin 7: PA6/ADC6/MOSI/SDA/OC1A/PCINT6
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,-.15,0))
        self.pad.append(point(-.12,-.15,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA6',14))
        #
        # pin 8: PA5/ADC5/DO/MISO/OC1B/PCINT5
        #
        self.shape = add(self.shape,translate(pad_SOICN,.12,-.15,0))
        self.pad.append(point(.12,-.15,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA5',14))
        #
        # pin 9: PA4/ADC4/USCK/SCL/T1/PCINT4
        #
        self.shape = add(self.shape,translate(pad_SOICN,.12,-.1,0))
        self.pad.append(point(.12,-.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA4',14))
        #
        # pin 10: PA3/ADC3/T0/PCINT3
        #
        self.shape = add(self.shape,translate(pad_SOICN,.12,-.05,0))
        self.pad.append(point(.12,-.05,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA3',14))
        #
        # pin 11: PA2/ADC2/AIN1/PCINT2
        #
        self.shape = add(self.shape,translate(pad_SOICN,.12,0,0))
        self.pad.append(point(.12,0,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA2',14))
        #
        # pin 12: PA1/ADC1/AIN0/PCINT1
        #
        self.shape = add(self.shape,translate(pad_SOICN,.12,.050,0))
        self.pad.append(point(.12,.05,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA1',14))
        #
        # pin 13: PA0/ADC0/AREF/PCINT0
        #
        self.shape = add(self.shape,translate(pad_SOICN,.12,.1,0))
        self.pad.append(point(.12,.1,0))
```

```
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA0',14))
      #
      # pin 14: GND
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.15,0))
      self.pad.append(point(.12,.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))

pad_TQFP = cube(-.043,.043,-.015,.015,0,0)

class ATmega88_TQFP(part):
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []

      #
      # pin 1: PD3/PCINT19/OC2B/INT1
      #
      self.shape = translate(pad_SOICN,-.12,.15,0)
      self.pad.append(point(-.12,.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: PD4/PCINT20/XCK/T0
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 3: GND
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 4: VCC
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 5: GND
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 6: VCC
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 7: PB6/PCINT6/XTAL1/TOSC1
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 8: PB7/PCINT7/XTAL2/TOSC2
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

      #
      # pin 9: PD5/PCINT21/OC0B/T1
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 10: PD6/PCINT22/OC0A/AIN0
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 11: PD7/PCINT23/AIN1
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 12: PB0/PCINT0/CLKO/ICP1
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 13: PB1/PCINT1/OC1A
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 14: PB2/PCINT2/-SS/OC1B
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
```

```
#
# pin 15: PB3/PCINT3/OC2A/MOSI
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 16: PB4/PCINT4/MISO
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

#
# pin 17: PB5/SCK/PCINT5
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 18: AVCC
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 19: ADC6
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 20: AREF
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 21: GND
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 22: ADC7
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 23: PC0/ADC0/PCINT8
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 24: PC1/ADC1/PCINT9
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

#
# pin 25: PC2/ADC2/PCINT10
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 26: PC2/ADC3/PCINT11
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 27: PC4/ADC4/SDA/PCINT12
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 28: PC5/ADC5/SCL/PCINT13
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 29: PC6/-RESET/PCINT14
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 30: PD0/RXD/PCINT16
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 31: PD1/TXD/PCINT17
#
```

```
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 32: PD2/INT0/PCINT18
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

#
# graphics
#

class CBA(part):
   def __init__(self,r=.02):
      self.value = ''
      self.pad = [point(0,0,0)]
      self.labels = []
      d = 3*r
      self.shape = circle(0,0,r)
      self.shape = add(self.shape,translate(circle(0,0,r),-d,d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),-d,0,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),-d,-d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),0,-d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),d,-d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),d,0,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),d,d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),0,d,0))

class hello(part):
   #
   # HELLO
   #
   def __init__(self,w,z):
      self.value = ''
      self.pad = [point(0,0,0)]
      self.labels = []
      # H
      self.shape = rectangle(-.01,.01,0,.1)
      self.shape = add(self.shape,rectangle(0,.05,.04,.06))
      self.shape = add(self.shape,rectangle(.04,.06,0,.1))
      # E
      self.shape = add(self.shape,rectangle(.08,.1,0,.1))
      self.shape = add(self.shape,rectangle(.1,.14,0,.02))
      self.shape = add(self.shape,rectangle(.1,.13,.04,.06))
      self.shape = add(self.shape,rectangle(.1,.14,.08,.1))
      # L
      self.shape = add(self.shape,rectangle(.16,.18,0,.1))
      self.shape = add(self.shape,rectangle(.18,.21,0,.02))
      # L
      self.shape = add(self.shape,rectangle(.23,.25,0,.1))
      self.shape = add(self.shape,rectangle(.25,.28,0,.02))
      # 0
      self.shape = add(self.shape,rectangle(.3,.32,0,.1))
      self.shape = add(self.shape,rectangle(.32,.345,0,.02))
      self.shape = add(self.shape,rectangle(.32,.345,.08,.1))
      self.shape = add(self.shape,rectangle(.345,.365,0,.1))

#
# define board
#

w = .016
width = 1.1
height = 1.03
x = 1
y = 1
z = -.005
d = .06
mask = .004

pcb = PCB(x,y,width,height,mask)

IC1 = ATtiny44_SOICN('IC1\nt44')
pcb = IC1.add(pcb,x+.35,y+.52,z)

J1 = MTA_ICP('J1\nICP')
pcb = J1.add(pcb,IC1.x,IC1.pad[7].y-.21,z,angle=180)

pcb = wire(pcb,w,
   IC1.pad[7],
   J1.pad[3])

pcb = wire(pcb,w,
   IC1.pad[4],
   J1.pad[4])

pcb = wire(pcb,w,
   IC1.pad[14],
   J1.pad[2])

pcb = wire(pcb,w,
   IC1.pad[9],
   J1.pad[5])

pcb = wire(pcb,w,
   IC1.pad[8],
   J1.pad[1])

J2 = MTA_power('J2\npower')
```

```
pcb = J2.add(pcb,IC1.x-.03,IC1.pad[1].y+.2,z,angle=0)

pcb = wire(pcb,w,
   IC1.pad[14],
   J2.pad[1])

IC2 = regulator_SOT23('IC2\n5V')
pcb = IC2.add(pcb,IC1.x-.28,IC1.pad[1].y+.03,z,angle=180)

pcb = wire(pcb,w,
   IC1.pad[1],
   IC2.pad[1])

pcb = wire(pcb,w,
   J2.pad[2],
   IC2.pad[2])

pcb = wire(pcb,.014,
   point(J2.pad[1].x,J2.pad[1].y+.03,z),
   IC2.pad[3])

C1 = C_1206('C1\n3.3uF')
pcb = C1.add(pcb,IC2.x,IC2.y-.18,z,angle=90)

pcb = wire(pcb,w,
   C1.pad[1],
   IC2.pad[3])

pcb = wire(pcb,w,
   C1.pad[2],
   point(C1.x+.07,IC1.pad[1].y,z),
   IC1.pad[1])

R1 = R_1206('R1\n10k')
pcb = R1.add(pcb,C1.x,C1.y-.2,z,angle=90)

pcb = wire(pcb,w,
   R1.pad[1],
   C1.pad[2])

pcb = wire(pcb,w,
   R1.pad[2],
   point(J1.pad[3].x-.05,J1.pad[4].y,z),
   J1.pad[4])

J3 = MTA_stepper('J3\nstepper')
pcb = J3.add(pcb,IC1.x+.58,IC1.y+.06,z,angle=90)

pcb = wire(pcb,w,
   J2.pad[2],
   J3.pad[1])

T1 = NMOSFET_SOT23('T1\nN')
pcb = T1.add(pcb,IC1.x+.32,IC1.y+.19,z,angle=90)

pcb = wire(pcb,w,
  IC1.pad[13],
  point(IC1.pad[13].x+.07,T1.pad[1].y,z),
  T1.pad[1])

pcb = wire(pcb,w,
   T1.pad[2],
   point(T1.x,J2.pad[1].y+.03,z),
   J2.pad[1])

pcb = wire(pcb,w,
   J3.pad[5],
   T1.pad[3])

T2 = NMOSFET_SOT23('T2\nN')
pcb = T2.add(pcb,T1.x,T1.y-.13,z,angle=90)

pcb = wire(pcb,w,
  IC1.pad[12],
  point(IC1.pad[12].x+.1,T2.pad[1].y,z),
  T2.pad[1])

pcb = wire(pcb,w,
   T1.pad[2],
   point(T2.x,T2.pad[2].y,z),
   T2.pad[2])

pcb = wire(pcb,w,
   J3.pad[2],
   T2.pad[3])

T3 = NMOSFET_SOT23('T3\nN')
pcb = T3.add(pcb,T2.x,T2.y-.13,z,angle=90)

pcb = wire(pcb,w,
  IC1.pad[11],
  point(IC1.pad[11].x+.1,T3.pad[1].y,z),
  T3.pad[1])

pcb = wire(pcb,w,
   T2.pad[2],
   point(T3.x,T3.pad[2].y,z),
   T3.pad[2])

pcb = wire(pcb,w,
   J3.pad[4],
```

```
    T3.pad[3])

T4 = NMOSFET_SOT23('T4\nN')
pcb = T4.add(pcb,T3.x,T3.y-.13,z,angle=90)

pcb = wire(pcb,w,
  IC1.pad[10],
  point(IC1.pad[10].x+.07,T4.pad[1].y,z),
  T4.pad[1])

pcb = wire(pcb,w,
    T3.pad[2],
    point(T4.x,T4.pad[2].y,z),
    T4.pad[2])

pcb = wire(pcb,w,
    T4.pad[3],
    J3.pad[3])

H1 = hello(w,z)
pcb = H1.add(pcb,J1.x+.36,J1.y-.06,z)

cad.function = add(pcb.board,pcb.exterior)

#
# uncomment to export traces
#

#cad.function = pcb.board

#
# uncomment to export exterior
#

#cad.function = pcb.exterior

#
# uncomment to export solder mask
#

#cad.function = pcb.mask

#
# uncomment to mill out board
#
# use 1/32 end-mill
# set tool diameter = .0312
# set xy, z speed = .5
# set # contours = 1
#

#cad.function = pcb.interior
#z = -.065

#
# define limits and parameters
#

cad.xmin = x-.1 # min x to render
cad.xmax = x+width+.1 # max x to render
cad.ymin = y-.1 # min y to render
cad.ymax = y+height+.1 # max y to render
cad.zmin = z
cad.zmax = .050
dpi = 200 # low resolution for previewing
#dpi = 500 # high resolution for machining
nxy = int(dpi*(cad.xmax-cad.xmin))
cad.nx = nxy # x points to render
cad.ny = nxy # y points to render
cad.nz = 1
cad.view('xy') # 2D view
```

```
#
# hello.array.44.cad
#
# Charlieplex LED array hello-world
#
# Neil Gershenfeld
# CBA MIT 10/18/07
#
# (c) Massachusetts Institute of Technology 2007
# Permission granted for experimental and personal use;
# license for commercial sale available from MIT.
#

#
# define shapes and transformation
#
# circle(x0, y0, r)
# cylinder(x0, y0, z0, z1, r)
# cone(x0, y0, z0, z1, r0)
# sphere(x0, y0, z0, r)
# torus(x0, y0, z0, r0, r1)
# rectangle(x0, x1, y0, y1)
# cube(x0, x1, y0, y1, z0, z1)
# triangle(x0, y0, x1, y1, x2, y2) (points in clockwise order)
# pyramid(x0, x1, y0, y1, z0, z1)
# function(Z_of_XY)
# functions(upper_Z_of_XY,lower_Z_of_XY)
# add(part1, part2)
# subtract(part1, part2)
# intersect(part1, part2)
# move(part,dx,dy)
# translate(part,dx,dy,dz)
# rotate(part, angle)
# rotate_x(part, angle)
# rotate_y(part, angle)
# rotate_z(part, angle)
# rotate_90(part)
# rotate_180(part)
# rotate_270(part)
# reflect_x(part)
# reflect_y(part)
# reflect_z(part)
# reflect_xy(part)
# reflect_xz(part)
# reflect_yz(part)
# scale_x(part, x0, sx)
# scale_y(part, y0, sy)
# scale_z(part, z0, sz)
# scale_xy(part, x0, y0, sxy)
# scale_xyz(part, x0, y0, z0, sxyz)
# coscale_x_y(part, x0, y0, y1, angle0, angle1, amplitude, offset)
# coscale_x_z(part, x0, z0, z1, angle0, angle1, amplitude, offset)
# coscale_xy_z(part, x0, y0, z0, z1, angle0, angle1, amplitude, offset)
# taper_x_y(part, x0, y0, y1, s0, s1)
# taper_x_z(part, x0, z0, z1, s0, s1)
# taper_xy_z(part, x0, y0, z0, z1, s0, s1)
# shear_x_y(part, y0, y1, dx0, dx1)
# shear_x_z(part, z0, z1, dx0, dx1)
# (more to come)

def circle(x0, y0, r):
   part = "(((X-x0)**2 + (Y-y0)**2) <= r**2)"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'r',str(r))
   return part

def cylinder(x0, y0, z0, z1, r):
   part = "(((X-x0)**2 + (Y-y0)**2 <= r**2) & (Z >= z0) & (Z <= z1))"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'r',str(r))
   return part

def cone(x0, y0, z0, z1, r0):
   part = cylinder(x0, y0, z0, z1, r0)
   part = taper_xy_z(part, x0, y0, z0, z1, 1.0, 0.0)
   return part

def sphere(x0, y0, z0, r):
   part = "(((X-x0)**2 + (Y-y0)**2 + (Z-z0)**2) <= r**2)"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'r',str(r))
   return part

def torus(x0, y0, z0, r0, r1):
   part = "(((r0 - sqrt((X-x0)**2 + (Y-y0)**2))**2 + (Z-z0)**2) <= r1**2)"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'r0',str(r0))
   part = replace(part,'r1',str(r1))
   return part

def rectangle(x0, x1, y0, y1):
   part = "((X >= x0) & (X <= x1) & (Y >= y0) & (Y <= y1))"
   part = replace(part,'x0',str(x0))
```

```
   part = replace(part,'x1',str(x1))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   return part

def cube(x0, x1, y0, y1, z0, z1):
   part = "((X >= x0) & (X <= x1) & (Y >= y0) & (Y <= y1) & (Z >= z0) & (Z <= z1))"
   part = replace(part,'x0',str(x0))
   part = replace(part,'x1',str(x1))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   return part

def triangle(x0, y0, x1, y1, x2, y2): # points in clockwise order
   part = "((((y1-y0)*(X-x0)-(x1-x0)*(Y-y0)) >= 0) & (((y2-y1)*(X-x1)-(x2-x1)*(Y-y1)) >= 0) & (((y0-y2)*(X-x2)-(x0-x2)*(Y-y2)) >= 0))"
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'x1',str(x1))
   part = replace(part,'y1',str(y1))
   part = replace(part,'x2',str(x2))
   part = replace(part,'y2',str(y2))
   return part

def pyramid(x0, x1, y0, y1, z0, z1):
   part = cube(x0, x1, y0, y1, z0, z1)
   part = taper_xy_z(part, (x0+x1)/2., (y0+y1)/2., z0, z1, 1.0, 0.0)
   return part

def function(Z_of_XY):
   part = '(Z <= '+Z_of_XY+')'
   return part

def functions(upper_Z_of_XY,lower_Z_of_XY):
   part = '(Z <= '+upper_Z_of_XY+') & (Z >= '+lower_Z_of_XY+')'
   return part

def add(part1, part2):
   part = "part1 | part2"
   part = replace(part,'part1',part1)
   part = replace(part,'part2',part2)
   return part

def subtract(part1, part2):
   part = "(part1) & ~(part2)"
   part = replace(part,'part1',part1)
   part = replace(part,'part2',part2)
   return part

def intersect(part1, part2):
   part = "(part1) & (part2)"
   part = replace(part,'part1',part1)
   part = replace(part,'part2',part2)
   return part

def move(part,dx,dy):
   part = replace(part,'X','(X-'+str(dx)+')')
   part = replace(part,'Y','(Y-'+str(dy)+')')
   return part

def translate(part,dx,dy,dz):
   part = replace(part,'X','(X-'+str(dx)+')')
   part = replace(part,'Y','(Y-'+str(dy)+')')
   part = replace(part,'Z','(Z-'+str(dz)+')')
   return part

def rotate(part, angle):
   angle = angle*pi/180
   part = replace(part,'X','(cos(angle)*X+sin(angle)*y)')
   part = replace(part,'Y','(-sin(angle)*X+cos(angle)*y)')
   part = replace(part,'y','Y')
   part = replace(part,'angle',str(angle))
   return part

def rotate_x(part, angle):
   angle = angle*pi/180
   part = replace(part,'Y','(cos(angle)*Y+sin(angle)*z)')
   part = replace(part,'Z','(-sin(angle)*Y+cos(angle)*z)')
   part = replace(part,'z','Z')
   part = replace(part,'angle',str(angle))
   return part

def rotate_y(part, angle):
   angle = angle*pi/180
   part = replace(part,'X','(cos(angle)*X+sin(angle)*z)')
   part = replace(part,'Z','(-sin(angle)*X+cos(angle)*z)')
   part = replace(part,'z','Z')
   part = replace(part,'angle',str(angle))
   return part

def rotate_z(part, angle):
   angle = angle*pi/180
   part = replace(part,'X','(cos(angle)*X+sin(angle)*y)')
   part = replace(part,'Y','(-sin(angle)*X+cos(angle)*y)')
   part = replace(part,'y','Y')
   part = replace(part,'angle',str(angle))
   return part

def rotate_90(part):
   part = reflect_xy(part)
```

```
   part = reflect_y(part)
   return part

def rotate_180(part):
   part = reflect_xy(part)
   part = reflect_y(part)
   part = reflect_xy(part)
   part = reflect_y(part)
   return part

def rotate_270(part):
   part = reflect_xy(part)
   part = reflect_y(part)
   part = reflect_xy(part)
   part = reflect_y(part)
   part = reflect_xy(part)
   part = reflect_y(part)
   return part

def reflect_x(part):
   part = replace(part,'X','-X')
   return part

def reflect_y(part):
   part = replace(part,'Y','-Y')
   return part

def reflect_z(part):
   part = replace(part,'Z','-Z')
   return part

def reflect_xy(part):
   part = replace(part,'X','temp')
   part = replace(part,'Y','X')
   part = replace(part,'temp','Y')
   return part

def reflect_xz(part):
   part = replace(part,'X','temp')
   part = replace(part,'Z','X')
   part = replace(part,'temp','Z')
   return part

def reflect_yz(part):
   part = replace(part,'Y','temp')
   part = replace(part,'Z','Y')
   part = replace(part,'temp','Z')
   return part

def scale_x(part, x0, sx):
   part = replace(part,'X','(x0 + (X-x0)/sx)')
   part = replace(part,'x0',str(x0))
   part = replace(part,'sx',str(sx))
   return part

def scale_y(part, y0, sy):
   part = replace(part,'Y','(y0 + (Y-y0)/sy)')
   part = replace(part,'y0',str(y0))
   part = replace(part,'sy',str(sy))
   return part

def scale_z(part, z0, sz):
   part = replace(part,'Z','(z0 + (Z-z0)/sz)')
   part = replace(part,'z0',str(z0))
   part = replace(part,'sz',str(sz))
   return part

def scale_xy(part, x0, y0, sxy):
   part = replace(part,'X','(x0 + (X-x0)/sxy)')
   part = replace(part,'Y','(y0 + (Y-y0)/sxy)')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'sxy',str(sxy))
   return part

def scale_xyz(part, x0, y0, z0, sxyz):
   part = replace(part,'X','(x0 + (X-x0)/sxyz)')
   part = replace(part,'Y','(y0 + (Y-y0)/sxyz)')
   part = replace(part,'Z','(z0 + (Z-z0)/sxyz)')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'sxyz',str(sxyz))
   return part

def coscale_x_y(part, x0, y0, y1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
   part = replace(part,'X','(x0 + (X-x0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Y-y0)/(y1-y0))))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

def coscale_x_z(part, x0, z0, z1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
```

```
   part = replace(part,'X','(x0 + (X-x0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0))))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

def coscale_xy_z(part, x0, y0, z0, z1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
   part = replace(part,'X','(x0 + (X-x0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0))))')
   part = replace(part,'Y','(y0 + (Y-y0)/(offset + amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0))))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

def taper_x_y(part, x0, y0, y1, s0, s1):
   part = replace(part,'X','(x0 + (X-x0)*(y1-y0)/(s1*(Y-y0) + s0*(y1-Y)))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'s0',str(s0))
   part = replace(part,'s1',str(s1))
   return part

def taper_x_z(part, x0, z0, z1, s0, s1):
   part = replace(part,'X','(x0 + (X-x0)*(z1-z0)/(s1*(Z-z0) + s0*(z1-Z)))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'s0',str(s0))
   part = replace(part,'s1',str(s1))
   return part

def taper_xy_z(part, x0, y0, z0, z1, s0, s1):
   part = replace(part,'X','(x0 + (X-x0)*(z1-z0)/(s1*(Z-z0) + s0*(z1-Z)))')
   part = replace(part,'Y','(y0 + (Y-y0)*(z1-z0)/(s1*(Z-z0) + s0*(z1-Z)))')
   part = replace(part,'x0',str(x0))
   part = replace(part,'y0',str(y0))
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'s0',str(s0))
   part = replace(part,'s1',str(s1))
   return part

def shear_x_y(part, y0, y1, dx0, dx1):
   part = replace(part,'X','(X - dx0 - (dx1-dx0)*(Y-y0)/(y1-y0))')
   part = replace(part,'y0',str(y0))
   part = replace(part,'y1',str(y1))
   part = replace(part,'dx0',str(dx0))
   part = replace(part,'dx1',str(dx1))
   return part

def shear_x_z(part, z0, z1, dx0, dx1):
   part = replace(part,'X','(X - dx0 - (dx1-dx0)*(Z-z0)/(z1-z0))')
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'dx0',str(dx0))
   part = replace(part,'dx1',str(dx1))
   return part

def coshear_x_z(part, z0, z1, angle0, angle1, amplitude, offset):
   phase0 = pi*angle0/180.
   phase1 = pi*angle1/180.
   part = replace(part,'X','(X - offset - amplitude*cos(phase0 + (phase1-phase0)*(Z-z0)/(z1-z0)))')
   part = replace(part,'z0',str(z0))
   part = replace(part,'z1',str(z1))
   part = replace(part,'phase0',str(phase0))
   part = replace(part,'phase1',str(phase1))
   part = replace(part,'amplitude',str(amplitude))
   part = replace(part,'offset',str(offset))
   return part

#
# PCB classes and definitions
#

cad.labels = []

class PCB:
   def __init__(self,x0,y0,width,height,mask):
      self.board = "False"
      self.interior = rectangle(x0,x0+width,y0,y0+height)
      self.exterior = subtract("True",rectangle(x0,x0+width,y0,y0+height))
      self.mask = "False"
   def add(self,part):
      self.board = add(self.board,part)
      self.mask = add(self.mask,move(part,-mask,mask))
      self.mask = add(self.mask,move(part,-mask,-mask))
      self.mask = add(self.mask,move(part,mask,mask))
      self.mask = add(self.mask,move(part,mask,-mask))
      return self
```

```
class part:
   def add(self,pcb,x,y,z=0,angle=0):
      self.x = x
      self.y = y
      self.z = z
      self.angle = angle
      if (angle == 90):
         self.shape = rotate_90(self.shape)
      elif (angle == 180):
         self.shape = rotate_180(self.shape)
      elif ((angle == 270) | (angle == -90)):
         self.shape = rotate_270(self.shape)
      angle = pi*angle/180
      self.shape = translate(self.shape,x,y,z)
      for i in range(len(self.pad)):
         xnew = cos(angle)*self.pad[i].x + sin(angle)*self.pad[i].y
         ynew = -sin(angle)*self.pad[i].x + cos(angle)*self.pad[i].y
           self.pad[i].x = x + xnew
           self.pad[i].y = y + ynew
         self.pad[i].z += z
      cad.labels.append(cad_text(x,y,z,self.value,size=14))
      for i in range(len(self.labels)):
         xnew = cos(angle)*self.labels[i].x + sin(angle)*self.labels[i].y
         ynew = -sin(angle)*self.labels[i].x + cos(angle)*self.labels[i].y
           self.labels[i].x = x + xnew
           self.labels[i].y = y + ynew
         self.labels[i].z += z
           cad.labels.append(self.labels[i])
      pcb = pcb.add(self.shape)
      return pcb

def wire(pcb,width,*points):
   for i in range(1,len(points)):
      x0 = points[i-1].x
      y0 = points[i-1].y
      z0 = points[i-1].z
      x1 = points[i].x
      y1 = points[i].y
      z1 = points[i].z
      if (x0 < x1):
         pcb.board = add(pcb.board,cube(x0-width/2,x1+width/2,y0-width/2,y0+width/2,z0,z0))
      elif (x1 < x0):
         pcb.board = add(pcb.board,cube(x1-width/2,x0+width/2,y0-width/2,y0+width/2,z0,z0))
      if (y0 < y1):
         pcb.board = add(pcb.board,cube(x1-width/2,x1+width/2,y0-width/2,y1+width/2,z0,z0))
      elif (y1 < y0):
         pcb.board = add(pcb.board,cube(x1-width/2,x1+width/2,y1-width/2,y0+width/2,z0,z0))
   return pcb

#
# PCB library
#

#
# discretes
#

pad_0402 = cube(-.0175,.0175,-.014,.014,0,0)

class R_0402(part):
   #
   # 0402 resistor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_0402,-.0265,0,0)
      self.pad.append(point(-.0265,0,0))
      self.shape = add(self.shape,translate(pad_0402,.0265,0,0))
      self.pad.append(point(.0265,0,0))

pad_1206 = cube(-.032,.032,-.034,.034,0,0)

class R_1206(part):
   #
   # 1206 resistor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.06,0,0))
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.06,0,0))

class C_1206(part):
   #
   # 1206 capacitor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.06,0,0))
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.06,0,0))
```

```
pad_1210 = cube(-.032,.032,-.048,.048,0,0)

class L_1210(part):
   #
   # 1210 inductor
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_1210,-.06,0,0)
      self.pad.append(point(-.06,0,0))
      self.shape = add(self.shape,translate(pad_1210,.06,0,0))
      self.pad.append(point(.06,0,0))

pad_choke = cube(-.06,.06,-.06,.06,0,0)

class choke(part):
   #
   # Panasonic ELLCTV
   #
   def __init__(self,value=''):
      self.value = value
      self.labels = []
      self.pad = [point(0,0,0)]
      self.shape = translate(pad_choke,-.177,-.177,0)
      self.pad.append(point(-.177,-.177,0))
      self.shape = add(self.shape,translate(pad_choke,.177,.177,0))
      self.pad.append(point(.177,.177,0))

#
# connectors
#

pad_MTA = cube(-.021,.021,-.041,.041,0,0)
pad_MTA_solder = cube(-.071,.071,-.041,.041,0,0)

class MTA_2(part):
   #
   # AMP 1445121-2
   # MTA .050 SMT 2-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1
      #
      self.shape = translate(pad_MTA,-.025,-.1,0)
      self.pad.append(point(-.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,.1,0))
      self.pad.append(point(.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'2',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.187,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.187,0,0))

class MTA_power(part):
   #
   # AMP 1445121-2
   # MTA .050 SMT 2-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: Gnd
      #
      self.shape = translate(pad_MTA,-.025,-.1,0)
      self.pad.append(point(-.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: Vcc
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,.1,0))
      self.pad.append(point(.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Vcc',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.187,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.187,0,0))

class MTA_i0(part):
   #
   # AMP 1445121-3
   # MTA .050 SMT 3-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
```

```
      # pin 1: GND
      #
      self.shape = translate(pad_MTA,.05,.1,0)
      self.pad.append(point(.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1\nGND',14))
      #
      # pin 2: power
      #
      self.shape = add(self.shape,translate(pad_MTA,-.05,.1,0))
      self.pad.append(point(-.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V',14))
      #
      # pin 3: data
      #
      self.shape = add(self.shape,translate(pad_MTA,0,-.1,0))
      self.pad.append(point(0,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'data',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.212,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.212,0,0))

class MTA_4(part):
   #
   # AMP 1445121-4
   # MTA .050 SMT 4-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1
      #
      self.shape = translate(pad_MTA,-.075,-.1,0)
      self.pad.append(point(-.075,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,-.1,0))
      self.pad.append(point(.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'2',14))
      #
      # pin 3
      #
      self.shape = add(self.shape,translate(pad_MTA,.075,.1,0))
      self.pad.append(point(.075,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'3',14))
      #
      # pin 4
      #
      self.shape = add(self.shape,translate(pad_MTA,-.025,.1,0))
      self.pad.append(point(-.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'4',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.237,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.237,0,0))

class MTA_LCD_data(part):
   #
   # AMP 1445121-4
   # MTA .050 SMT 4-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1
      #
      self.shape = translate(pad_MTA,-.075,-.1,0)
      self.pad.append(point(-.075,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DB7\n14',14))
      #
      # pin 2
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,-.1,0))
      self.pad.append(point(.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DB5\n12',14))
      #
      # pin 3
      #
      self.shape = add(self.shape,translate(pad_MTA,.075,.1,0))
      self.pad.append(point(.075,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DB4\n11',14))
      #
      # pin 4
      #
      self.shape = add(self.shape,translate(pad_MTA,-.025,.1,0))
      self.pad.append(point(-.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DB6\n13',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.237,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.237,0,0))
```

```
class MTA_serial(part):
   #
   # AMP 1445121-4
   # MTA .050 SMT 4-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: Gnd
      #
      self.shape = translate(pad_MTA,-.075,-.1,0)
      self.pad.append(point(-.075,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: Tx
      #
      self.shape = add(self.shape,translate(pad_MTA,.025,-.1,0))
      self.pad.append(point(.025,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Tx',14))
      #
      # pin 3: Rx
      #
      self.shape = add(self.shape,translate(pad_MTA,.075,.1,0))
      self.pad.append(point(.075,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Rx',14))
      #
      # pin 4: DTR
      #
      self.shape = add(self.shape,translate(pad_MTA,-.025,.1,0))
      self.pad.append(point(-.025,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DTR',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.237,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.237,0,0))

class MTA_5(part):
   #
   # AMP 1445121-5
   # MTA .050 SMT 5-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1
      #
      self.shape = translate(pad_MTA,-.1,-.1,0)
      self.pad.append(point(-.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2
      #
      self.shape = add(self.shape,translate(pad_MTA,0,-.1,0))
      self.pad.append(point(0,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'2',14))
      #
      # pin 3
      #
      self.shape = add(self.shape,translate(pad_MTA,.1,-.1,0))
      self.pad.append(point(.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'3',14))
      #
      # pin 4
      #
      self.shape = add(self.shape,translate(pad_MTA,.05,.1,0))
      self.pad.append(point(.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'4',14))
      #
      # pin 5
      #
      self.shape = add(self.shape,translate(pad_MTA,-.05,.1,0))
      self.pad.append(point(-.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'5',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.262,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.262,0,0))

class MTA_stepper(part):
   #
   # AMP 1445121-5
   # MTA .050 SMT 5-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1
      #
      self.shape = translate(pad_MTA,-.1,-.1,0)
      self.pad.append(point(-.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'red,green',14))
      #
      # pin 2
```

```
      #
      self.shape = add(self.shape,translate(pad_MTA,0,-.1,0))
      self.pad.append(point(0,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'black',14))
      #
      # pin 3
      #
      self.shape = add(self.shape,translate(pad_MTA,.1,-.1,0))
      self.pad.append(point(.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'orange',14))
      #
      # pin 4
      #
      self.shape = add(self.shape,translate(pad_MTA,.05,.1,0))
      self.pad.append(point(.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'yellow',14))
      #
      # pin 5
      #
      self.shape = add(self.shape,translate(pad_MTA,-.05,.1,0))
      self.pad.append(point(-.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'brown',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.262,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.262,0,0))

class MTA_LCD_ctrl(part):
   #
   # AMP 1445121-5
   # MTA .050 SMT 5-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1
      #
      self.shape = translate(pad_MTA,-.1,-.1,0)
      self.pad.append(point(-.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'E\n6',14))
      #
      # pin 2
      #
      self.shape = add(self.shape,translate(pad_MTA,0,-.1,0))
      self.pad.append(point(0,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V0\n3',14))
      #
      # pin 3
      #
      self.shape = add(self.shape,translate(pad_MTA,.1,-.1,0))
      self.pad.append(point(.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND,R/W\n1,5',14))
      #
      # pin 4
      #
      self.shape = add(self.shape,translate(pad_MTA,.05,.1,0))
      self.pad.append(point(.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'VCC\n2',14))
      #
      # pin 5
      #
      self.shape = add(self.shape,translate(pad_MTA,-.05,.1,0))
      self.pad.append(point(-.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'RS\n4',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.262,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.262,0,0))

class MTA_ICP(part):
   #
   # AMP 1445121-5
   # MTA .050 SMT 4-pin vertical
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: MISO
      #
      self.shape = translate(pad_MTA,-.1,-.1,0)
      self.pad.append(point(-.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'MISO',14))
      #
      # pin 2: GND
      #
      self.shape = add(self.shape,translate(pad_MTA,0,-.1,0))
      self.pad.append(point(0,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))
      #
      # pin 3: MOSI
      #
      self.shape = add(self.shape,translate(pad_MTA,.1,-.1,0))
      self.pad.append(point(.1,-.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'MOSI',14))
      #
```

```
      # pin 4: -RESET
      #
      self.shape = add(self.shape,translate(pad_MTA,.05,.1,0))
      self.pad.append(point(.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'-RESET',14))
      #
      # pin 5: SCK
      #
      self.shape = add(self.shape,translate(pad_MTA,-.05,.1,0))
      self.pad.append(point(-.05,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'SCK',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_MTA_solder,-.262,0,0))
      self.shape = add(self.shape,translate(pad_MTA_solder,.262,0,0))

class power_65mm(part):
   #
   # CUI PJ1-023-SMT
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: power
      #
      self.shape = cube(.433,.512,-.047,.047,0,0)
      self.pad.append(point(.467,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'P',14))
      #
      # pin 2: ground
      #
      self.shape = add(self.shape,cube(.285,.423,-.189,-.098,0,0))
      self.pad.append(point(.354,-.144,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 3: contact
      #
      self.shape = add(self.shape,cube(.325,.463,.098,.189,0,0))
      self.pad.append(point(.394,.144,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,cube(.108,.246,-.169,-.110,0,0))
      self.shape = add(self.shape,cube(.069,.207,.110,.169,0,0))

pad_stereo_2_5mm = cube(-.03,.03,-.05,.05,0,0)

class stereo_2_5mm(part):
   #
   # CUI SJ1-2533-SMT
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: base
      #
      self.shape = translate(pad_stereo_2_5mm,-.130,-.16,0)
      self.pad.append(point(-.130,-.149,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'base',14))
      #
      # pin 2: tip
      #
      self.shape = add(self.shape,translate(pad_stereo_2_5mm,.197,.15,0))
      self.pad.append(point(.197,.141,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'tip',14))
      #
      # pin 3: middle
      #
      self.shape = add(self.shape,translate(pad_stereo_2_5mm,-.012,-.16,0))
      self.pad.append(point(-.012,-.149,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'middle',14))

pad_Molex = cube(-.0155,.0155,-.0265,.0265,0,0)
pad_Molex_solder = cube(-.055,.055,-.065,.065,0,0)

class Molex_serial(part):
   #
   # Molex 53261-0471
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: Rx
      #
      self.shape = translate(pad_Molex,-.075,.064,0)
      self.pad.append(point(-.075,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Rx',14))
      #
      # pin 2: Tx
      #
      self.shape = add(self.shape,translate(pad_Molex,-.025,.064,0))
      self.pad.append(point(-.025,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Tx',14))
```

```
      #
      # pin 3: DTR
      #
      self.shape = add(self.shape,translate(pad_Molex,.025,.064,0))
      self.pad.append(point(.025,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DTR',14))
      #
      # pin 4: GND
      #
      self.shape = add(self.shape,translate(pad_Molex,.075,.064,0))
      self.pad.append(point(.075,.064,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))
      #
      # solder pads
      #
      self.shape = add(self.shape,translate(pad_Molex_solder,-.16,-.065,0))
      self.shape = add(self.shape,translate(pad_Molex_solder,.16,-.065,0))

#
# switches
#

pad_button_6mm = cube(-.04,.04,-.03,.03,0,0)

class button_6mm(part):
   #
   # Omron 6mm pushbutton
   # B3SN-3112P
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # left 1
      #
      self.shape = translate(pad_button_6mm,-.125,.08,0)
      self.pad.append(point(-.125,.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'L1',14))
      #
      # right 1
      #
      self.shape = add(self.shape,translate(pad_button_6mm,-.125,-.08,0))
      self.pad.append(point(-.125,-.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'R1',14))
      #
      # right 2
      #
      self.shape = add(self.shape,translate(pad_button_6mm,.125,-.08,0))
      self.pad.append(point(.125,-.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'R2',14))
      #
      # left 2
      #
      self.shape = add(self.shape,translate(pad_button_6mm,.125,.08,0))
      self.pad.append(point(.125,.08,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'L2',14))

#
# crystals and resonators
#

pad_XTAL = cube(-.016,.016,-.077,.077,0,0)

class XTAL(part):
   #
   # Panasonic EFOBM series
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # left
      #
      self.shape = translate(pad_XTAL,-.053,0,0)
      self.pad.append(point(-.053,0,0))
      #
      # ground
      #
      self.shape = add(self.shape,translate(pad_XTAL,0,0,0))
      self.pad.append(point(0,0,0))
      #
      # right
      #
      self.shape = add(self.shape,translate(pad_XTAL,.053,0,0))
      self.pad.append(point(.053,0,0))

#
# diodes, transistors, regulators
#

class D_1206(part):
   #
   # 1206 diode
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
```

```
      # anode
      #
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A',14))
      #
      # cathode
      #
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))

class LED_1206(part):
   #
   # 1206 LED
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # anode
      #
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A',14))
      #
      # cathode
      #
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))

class phototransistor_1206(part):
   #
   # 1206 phototransistor
   # OPTEK 520,521
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # collector
      #
      self.shape = translate(pad_1206,-.06,0,0)
      self.pad.append(point(-.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))
      #
      # emitter
      #
      self.shape = add(self.shape,translate(pad_1206,.06,0,0))
      self.pad.append(point(.055,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'E',14))

pad_SOD_123 = cube(-.02,.02,-.024,.024,0,0)

class D_SOD_123(part):
   #
   # SOD-123 diode
   #
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # anode
      #
      self.shape = translate(pad_SOD_123,-.07,0,0)
      self.pad.append(point(-.07,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A',14))
      #
      # cathode
      #
      self.shape = add(self.shape,translate(pad_SOD_123,.07,0,0))
      self.pad.append(point(.07,0,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'C',14))

pad_SOT23 = cube(-.015,.015,-.02,.02,0,0)

class NMOSFET_SOT23(part):
   #
   # Fairchild NDS355AN
   #
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: gate
      #
      self.shape = translate(pad_SOT23,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 2: source
      #
      self.shape = add(self.shape,translate(pad_SOT23,.0375,-.045,0))
```

```
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'S',14))
      #
      # pin 3: drain
      #
      self.shape = add(self.shape,translate(pad_SOT23,0,.045,0))
      self.pad.append(point(0,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'D',14))

class PMOSFET_SOT23(part):
   #
   # Fairchild NDS356AP
   #
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: gate
      #
      self.shape = translate(pad_SOT23,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 2: source
      #
      self.shape = add(self.shape,translate(pad_SOT23,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'S',14))
      #
      # pin 3: drain
      #
      self.shape = add(self.shape,translate(pad_SOT23,0,.045,0))
      self.pad.append(point(0,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'D',14))

class regulator_SOT23(part):
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: output
      #
      self.shape = translate(pad_SOT23,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: input
      #
      self.shape = add(self.shape,translate(pad_SOT23,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'in',14))
      #
      # pin 3: ground
      #
      self.shape = add(self.shape,translate(pad_SOT23,0,.045,0))
      self.pad.append(point(0,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'gnd',14))

pad_SOT223 = cube(-.02,.02,-.03,.03,0,0)
pad_SOT223_ground = cube(-.065,.065,-.03,.03,0,0)

class regulator_SOT223(part):
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: input
      #
      self.shape = translate(pad_SOT223,-.09,-.12,0)
      self.pad.append(point(-.09,-.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1I',14))
      #
      # pin 2: ground
      #
      self.shape = add(self.shape,translate(pad_SOT223,0,-.12,0))
      self.pad.append(point(0,-.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
      #
      # pin 3: output
      #
      self.shape = add(self.shape,translate(pad_SOT223,.09,-.12,0))
      self.pad.append(point(.09,-.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'O',14))
      #
      # pin 4: ground
      #
      self.shape = add(self.shape,translate(pad_SOT223_ground,0,.12,0))
      self.pad.append(point(0,.12,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'G',14))
```

```
pad_SM8 = cube(-.035,.035,-.016,.016,0,0)

class H_bridge_SM8(part):
   #
   # Zetex ZXMHC3A01T8
   #
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      d = .13
      #
      # pin 1: G3 (right N gate)
      #
      self.shape = translate(pad_SM8,-d,.09,0)
      self.pad.append(point(-d,.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GRN',14))
      #
      # pin 2: S2 S3 (N source)
      #
      self.shape = add(self.shape,translate(pad_SM8,-d,.03,0))
      self.pad.append(point(-d,.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'SN',14))
      #
      # pin 3: G2 (left N gate)
      #
      self.shape = add(self.shape,translate(pad_SM8,-d,-.03,0))
      self.pad.append(point(-d,-.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GLN',14))
      #
      # pin 4: G1 (left P gate)
      #
      self.shape = add(self.shape,translate(pad_SM8,-d,-.09,0))
      self.pad.append(point(-d,-.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GLP',14))
      #
      # pin 5: D1 D2 (left drain)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,-.09,0))
      self.pad.append(point(d,-.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DL',14))
      #
      # pin 6: S1 S4 (P source)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,-.03,0))
      self.pad.append(point(d,-.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'SP',14))
      #
      # pin 7: D3 D4 (right drain)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,.03,0))
      self.pad.append(point(d,.03,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'DR',14))
      #
      # pin 8: G4 (right N gate)
      #
      self.shape = add(self.shape,translate(pad_SM8,d,.09,0))
      self.pad.append(point(d,.09,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GRP',14))

#
# ICs
#

pad_SOT23_5 = cube(-.01,.01,-.02,.02,0,0)

class op_amp_SOT23_5(part):
   def __init__(self,value=''):
      self.value = value
      self.x = 0
      self.y = 0
      self.z = 0
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: output
      #
      self.shape = translate(pad_SOT23_5,-.0375,-.045,0)
      self.pad.append(point(-.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'O',14))
      #
      # pin 2: V-
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,0,-.045,0))
      self.pad.append(point(0,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V-',14))
      #
      # pin 3: I+
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,.0375,-.045,0))
      self.pad.append(point(.0375,-.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'I+',14))
      #
      # pin 4: I-
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,.0375,.045,0))
      self.pad.append(point(.0375,.045,0))
```

```
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'I-',14))
      #
      # pin 5: V+
      #
      self.shape = add(self.shape,translate(pad_SOT23_5,-.0375,.045,0))
      self.pad.append(point(-.0375,.045,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V+',14))

pad_SOICN = cube(-.035,.035,-.015,.015,0,0)

class op_amp_SOICN(part):
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: A out
      #
      self.shape = translate(pad_SOICN,-.12,.075,0)
      self.pad.append(point(-.12,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1 Ao',14))
      #
      # pin 2: A-
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.025,0))
      self.pad.append(point(-.12,.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A-',14))
      #
      # pin 3: A+
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.025,0))
      self.pad.append(point(-.12,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'A+',14))
      #
      # pin 4: V-
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,-.075,0))
      self.pad.append(point(-.12,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V-',14))
      #
      # pin 5: B+
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.075,0))
      self.pad.append(point(.12,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'B+',14))
      #
      # pin 6: B-
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,-.025,0))
      self.pad.append(point(.12,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'B-',14))
      #
      # pin 7: B out
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.025,0))
      self.pad.append(point(.12,.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'Bo',14))
      #
      # pin 8: V+
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.075,0))
      self.pad.append(point(.12,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'V+',14))

pad_SOIC = cube(-.043,.043,-.015,.015,0,0)

class ATtiny45_SOIC(part):
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []
      #
      # pin 1: PB5/dW/ADC0/-RESET/PCINT5
      #
      self.shape = translate(pad_SOIC,-.14,.075,0)
      self.pad.append(point(-.14,.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: PB3/ADC3/-OC1B/CLKI/XTAL1/PCINT3
      #
      self.shape = add(self.shape,translate(pad_SOIC,-.14,.025,0))
      self.pad.append(point(-.14,.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB3',14))
      #
      # pin 3: PB4/ADC2/OC1B/CLKO/XTAL2/PCINT4
      #
      self.shape = add(self.shape,translate(pad_SOIC,-.14,-.025,0))
      self.pad.append(point(-.14,-.025,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB4',14))
      #
      # pin 4: GND
      #
      self.shape = add(self.shape,translate(pad_SOIC,-.14,-.075,0))
      self.pad.append(point(-.14,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))
      #
      # pin 5: PB0/MOSI/DI/SDA/AIN0/OC0A/-OC1A/AREF/PCINT0
      #
      self.shape = add(self.shape,translate(pad_SOIC,.14,-.075,0))
      self.pad.append(point(.14,-.075,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
```

```
        #
        # pin 6: PB1/MISO/DO/AIN1/OC0B/OC1A/PCINT1
        #
        self.shape = add(self.shape,translate(pad_SOIC,.14,-.025,0))
        self.pad.append(point(.14,-.025,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB1',14))
        #
        # pin 7: PB2/SCK/USCK/SCL/ADC1/T0/INT0/PCINT2
        #
        self.shape = add(self.shape,translate(pad_SOIC,.14,.025,0))
        self.pad.append(point(.14,.025,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB2',14))
        #
        # pin 8: VCC
        #
        self.shape = add(self.shape,translate(pad_SOIC,.14,.075,0))
        self.pad.append(point(.14,.075,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'VCC',14))

class ATtiny44_SOICN(part):
   def __init__(self,value=''):
        self.value = value
        self.pad = [point(0,0,0)]
        self.labels = []
        #
        # pin 1: VCC
        #
        self.shape = translate(pad_SOICN,-.12,.15,0)
        self.pad.append(point(-.12,.15,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
        #
        # pin 2: PB0/XTAL1/PCINT8
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
        self.pad.append(point(-.12,.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
        #
        # pin 3: PB1/XTAL2/PCINT9
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,.050,0))
        self.pad.append(point(-.12,.05,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB1',14))
        #
        # pin 4: PB3/dW/-RESET/PCINT11
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,0,0))
        self.pad.append(point(-.12,0,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB3',14))
        #
        # pin 5: PB2/CKOUT/OC0A/INT0/PCINT10
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,-.05,0))
        self.pad.append(point(-.12,-.05,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB2',14))
        #
        # pin 6: PA7/ADC7/OC0B/ICP/PCINT7
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,-.1,0))
        self.pad.append(point(-.12,-.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA7',14))
        #
        # pin 7: PA6/ADC6/MOSI/SDA/OC1A/PCINT6
        #
        self.shape = add(self.shape,translate(pad_SOICN,-.12,-.15,0))
        self.pad.append(point(-.12,-.15,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA6',14))
        #
        # pin 8: PA5/ADC5/DO/MISO/OC1B/PCINT5
        #
        self.shape = add(self.shape,translate(pad_SOICN,.12,-.15,0))
        self.pad.append(point(.12,-.15,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA5',14))
        #
        # pin 9: PA4/ADC4/USCK/SCL/T1/PCINT4
        #
        self.shape = add(self.shape,translate(pad_SOICN,.12,-.1,0))
        self.pad.append(point(.12,-.1,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA4',14))
        #
        # pin 10: PA3/ADC3/T0/PCINT3
        #
        self.shape = add(self.shape,translate(pad_SOICN,.12,-.05,0))
        self.pad.append(point(.12,-.05,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA3',14))
        #
        # pin 11: PA2/ADC2/AIN1/PCINT2
        #
        self.shape = add(self.shape,translate(pad_SOICN,.12,0,0))
        self.pad.append(point(.12,0,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA2',14))
        #
        # pin 12: PA1/ADC1/AIN0/PCINT1
        #
        self.shape = add(self.shape,translate(pad_SOICN,.12,.050,0))
        self.pad.append(point(.12,.05,0))
        self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA1',14))
        #
        # pin 13: PA0/ADC0/AREF/PCINT0
        #
        self.shape = add(self.shape,translate(pad_SOICN,.12,.1,0))
        self.pad.append(point(.12,.1,0))
```

```
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PA0',14))
      #
      # pin 14: GND
      #
      self.shape = add(self.shape,translate(pad_SOICN,.12,.15,0))
      self.pad.append(point(.12,.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'GND',14))

pad_TQFP = cube(-.043,.043,-.015,.015,0,0)

class ATmega88_TQFP(part):
   def __init__(self,value=''):
      self.value = value
      self.pad = [point(0,0,0)]
      self.labels = []

      #
      # pin 1: PD3/PCINT19/OC2B/INT1
      #
      self.shape = translate(pad_SOICN,-.12,.15,0)
      self.pad.append(point(-.12,.15,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'1',14))
      #
      # pin 2: PD4/PCINT20/XCK/T0
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 3: GND
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 4: VCC
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 5: GND
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 6: VCC
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 7: PB6/PCINT6/XTAL1/TOSC1
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 8: PB7/PCINT7/XTAL2/TOSC2
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

      #
      # pin 9: PD5/PCINT21/OC0B/T1
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 10: PD6/PCINT22/OC0A/AIN0
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 11: PD7/PCINT23/AIN1
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 12: PB0/PCINT0/CLKO/ICP1
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 13: PB1/PCINT1/OC1A
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 14: PB2/PCINT2/-SS/OC1B
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
```

```
#
# pin 15: PB3/PCINT3/OC2A/MOSI
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 16: PB4/PCINT4/MISO
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

#
# pin 17: PB5/SCK/PCINT5
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 18: AVCC
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 19: ADC6
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 20: AREF
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 21: GND
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 22: ADC7
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 23: PC0/ADC0/PCINT8
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 24: PC1/ADC1/PCINT9
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

#
# pin 25: PC2/ADC2/PCINT10
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 26: PC2/ADC3/PCINT11
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 27: PC4/ADC4/SDA/PCINT12
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 28: PC5/ADC5/SCL/PCINT13
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 29: PC6/-RESET/PCINT14
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 30: PD0/RXD/PCINT16
#
self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
self.pad.append(point(-.12,.1,0))
self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
#
# pin 31: PD1/TXD/PCINT17
#
```

```
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))
      #
      # pin 32: PD2/INT0/PCINT18
      #
      self.shape = add(self.shape,translate(pad_SOICN,-.12,.1,0))
      self.pad.append(point(-.12,.1,0))
      self.labels.append(cad_text(self.pad[-1].x,self.pad[-1].y,self.pad[-1].z,'PB0',14))

#
# graphics
#

class CBA(part):
   def __init__(self,r=.02):
      self.value = ''
      self.pad = [point(0,0,0)]
      self.labels = []
      d = 3*r
      self.shape = circle(0,0,r)
      self.shape = add(self.shape,translate(circle(0,0,r),-d,d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),-d,0,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),-d,-d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),0,-d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),d,-d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),d,0,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),d,d,0))
      self.shape = add(self.shape,translate(rectangle(-r,r,-r,r),0,d,0))

class hello(part):
   #
   # HELLO
   #
   def __init__(self,w,z):
      self.value = ''
      self.pad = [point(0,0,0)]
      self.labels = []
      # H
      self.shape = rectangle(-.01,.01,0,.1)
      self.shape = add(self.shape,rectangle(0,.05,.04,.06))
      self.shape = add(self.shape,rectangle(.04,.06,0,.1))
      # E
      self.shape = add(self.shape,rectangle(.08,.1,0,.1))
      self.shape = add(self.shape,rectangle(.1,.14,0,.02))
      self.shape = add(self.shape,rectangle(.1,.13,.04,.06))
      self.shape = add(self.shape,rectangle(.1,.14,.08,.1))
      # L
      self.shape = add(self.shape,rectangle(.16,.18,0,.1))
      self.shape = add(self.shape,rectangle(.18,.21,0,.02))
      # L
      self.shape = add(self.shape,rectangle(.23,.25,0,.1))
      self.shape = add(self.shape,rectangle(.25,.28,0,.02))
      # 0
      self.shape = add(self.shape,rectangle(.3,.32,0,.1))
      self.shape = add(self.shape,rectangle(.32,.345,0,.02))
      self.shape = add(self.shape,rectangle(.32,.345,.08,.1))
      self.shape = add(self.shape,rectangle(.345,.365,0,.1))

#
# define board
#

w = .016
width = 1.66
height = 1.03
x = 1
y = 1
z = -.005
d = .06
mask = .004

pcb = PCB(x,y,width,height,mask)

IC1 = ATtiny44_SOICN('IC1\nt44')
pcb = IC1.add(pcb,x+.35,y+.52,z)

J1 = MTA_ICP('J1\nICP')
pcb = J1.add(pcb,IC1.x,IC1.pad[7].y-.21,z,angle=180)

pcb = wire(pcb,w,
   IC1.pad[7],
   J1.pad[3])

pcb = wire(pcb,w,
   IC1.pad[4],
   J1.pad[4])

pcb = wire(pcb,w,
   IC1.pad[14],
   J1.pad[2])

pcb = wire(pcb,w,
   IC1.pad[9],
   J1.pad[5])

pcb = wire(pcb,w,
   IC1.pad[8],
   J1.pad[1])

J2 = MTA_power('J2\npower')
```

```
pcb = J2.add(pcb,IC1.x-.03,IC1.pad[1].y+.2,z,angle=0)

pcb = wire(pcb,w,
   IC1.pad[14],
   J2.pad[1])

IC2 = regulator_SOT23('IC2\n5V')
pcb = IC2.add(pcb,IC1.x-.28,IC1.pad[1].y+.03,z,angle=180)

pcb = wire(pcb,w,
   IC1.pad[1],
   IC2.pad[1])

pcb = wire(pcb,w,
   J2.pad[2],
   IC2.pad[2])

pcb = wire(pcb,.014,
   point(J2.pad[1].x,J2.pad[1].y+.03,z),
   IC2.pad[3])

C1 = C_1206('C1\n3.3uF')
pcb = C1.add(pcb,IC2.x,IC2.y-.18,z,angle=90)

pcb = wire(pcb,w,
   C1.pad[1],
   IC2.pad[3])

pcb = wire(pcb,w,
   C1.pad[2],
   point(C1.x+.07,IC1.pad[1].y,z),
   IC1.pad[1])

R1 = R_1206('R1\n10k')
pcb = R1.add(pcb,C1.x,C1.y-.2,z,angle=90)

pcb = wire(pcb,w,
   R1.pad[1],
   C1.pad[2])

pcb = wire(pcb,w,
   R1.pad[2],
   point(J1.pad[3].x-.05,J1.pad[4].y,z),
   J1.pad[4])

dr = .085
dd = .115
dx = .058
dy = .2

#
# row 1
#

D11 = LED_1206('D11')
pcb = D11.add(pcb,IC1.x+.54,IC1.y+.39,z,angle=90)

R11 = R_1206('R11\n0')
pcb = R11.add(pcb,D11.x+dr,D11.y,z,angle=90)

pcb = wire(pcb,w,D11.pad[1],R11.pad[1])

D12 = LED_1206('D12')
pcb = D12.add(pcb,R11.x+dd,R11.y,z,angle=90)

pcb = wire(pcb,w,D11.pad[2],point(D11.x,D11.y,z),D12.pad[2])

R12 = R_1206('R12\n0')
pcb = R12.add(pcb,D12.x+dr,D12.y,z,angle=90)

pcb = wire(pcb,w,D12.pad[1],R12.pad[1])

D13 = LED_1206('D13')
pcb = D13.add(pcb,R12.x+dd,R12.y,z,angle=90)

pcb = wire(pcb,w,D12.pad[2],point(D12.x,D12.y,z),D13.pad[2])

R13 = R_1206('R13\n0')
pcb = R13.add(pcb,D13.x+dr,D13.y,z,angle=90)

pcb = wire(pcb,w,D13.pad[1],R13.pad[1])

D14 = LED_1206('D14')
pcb = D14.add(pcb,R13.x+dd,R13.y,z,angle=90)

pcb = wire(pcb,w,D13.pad[2],point(D13.x,D13.y,z),D14.pad[2])

R14 = R_1206('R14\n0')
pcb = R14.add(pcb,D14.x+dr,D14.y,z,angle=90)

pcb = wire(pcb,w,D14.pad[1],R14.pad[1])

#
# row 2
#

D21 = LED_1206('D21')
pcb = D21.add(pcb,D11.x,D11.y-dy,z,angle=90)

pcb = wire(pcb,w,D11.pad[2],D21.pad[1])
```

```
R21 = R_1206('R21\n0')
pcb = R21.add(pcb,D21.x+dr,D21.y,z,angle=90)

pcb = wire(pcb,w,R11.pad[2],point(R21.x+dx,R21.y,z))

pcb = wire(pcb,w,D21.pad[1],R21.pad[1])

D22 = LED_1206('D22')
pcb = D22.add(pcb,R21.x+dd,R21.y,z,angle=90)

pcb = wire(pcb,w,D21.pad[2],point(D21.x,D21.y,z),D22.pad[2])

R22 = R_1206('R22\n0')
pcb = R22.add(pcb,D22.x+dr,D22.y,z,angle=90)

pcb = wire(pcb,w,R22.pad[1],R12.pad[2])

pcb = wire(pcb,w,D22.pad[1],R22.pad[1])

D23 = LED_1206('D23')
pcb = D23.add(pcb,R22.x+dd,R22.y,z,angle=90)

pcb = wire(pcb,w,D22.pad[2],point(D22.x,D22.y,z),D23.pad[2])

R23 = R_1206('R23\n0')
pcb = R23.add(pcb,D23.x+dr,D23.y,z,angle=90)

pcb = wire(pcb,w,R23.pad[1],R13.pad[2])

pcb = wire(pcb,w,D23.pad[1],R23.pad[1])

D24 = LED_1206('D24')
pcb = D24.add(pcb,R23.x+dd,R23.y,z,angle=90)

pcb = wire(pcb,w,D23.pad[2],point(D23.x,D23.y,z),D24.pad[2])

R24 = R_1206('R24\n0')
pcb = R24.add(pcb,D24.x+dr,D24.y,z,angle=90)

pcb = wire(pcb,w,R24.pad[1],R14.pad[2])

pcb = wire(pcb,w,D24.pad[1],R24.pad[1])

#
# row 3
#

D31 = LED_1206('D31')
pcb = D31.add(pcb,D21.x,D21.y-dy,z,angle=90)

R31 = R_1206('R31\n0')
pcb = R31.add(pcb,D31.x+dr,D31.y,z,angle=90)

pcb = wire(pcb,w,D31.pad[1],R31.pad[1])

pcb = wire(pcb,w,R21.pad[2],R31.pad[1])

D32 = LED_1206('D32')
pcb = D32.add(pcb,R31.x+dd,R31.y,z,angle=90)

pcb = wire(pcb,w,D22.pad[2],D32.pad[1])

pcb = wire(pcb,w,D31.pad[2],point(D31.x,D31.y,z),D32.pad[2])

R32 = R_1206('R32\n0')
pcb = R32.add(pcb,D32.x+dr,D32.y,z,angle=90)

pcb = wire(pcb,w,R22.pad[2],point(R32.x+dx,R32.y,z))

pcb = wire(pcb,w,D32.pad[1],R32.pad[1])

D33 = LED_1206('D33')
pcb = D33.add(pcb,R32.x+dd,R32.y,z,angle=90)

pcb = wire(pcb,w,D32.pad[2],point(D32.x,D32.y,z),D33.pad[2])

R33 = R_1206('R33\n0')
pcb = R33.add(pcb,D33.x+dr,D33.y,z,angle=90)

pcb = wire(pcb,w,R33.pad[1],R23.pad[2])

pcb = wire(pcb,w,D33.pad[1],R33.pad[1])

D34 = LED_1206('D34')
pcb = D34.add(pcb,R33.x+dd,R33.y,z,angle=90)

pcb = wire(pcb,w,D33.pad[2],point(D33.x,D33.y,z),D34.pad[2])

R34 = R_1206('R34\n0')
pcb = R34.add(pcb,D34.x+dr,D34.y,z,angle=90)

pcb = wire(pcb,w,R34.pad[1],R24.pad[2])

pcb = wire(pcb,w,D34.pad[1],R34.pad[1])

#
# row 4
#

D41 = LED_1206('D41')
pcb = D41.add(pcb,D31.x,D31.y-dy,z,angle=90)
```

```
R41 = R_1206('R41\n0')
pcb = R41.add(pcb,D41.x+dr,D41.y,z,angle=90)

pcb = wire(pcb,w,D41.pad[1],R41.pad[1])

pcb = wire(pcb,w,R31.pad[2],R41.pad[1])

D42 = LED_1206('D42')
pcb = D42.add(pcb,R41.x+dd,R41.y,z,angle=90)

pcb = wire(pcb,w,D41.pad[2],point(D41.x,D41.y,z),D42.pad[2])

R42 = R_1206('R42\n0')
pcb = R42.add(pcb,D42.x+dr,D42.y,z,angle=90)

pcb = wire(pcb,w,R42.pad[1],R32.pad[2])

pcb = wire(pcb,w,D42.pad[1],R42.pad[1])

D43 = LED_1206('D43')
pcb = D43.add(pcb,R42.x+dd,R42.y,z,angle=90)

pcb = wire(pcb,w,D33.pad[2],D43.pad[1])

pcb = wire(pcb,w,D42.pad[2],point(D42.x,D42.y,z),D43.pad[2])

R43 = R_1206('R43\n0')
pcb = R43.add(pcb,D43.x+dr,D43.y,z,angle=90)

pcb = wire(pcb,w,R33.pad[2],point(R43.x+dx,R43.y,z))

pcb = wire(pcb,w,D43.pad[1],R43.pad[1])

D44 = LED_1206('D44')
pcb = D44.add(pcb,R43.x+dd,R43.y,z,angle=90)

pcb = wire(pcb,w,D43.pad[2],point(D43.x,D43.y,z),D44.pad[2])

R44 = R_1206('R44\n0')
pcb = R44.add(pcb,D44.x+dr,D44.y,z,angle=90)

pcb = wire(pcb,w,R44.pad[1],R34.pad[2])

pcb = wire(pcb,w,D44.pad[1],R44.pad[1])

#
# row 5
#

D51 = LED_1206('D51')
pcb = D51.add(pcb,D41.x,D41.y-dy,z,angle=90)

R51 = R_1206('R51\n0')
pcb = R51.add(pcb,D51.x+dr,D51.y,z,angle=90)

pcb = wire(pcb,w,D51.pad[1],R51.pad[1])

pcb = wire(pcb,w,R41.pad[2],R51.pad[1])

D52 = LED_1206('D52')
pcb = D52.add(pcb,R51.x+dd,R51.y,z,angle=90)

pcb = wire(pcb,w,D51.pad[2],point(D51.x,D51.y,z),D52.pad[2])

R52 = R_1206('R52\n0')
pcb = R52.add(pcb,D52.x+dr,D52.y,z,angle=90)

pcb = wire(pcb,w,R52.pad[1],R42.pad[2])

pcb = wire(pcb,w,D52.pad[1],R52.pad[1])

D53 = LED_1206('D53')
pcb = D53.add(pcb,R52.x+dd,R52.y,z,angle=90)

pcb = wire(pcb,w,D52.pad[2],point(D52.x,D52.y,z),D53.pad[2])

R53 = R_1206('R53\n0')
pcb = R53.add(pcb,D53.x+dr,D53.y,z,angle=90)

pcb = wire(pcb,w,R53.pad[1],R43.pad[2])

pcb = wire(pcb,w,D53.pad[1],R53.pad[1])

D54 = LED_1206('D54')
pcb = D54.add(pcb,R53.x+dd,R53.y,z,angle=90)

pcb = wire(pcb,w,D54.pad[1],D44.pad[2])

pcb = wire(pcb,w,D53.pad[2],point(D53.x,D53.y,z),D54.pad[2])

R54 = R_1206('R54\n0')
pcb = R54.add(pcb,D54.x+dr,D54.y,z,angle=90)

pcb = wire(pcb,w,R44.pad[2],point(R54.x+dx,R54.y,z),D54.pad[2])

pcb = wire(pcb,w,D54.pad[1],R54.pad[1])

#
# current-limiting resistors
#
```

```
dc = .14

R10 = R_1206('R10\n499')
pcb = R10.add(pcb,D11.x-dr,D11.y,z,angle=90)

pcb = wire(pcb,w,R10.pad[1],point(R10.x,R10.y,z),D11.pad[2])

R20 = R_1206('R20\n499')
pcb = R20.add(pcb,D21.x-dr,D21.y,z,angle=90)

pcb = wire(pcb,w,
   IC1.pad[12],
   point(R10.x-.2,R20.pad[2].y+.02,z),
   point(R10.x-.06,R10.pad[2].y,z),
   R10.pad[2])

pcb = wire(pcb,w,R20.pad[1],point(R20.x,R20.y,z),D21.pad[2])

pcb = wire(pcb,w,
   IC1.pad[11],
   point(R20.x-.1,R20.pad[2].y-.02,z),
   R20.pad[2])

R30 = R_1206('R30\n499')
pcb = R30.add(pcb,D31.x-dr,D31.y,z,angle=90)

pcb = wire(pcb,w,IC1.pad[10],R30.pad[2])

pcb = wire(pcb,w,R30.pad[1],point(R30.x,R30.y,z),D31.pad[2])

R40 = R_1206('R40\n499')
pcb = R40.add(pcb,D41.x-dr,D41.y,z,angle=90)

pcb = wire(pcb,w,R40.pad[1],point(R40.x,R40.y,z),D41.pad[2])

pcb = wire(pcb,w,R40.pad[2],point(R40.x-.2,IC1.pad[9].y,z),IC1.pad[9])

R50 = R_1206('R50\n499')
pcb = R50.add(pcb,D51.x-dr,D51.y,z,angle=90)

pcb = wire(pcb,w,R50.pad[1],point(R50.x,R50.y,z),D51.pad[2])

pcb = wire(pcb,w,R50.pad[2],J1.pad[1])

H1 = hello(w,z)
pcb = H1.add(pcb,J2.x+.03,J2.y-.16,z)

cad.function = add(pcb.board,pcb.exterior)

#
# uncomment to export traces
#

#cad.function = pcb.board

#
# uncomment to export exterior
#

#cad.function = pcb.exterior

#
# uncomment to export solder mask
#

#cad.function = pcb.mask

#
# uncomment to mill out board
#
# use 1/32 end-mill
# set tool diameter = .0312
# set xy, z speed = .5
# set # contours = 1
#

#cad.function = pcb.interior
#z = -.065

#
# define limits and parameters
#

cad.xmin = x-.1 # min x to render
cad.xmax = x+width+.1 # max x to render
cad.ymin = y-.1 # min y to render
cad.ymax = y+height+.1 # max y to render
cad.zmin = z
cad.zmax = .050
dpi = 200 # low resolution for previewing
#dpi = 500 # high resolution for machining
nxy = int(dpi*(cad.xmax-cad.xmin))
cad.nx = nxy # x points to render
cad.ny = nxy # y points to render
cad.nz = 1
cad.view('xy') # 2D view
```