**Laser Cut Linear Path Models**

There are a number of methods to translate a three-dimensional digital model into a physical model. Two basic methods can be used: subtractive and additive. The subtractive method uses CNC, computer numerically controlled, equipment which can remove material from a blank material mass. Additive methods assemble a model by adding material layer by layer. Rapid prototyping systems can create layers using wax, resin, plastic, starch, plaster, and even metal. Another method is to create layers by cutting two-dimensional materials using a laser cutter using the layer-by-layer concept.

Laser cutting requires that you develop a three-dimensional model by defining sections that are the thickness of the material used. Laser cutter can cut, actually burn, most common sheet materials, including plastic and wood.

The material thickness itself creates a texture that enhances the sectional concept as applied to architectural forms. The textured surface is an interesting alternative to completely smooth surfaces.

The previously discussed linear path example is an excellent candidate for laser cutting, as both are based on cross sections of the form. The digital model is created with a series of sections; a laser cutter can use these two-dimensional outlines for cutting.

Start a new drawing and set the units to architectural. Set the grid to 1 foot, and turn off all snaps and dynamic input.

The AutoLISP file, CH05A_LASERCUT.LSP, includes two functions to create laser cut models.

The first function, PROGLC01, is used to display an assembled three-dimensional version of the laser cut model. The model includes each laser cut section, a base, and text displaying the basic parameters for the model.

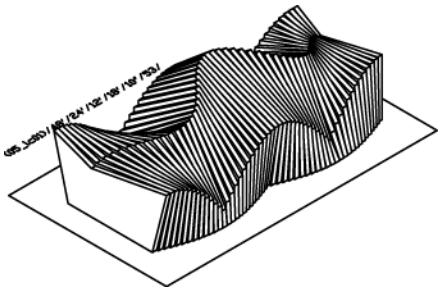

Figure 5.A01: PROGLC01, display three-dimensional version of laser cut model, isometric view

(05_lc01)/48'/24'/12'/18'/10"/53/

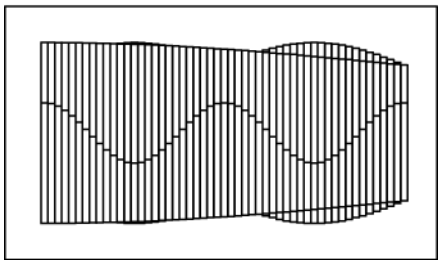

Figure 5.A02: PROGLC01, display three-dimensional version of laser cut model, plan view

The text includes script id, overall length and width, edge and midpoint heights, model length, and number of cuts.

Function PROGLC01 is based on PROG18 with a few added features.

Review the sample script file for PROGLC01 for a description of the required parameters. The top and bottom edge lengths, edge and midpoint heights, and midpoint offset can all be articulated. Additional parameters are included for the laser cut itself.

Laser cut parameters include: model id, model length, sheet dimensions, and thickness. For this example 3/16" formcore was used.

Sample script file, 05_LC01.SCR, for PROGLC01:

```
;-----------------------------------
; proglc01 view 3D
; proglc02 laser cut dwg
;
(proglc01)
; Script id
(05_lc01)
; Model id
A
; Length (ft)
48'
; Width (ft)
24'
; Edge height (ft)
12'
; Midpoint height (ft)
18'
; Bottom edge length:
; keep line factor
0.75
; start and end angle
0
360
; function (sin or cos)
sin
; Top edge length:
; keep line factor
0.75
; start and end angle
90
180
; function (sin or cos)
sin
; Edge height:
; keep line factor
0.25
; start and end angle
```

```
0
360
; function (sin or cos)
cos
; Midpoint height:
; keep line factor
0.5
; start and end angle
0
180
; function (sin or cos)
sin
; Midpoint offset (ft)
4'
; Midpoint offset
; start angle and end angle
0
720
; function (sin or cos)
cos
;
; Fillet radius (in)
9"
; Surface offset (in)
9"
; Negative offset, top and sides (in)
12"
; Negative offset, bottom (in)
12"
;
; Model and cut dimensions
;
; Model length (in)
10"
; Sheet thickness (in)
; 1/8=0.125 3/16=0.1875
0.1875"
; Sheet width (in)
20"
; Sheet length (in)
15"
;-----------------------------------
```

Load the CH05_LASERCUT.LSP file and execute this sample script, varying the model parameters.

Function PROGLC01 is based on function PROG18. It differs by expanding the options for start and end angle for each of five articulated dimensions; top and bottom lengths, edge and midpoint heights, and midpoint offset. These also include an option for which curve type to use: sine or cosine.

The selection of the curve type is implemented by defining a function at execution time for that curve, for example:

```
; set angle computations
(if (= (strcase eblenfunc) "SIN")
 (defun blfuncang (xrad) (sin xrad))
 (defun blfuncang (xrad) (cos xrad))
)
```

Other curve type variations could also be added, for example:

```
; set angle computations
(if (= (strcase eblenfunc) "SIN")
 (defun blfuncang (xrad) (sin xrad))
)
(if (= (strcase eblenfunc) "SINA")
 (defun blfuncang (xrad) (abs (sin xrad)))
)
(if (= (strcase eblenfunc) "SINAN")
 (defun blfuncang (xrad)
 (* (abs (sin xrad)) -1))
)
 (if (= (strcase eblenfunc) "COS”)
 (defun blfuncang (xrad) (cos xrad))
)
(if (= (strcase eblenfunc) "COSA")
 (defun blfuncang (xrad) (abs (cos xrad)))
)
(if (= (strcase eblenfunc) "COSAN")
 (defun blfuncang (xrad)
 (* (abs (cos xrad)) -1))
)
```

In this example the sine and cosine are defined for their positive, absolute, and negative values. The options would be entered as: SIN, SINA, SINAN, COS, COSA, or COSAN.

Another example of manging multiple curves can be found at the end of this section.

Each element of the model is placed on its own layer, for example:

```
; extrude 3DPOLY for 3D
(setvar "CLAYER" "3DSECT")
(command ".3DPOLY" epnt3 epnt1
 mpnt epnt2 epnt4 epnt3 "")
```

The system variable CLAYER is used to set the current layer. All the required layers are created at the start of the function.

The base of the model is created using the 3DFACE command and the displayed text using the TEXT command.

Predefine functions (c:vt) and (c:vsw) are used to set the model to top and isometric views.

The general outline for this function is:

```
;-----------------------------------
(defun proglc01 ()
 ; 3D view of laser cut model
 ; clear drwg and set layers
  .
 ; get boundary points and input
 ; parameters
  .
 ; set model parms
  .
 ; set angle computations
  .
 ; start start angs
  .
 ; start pts for sections
  .
 ; loop to repeat sections
 (repeat numtimes
  ; compute bottom endpoint
   .
  ; compute top endpoint
   .
  ; compute outside height
   .
  ; compute midpoint offset
   .
  ; compute midpoint height
   .
  ; extrude 3DPOLY for 3D
   .
  ; inc x coord and angs
 )
 ; base 3D
  .
 ; id text
  .
 ; reset layers
  .
)
;-----------------------------------
```

Completed function PROGLC01:

```
;-----------------------------------
(defun proglc01 ()
 ; 3D view of laser cut model
 (setvar "CMDECHO" 0)
 ; clear drwg
 (command ".LAYER" "SET" "0" "")
 (command ".LAYER"
  "UNLOCK" "2D*,3D*"
  "THAW" "2D*,3D*"
  "ON" "2D*,3D*" "")
 (c:vt)
 (command ".ZOOM" "e")
 (command ".LAYER" "MAKE" "3DSECT"
  "MAKE" "3DBASE" "MAKE" "3DID" "")
 (command ".LAYER" "MAKE" "2DTEXT"
  "MAKE" "2DCUT" "MAKE" "2DLAYOUT" "")
 (command ".LAYER" "SET" "0" "")
 (command ".LAYER" "COLOR"
  "7" "*" "")
 (command ".LAYER" "COLOR"
  "1" "2DCUT" "")
 (command ".LAYER" "COLOR"
  "3" "2DTEXT" "")
 (command ".LAYER"
  "OFF" "2D*"
  "FREEZE" "2D*"
  "LOCK" "2D*"  "")
 (command ".LAYER" "ON" "3D*" "")
 (command ".ERASE" "ALL" "")
 ; get boundary points and parameters
 (prompt
  "\nPROGLC01 - 3D laser cut model")
 ; start
 (setq spnt (list 0.0 0.0 0.0))
 ; script and model ids
 (setq scriptid (getstring))
 (setq modelid (getstring))
 (setq modelid
  (substr (strcase modelid) 1 1))
 ; overall dimensions
 (setq clen (getdist))
 (setq cwid (getdist))
 (setq zedge (getdist))
 (setq zmid (getdist))
 ; bottom edge length
 (setq eblenfact (getreal))
 (setq eblsang (getreal))
 (setq ebleang (getreal))
 (setq eblenfunc (getstring))
 ; top edge length
 (setq etlenfact (getreal))
 (setq etlsang (getreal))
 (setq etleang (getreal))
 (setq etlenfunc (getstring))
 ; edge height
 (setq ehgtfact (getreal))
 (setq ehsang (getreal))
 (setq eheang (getreal))
 (setq ehgtfunc (getstring))
 ; midpoint height
 (setq mhgtfact (getreal))
 (setq mhsang (getreal))
 (setq mheang (getreal))
 (setq mhgtfunc (getstring))
 ; midpoint offset
 (setq zmidoff (getdist))
 (setq mhoffsang (getreal))
 (setq mhoffeang (getreal))
 (setq mhgtofffunc (getstring))
 ; fillet radius
 (setq frad (getdist))
 ; frame offsets
 (setq foff (getdist))
 (setq noff (getdist))
 (setq nboff (getdist))
 ; model dimensions
 (setq modellen (getdist))
 (setq bthick (getdist))
 (setq bwid (getdist))
 (setq blen (getdist))
 ;
 ; set model parms
 (setq numtimes
  (fix (/ modellen bthick)))
 ; model scale
 (setq mscale
  (/ modellen clen))
 (setq mclen
  (* clen mscale))
 (setq mcwid
  (* cwid mscale))
 (setq cpnt
  (list mclen (/ mcwid 2.0) 0.0))
 (setq mzedge
  (* zedge mscale))
 (setq mzmid
  (* zmid mscale))
 (setq mzmidoff
  (* zmidoff mscale))
 (setq mfrad (* frad mscale))
 (setq mfoff (* foff mscale))
 (setq mnoff (* noff mscale))
 (setq mnboff (* nboff mscale))
 ;
 ; compute line length, width of box
 (setq linelen
  (- (nth 1 cpnt) (nth 1 spnt)))
 ; compute increment across box
 (setq xinc
  (/ (- (nth 0 cpnt) (nth 0 spnt))
  (- numtimes 1)))
 ; computer ang inc
 (setq blanginc
  (/ (- ebleang eblsang)
  (- numtimes 1)))
 (setq tlanginc
  (/ (- etleang etlsang)
  (- numtimes 1)))
 (setq eanginc
  (/ (- eheang ehsang)
  (- numtimes 1)))
 (setq manginc
  (/ (- mheang mhsang)
  (- numtimes 1)))
 (setq oanginc
  (/ (- mhoffeang mhoffsang)
  (- numtimes 1)))
 ; set first point and first x coord
 (setq xpnt spnt)
 (setq xpt (nth 0 spnt))
 ;
 ; set angle computations
 (if (= (strcase eblenfunc) "SIN")
  (defun blfuncang (xrad) (sin xrad))
  (defun blfuncang (xrad) (cos xrad))
 )
 (if (= (strcase etlenfunc) "SIN")
  (defun tlfuncang (xrad) (sin xrad))
  (defun tlfuncang (xrad) (cos xrad))
 )
 (if (= (strcase ehgtfunc) "SIN")
  (defun efuncang (xrad) (sin xrad))
  (defun efuncang (xrad) (cos xrad))
 )
 (if (= (strcase mhgtfunc) "SIN")
  (defun mfuncang (xrad) (sin xrad))
  (defun mfuncang (xrad) (cos xrad))
 )
 (if (= (strcase mhgtofffunc) "SIN")
  (defun ofuncang (xrad) (sin xrad))
  (defun ofuncang (xrad) (cos xrad))
 )
 ; start start angs
 (setq blang eblsang)
 (setq tlang etlsang)
 (setq eang ehsang)
 (setq mang mhsang)
 (setq oang mhoffsang)
 ; set extrude height
 (command ".ELEV" 0.0 0.0)
 ;
 ; start pts for sections
 (setq sxcpnt (- (nth 0 spnt) 1.0))
 (setq xcpnt sxcpnt)
 (setq ycpnt
  (- (nth 1 cpnt)
```

```
 (+ (* linelen 3.0) 1.1)))
; loop to repeat sections
(setq nsect 0)
(repeat numtimes
 ; compute bottom endpoint
 (setq line1 (* linelen eblenfact))
 (setq line2 (* (* (* linelen
  (- 1.0 eblenfact))
  (abs (blfuncang (dtr blang))))))
 (setq newlen (+ line1 line2))
 (setq epnt3
  (polar xpnt (dtr 90) newlen))
 (setq epnt4
  (polar xpnt (dtr 270) newlen))
 ; compute top endpoint
 (setq line1 (* linelen etlenfact))
 (setq line2 (* (* (* linelen
  (- 1.0 etlenfact))
  (abs (tlfuncang (dtr tlang))))))
 (setq newlen (+ line1 line2))
 (setq epnt1
  (polar xpnt (dtr 90) newlen))
 (setq epnt2
  (polar xpnt (dtr 270) newlen))
 ; compute outside height
 (setq line1 (* mzedge ehgtfact))
 (setq line2 (* (* (* mzedge
  (- 1.0 ehgtfact))
  (abs (efuncang (dtr eang))))))
 (setq newzedge (+ line1 line2))
 (setq epnt1 (list
  (nth 0 epnt1) (nth 1 epnt1)
  newzedge))
 (setq epnt2 (list
  (nth 0 epnt2) (nth 1 epnt2)
  newzedge))
 ; compute midpoint offset
 (setq newzmidoff (* (* mzmidoff
  (ofuncang (dtr oang)))))
 (setq mpnt (list
  (nth 0 xpnt) (+ (nth 1 xpnt)
  newzmidoff) 0.0))
 ; compute midpoint height
 (setq line1 (* mzmid mhgtfact))
 (setq line2 (* (* (* mzmid
  (- 1.0 mhgtfact))
  (abs (mfuncang (dtr mang))))))
 (setq newzmid (+ line1 line2))
 (setq mpnt (list
  (nth 0 mpnt) (nth 1 mpnt)
  newzmid))
 ; inc section
 (setq nsect (+ nsect 1))
 ; extrude 3DPOLY for 3D
 (setvar "CLAYER" "3DSECT")
 (command ".3DPOLY" epnt3 epnt1
  mpnt epnt2 epnt4 epnt3 "")
 (command ".EXTRUDE" "LAST" ""
  bthick )
 ; versions prior to 2007 use
 ;(command ".EXTRUDE" "LAST" ""
 ; bthick "0" )
 (command ".ZOOM" "E")
 ; inc x coord
 (setq xpt (+ xpt xinc))
 (setq xpnt
  (cons xpt (cdr xpnt)))
 ; inc angs
 (setq blang
  (+ blang blanginc))
 (setq tlang
  (+ tlang tlanginc))
 (setq eang
  (+ eang eanginc))
 (setq mang
  (+ mang manginc))
 (setq oang
  (+ oang oanginc))
)
; base 3D
(setq epnt2 (list
 (- (nth 0 spnt) 1.0)
 (+ (nth 1 cpnt) 1.0) 0.0))
(setq epnt3 (list
  (+ (nth 0 cpnt) 1.0)
  (+ (nth 1 cpnt) 1.0) 0.0))
 (setq epnt4 (list
  (+ (nth 0 cpnt) 1.0)
  (* (+ (nth 1 cpnt) 1.0)
  -1.0) 0.0))
 (setq epnt5 (list
  (- (nth 0 spnt) 1.0)
  (* (+ (nth 1 cpnt) 1.0)
  -1.0) 0.0))
 (setvar "CLAYER" "3DBASE")
 (command ".3DFACE" epnt2 epnt3
  epnt4 epnt5  "")
 (command ".ZOOM" "e")
 ; id text
 (setq epnt1 (list
  (+ (nth 0 spnt) 0.5)
  (+ (nth 1 cpnt) 2.5) 0.0))
 (setvar "CLAYER" "3DID")
 (command ".TEXT" epnt1 (/ 1.0 4.0)
  "0" (strcat scriptid "/"
  (rtos clen 4 2) "/"
  (rtos cwid 4 2 ) "/"
  (rtos zedge 4 2 ) "/"
  (rtos zmid 4 2 ) "/"
  (rtos modellen 4 2 ) "/"
  (itoa numtimes) "/" )
 )
 ; reset layers
 (command ".LAYER" "SET" "0" "")
 (c:vsw)
 (command ".HIDE")
 (princ)
)
;-----------------------------------
```

Once the model parameters are determined, a two-dimensional drawing of the sections to be laser cut can be produced.

Using the same script file as for PROGLC01, change the function call from (proglc01) to (proglc02). Run the modified script file.

This second function, will create on separate layers the laser cut sections.

This laser cut function was developed to take full advantage of the section data being computed. Three laser cut models are created: the first, the innermost, representing a solid volume of the form, the second, the surface of the volume, and third the negative of the volume. This represents the inside, in-between, and outside of the model. This is accomplished by simply nesting the three laser cuts.

The script file contains the following options for laser cutting:

```
;-----------------------------------
; Fillet radius (in)
9"
; Surface offset (in)
9"
; Negative offset, top and sides (in)
12"
; Negative offset, bottom (in)
12"
;
; Model and cut dimensions
;
; Model length (in)
10"
; Sheet thickness (in)
; 1/8=0.125 3/16=0.1875
0.1875"
; Sheet width (in)
20"
; Sheet length (in)
```

```
15"
;---------------------------------
```

The fillet radius, surface offset, and negative offsets are all optional and can be set to zero.

Each section is placed next to each other using the sheet size as a layout guide. The sheet outline is included, as well as, text displaying the basic model data. The total length of the laser cut is also displayed so the total amount of material can be estimated.

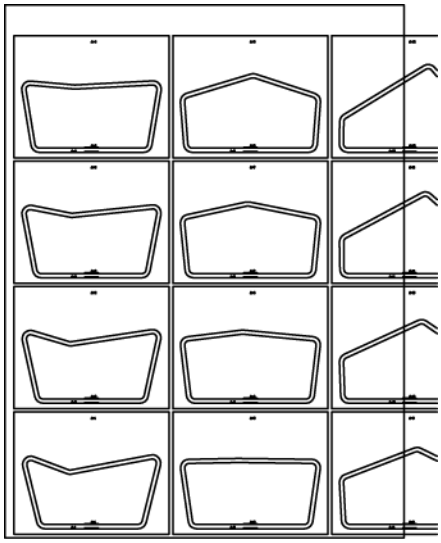

Figure 5.A03: PROGLC02, sample of the first set of laser cuts

For each laser cut section, the Model Id and section number is included on each model part. Also included are cutouts which are used to register each section.

All the cuts are assigned the color red and all the text the color green. These colors can be used to set the laser cutter to cut and etch.

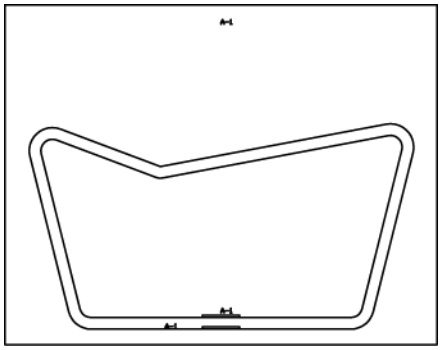

Figure 5.A04: PROGLC02, sample of a typical laser cut section

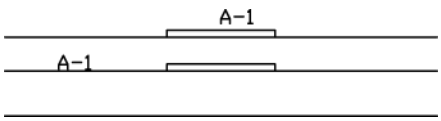


Figure 5.A05: PROGLC02, sample of the section text and registration marks

Function PROGLC02 is based on PROGLC01. It differs in that it creates each section in plan, not in elevation, and includes the extra options for the layout and lines for each part of the laser cut. The section is drawn by the PLINE command, the rounding of the corners the FILLET command, the outside surface OFFSET command, and the negative boundary by the PLINE command.

The general outline for this function is:

```
;-----------------------------------
(defun proglc02 ()
 ; laser cut drwg
 ; three part section cut
 ; clear drwg and set layers
  .
 ; get boundary points and input
 ; parameters
  .
 ; set model parms
  .
 ; set angle computations
  .
 ; start start angs
  .
 ; start pts for sections
  .
 ; loop to repeat sections
 (repeat numtimes
  ; compute bottom endpoint
   .
  ; compute top endpoint
   .
  ; compute outside height
   .
  ; compute midpoint offset
   .
  ; compute midpoint height
   .
  ; center slot
   .
  ; PLINE for 2D
   .
  ; fillet corners
   .
  ; duplicate by offset
   .
  ; top and bottom tab cut
   .
  ; negative boundary
   .
  ; add sect id and numbers
   .
  ; check for next cut location
   .
  ; inc angs
 )
 ; layout boundary
  .
 ; model id text
  .
 ; reset layers
  .
)
;-----------------------------------
```

Completed function PROGLC02:

```
;-----------------------------------
(defun proglc02 ()
 ; laser cut drwg
 ; three part section cut
 ; red = cut
 ; green = etch
 (setvar "CMDECHO" 0)
 ; clear drwg
 (command ".LAYER" "SET" "0" "")
 (command ".LAYER"
  "UNLOCK" "2D*,3D*"
```

```
 "THAW" "2D*,3D*"
 "ON" "2D*,3D*" "")
(c:vt)
(command ".ZOOM" "e")
(command ".LAYER" "MAKE" "3DSECT"
 "MAKE" "3DBASE" "MAKE" "3DID" "")
(command ".LAYER" "MAKE" "2DTEXT"
 "MAKE" "2DCUT" "MAKE" "2DLAYOUT" "")
(command ".LAYER" "SET" "0" "")
(command ".LAYER" "COLOR"
 "7" "*" "")
(command ".LAYER" "COLOR"
 "1" "2DCUT" "")
(command ".LAYER" "COLOR"
 "3" "2DTEXT" "")
(command ".LAYER"
 "OFF" "3D*"
 "FREEZE" "3D*"
 "LOCK" "3D*"  "")
(command ".LAYER"
 "ON" "2D*" "")
(command ".ERASE" "ALL" "")
; get boundary points and parameters
(prompt
 "\nPROGLC02 - laser cut drwg")
; start
(setq spnt (list 0.0 0.0 0.0))
; script and model ids
(setq scriptid (getstring))
(setq modelid (getstring))
(setq modelid
 (substr (strcase modelid) 1 1))
; overall dimensions
(setq clen (getdist))
(setq cwid (getdist))
(setq zedge (getdist))
(setq zmid (getdist))
; bottom edge length
(setq eblenfact (getreal))
(setq eblsang (getreal))
(setq ebleang (getreal))
(setq eblenfunc (getstring))
; top edge length
(setq etlenfact (getreal))
(setq etlsang (getreal))
(setq etleang (getreal))
(setq etlenfunc (getstring))
; edge height
(setq ehgtfact (getreal))
(setq ehsang (getreal))
(setq eheang (getreal))
(setq ehgtfunc (getstring))
; midpoint height
(setq mhgtfact (getreal))
(setq mhsang (getreal))
(setq mheang (getreal))
(setq mhgtfunc (getstring))
; midpoint offset
(setq zmidoff (getdist))
(setq mhoffsang (getreal))
(setq mhoffeang (getreal))
(setq mhgtofffunc (getstring))
; fillet radius
(setq frad (getdist))
; frame offsets
(setq foff (getdist))
(setq noff (getdist))
(setq nboff (getdist))
; model dimensions
(setq modellen (getdist))
(setq bthick (getdist))
(setq bwid (getdist))
(setq blen (getdist))
;
; set model parms
(setq numtimes
 (fix (/ modellen bthick)))
; model scale
(setq mscale
 (/ modellen clen))
(setq mclen
 (* clen mscale))
(setq mcwid
 (* cwid mscale))
(setq cpnt
 (list mclen (/ mcwid 2.0) 0.0))
(setq mzedge
 (* zedge mscale))
(setq mzmid
 (* zmid mscale))
(setq mzmidoff
 (* zmidoff mscale))
(setq mfrad (* frad mscale))
(setq mfoff (* foff mscale))
(setq mnoff (* noff mscale))
(setq mnboff (* nboff mscale))
;
; compute line length, width of box
(setq linelen
 (- (nth 1 cpnt) (nth 1 spnt)))
; compute increment across box
(setq xinc
 (/ (- (nth 0 cpnt) (nth 0 spnt))
 (- numtimes 1)))
; computer ang inc
(setq blanginc
 (/ (- ebleang eblsang)
 (- numtimes 1)))
(setq tlanginc
 (/ (- etleang etlsang)
 (- numtimes 1)))
(setq eanginc
 (/ (- eheang ehsang)
 (- numtimes 1)))
(setq manginc
 (/ (- mheang mhsang)
 (- numtimes 1)))
(setq oanginc
 (/ (- mhoffeang mhoffsang)
 (- numtimes 1)))
; set first point
(setq xpnt spnt)
;
; set angle computations
(if (= (strcase eblenfunc) "SIN")
 (defun blfuncang (xrad) (sin xrad))
 (defun blfuncang (xrad) (cos xrad))
)
(if (= (strcase etlenfunc) "SIN")
 (defun tlfuncang (xrad) (sin xrad))
 (defun tlfuncang (xrad) (cos xrad))
)
(if (= (strcase ehgtfunc) "SIN")
 (defun efuncang (xrad) (sin xrad))
 (defun efuncang (xrad) (cos xrad))
)
(if (= (strcase mhgtfunc) "SIN")
 (defun mfuncang (xrad) (sin xrad))
 (defun mfuncang (xrad) (cos xrad))
)
(if (= (strcase mhgtofffunc) "SIN")
 (defun ofuncang (xrad) (sin xrad))
 (defun ofuncang (xrad) (cos xrad))
)
; start start angs
(setq blang eblsang)
(setq tlang etlsang)
(setq eang ehsang)
(setq mang mhsang)
(setq oang mhoffsang)
; set extrude height
(command ".ELEV" 0.0 0.0)
;
; loop to repeat sections
(setq nsect 0)
(repeat numtimes
 ; compute bottom endpoint
 (setq line1 (* linelen eblenfact))
 (setq line2 (* (* (* linelen
  (- 1.0 eblenfact))
  (abs (blfuncang (dtr blang))))))
 (setq newlen (+ line1 line2))
 (setq epnt1
  (polar xpnt (dtr 90) newlen))
 (setq epnt2
  (polar xpnt (dtr 270) newlen))
 ; compute top endpoint
 (setq line1 (* linelen etlenfact))
```

```
(setq line2 (* (* (* linelen
 (- 1.0 etlenfact))
 (abs (tlfuncang (dtr tlang))))))
(setq newlen (+ line1 line2))
(setq epnt3
 (polar xpnt (dtr 90) newlen))
(setq epnt4
 (polar xpnt (dtr 270) newlen))
; compute outside height
(setq line1 (* mzedge ehgtfact))
(setq line2 (* (* (* mzedge
 (- 1.0 ehgtfact))
 (abs (efuncang (dtr eang))))))
(setq newzedge (+ line1 line2))
(setq epnt3 (list
 (+ (nth 0 epnt3) newzedge)
 (nth 1 epnt3) ))
(setq epnt4 (list
 (+ (nth 0 epnt4) newzedge)
 (nth 1 epnt4) ))
; compute midpoint offset
(setq newzmidoff
 (* (* mzmidoff
 (ofuncang (dtr oang)))))
(setq epnt5 (list
 (nth 0 xpnt) (+ (nth 1 xpnt)
 newzmidoff)))
; compute midpoint height
(setq line1 (* mzmid mhgtfact))
(setq line2 (* (* (* mzmid
 (- 1.0 mhgtfact))
 (abs (mfuncang (dtr mang))))))
(setq newzmid (+ line1 line2))
(setq epnt5
 (list (+ (nth 0 epnt5) newzmid)
 (nth 1 epnt5) ))
; center slot
(setq xpntc (list
 (nth 0 xpnt) (nth 1 xpnt) ))
(setq epnt6 (list
 (nth 0 xpntc)
 (+ (nth 1 xpntc) (/ 1.0 4.0)) ))
(setq epnt7 (list
 (+ (nth 0 xpntc) (/ 1.0 32.0))
 (+ (nth 1 xpntc) (/ 1.0 4.0)) ))
(setq epnt10 (list
 (+ (nth 0 xpntc) (/ 1.0 32.0))
 (nth 1 xpntc) ))
(setq epnt8 (list
 (+ (nth 0 xpntc) (/ 1.0 32.0))
 (- (nth 1 xpntc) (/ 1.0 4.0)) ))
(setq epnt9 (list
 (nth 0 xpntc)
 (- (nth 1 xpntc) (/ 1.0 4.0)) ))
; inc section
(setq nsect (+ nsect 1))
;
; pline for 2D
(setvar "CLAYER" "2DCUT")
(command ".PLINE" epnt6 epnt1
 epnt3 epnt5 epnt4 epnt2 epnt9 "C")
(command ".ZOOM" "E" ".ZOOM" "0.6x")
;
; fillet corners
(if (> mfrad 0.0) (progn
 (command ".FILLET" "R" mfrad)
 (command ".FILLET" "P" epnt1)
))
; duplicate by offset
(if (> mfoff 0.0) (progn
 (setq opnt (list
  (- (nth 0 epnt6) mfoff)
  (nth 1 epnt6)))
 (command ".OFFSET" mfoff epnt6
  opnt "")
))
; top tab cut
(command ".PLINE" epnt6 epnt7
 epnt10 epnt8 epnt9 "")
; bottom tab cut
(if (> mfoff 0.0) (progn
 (setq epnt6 (list
  (- (nth 0 xpntc) mfoff)
  (+ (nth 1 xpntc) (/ 1.0 4.0)) ))
 (setq epnt7 (list
  (+ (- (nth 0 xpntc) mfoff)
  (/ 1.0 32.0))
  (+ (nth 1 xpntc) (/ 1.0 4.0)) ))
 (setq epnt10 (list
  (+ (- (nth 0 xpntc) mfoff)
  (/ 1.0 32.0)) (nth 1 xpntc) ))
 (setq epnt8 (list
  (+ (- (nth 0 xpntc) mfoff)
  (/ 1.0 32.0))
  (- (nth 1 xpntc) (/ 1.0 4.0)) ))
 (setq epnt9 (list
  (- (nth 0 xpntc) mfoff)
  (- (nth 1 xpntc) (/ 1.0 4.0)) ))
 (command ".PLINE" epnt6 epnt7
  epnt10 epnt8 epnt9 "")
))
(command ".ZOOM" "E" ".ZOOM" "0.6x")
; negative frame
(if (> mnoff 0.0) (progn
 (setq xpnt1 (list
  (- (nth 0 xpntc) (+ mfoff mnboff))
  (+ (+ (nth 1 xpntc) (/ mcwid 2))
  (* mnoff 2)) ))
 (setq xpnt2 (list
  (- (nth 0 xpntc) (+ mfoff mnboff))
  (- (- (nth 1 xpntc) (/ mcwid 2))
  (* mnoff 2)) ))
 (setq xpnt3 (list
  (+ (nth 0 xpnt1)
  (+ (max mzedge mzmid)
  (+ (* mnoff 3) mnboff)))
  (nth 1 xpnt1) ))
 (setq xpnt4 (list
  (+ (nth 0 xpnt2)
  (+ (max mzedge mzmid)
  (+ (* mnoff 3) mnboff)))
  (nth 1 xpnt2) ))
 (command ".PLINE" xpnt1 xpnt3
  xpnt4 xpnt2 xpnt1 "")
))
; add sect number
; positive
(command ".TEXT"
 (list (+ (nth 0 xpntc) (/ 1.0 16.0))
 (nth 1 xpntc)) (/ 1.0 16.0) "-90"
 (strcat modelid "-" (itoa nsect)))
(command ".CHPROP" "last" ""
 "LAYER" "2DTEXT" "")
; positive
(if (> mfoff 0.0) (progn
 (setvar "CLAYER" "2DTEXT")
 (command ".TEXT" (list
  (+ (- (nth 0 xpntc) mfoff)
  (/ 1.0 128.0))
  (+ (nth 1 xpntc) (/ 3.0 4.0)))
  (/ 1.0 16.0) "-90"
  (strcat modelid "-" (itoa nsect)))
))
; negative
(if (> mnoff 0.0) (progn
 (setvar "CLAYER" "2DTEXT")
 (command ".TEXT" (list
  (+ (nth 0 xpntc)
  (+ (max mzedge mzmid)
  (* mnoff 1))) (nth 1 xpntc))
  (/ 1.0 16.0) "-90"
  (strcat modelid "-" (itoa nsect)))
))
(command ".ZOOM" "E" ".ZOOM" "0.6x")
; next location
(setq xnext (+ (+ 0 (+
 (max mzedge mzmid)
 (+ (* mnoff 3) mnboff)))
 (/ 1.0 8.0)) )
(setq ynext (+ (+
 (/ mcwid 1)
 (* (+ mnoff mfoff) 2))
 (/ 1.0 4.0)))
(setq xpnt (list
 (+ (nth 0 xpnt) xnext)
 (nth 1 xpnt) ))
(if (> (+ (nth 0 xpnt) xnext)
 (+ (nth 0 spnt) bwid)) (progn
```

```
   (setq xpnt (list
    (nth 0 spnt)
    (- (nth 1 xpnt) ynext)))
  ))
  ; inc ang
  (setq blang (+ blang blanginc))
  (setq tlang (+ tlang tlanginc))
  (setq eang (+ eang eanginc))
  (setq mang (+ mang manginc))
  (setq oang (+ oang oanginc))
 )
 ; layout 2D
 (command ".LAYER"
  "SET" "2DLAYOUT" "")
 (setq epnt1 (list
  (- (nth 0 spnt) 0.5)
  (+ (nth 1 cpnt) 0.75) 0.0))
 (setq epnt2 (list
  (+ (nth 0 epnt1) bwid)
  (nth 1 epnt1) 0.0))
 (setq epnt3 (list
  (+ (nth 0 epnt1) bwid)
  (- (nth 1 epnt1) blen) 0.0))
 (setq epnt4 (list
  (- (nth 0 spnt) 0.5)
  (nth 1 epnt3) 0.0))
 (command ".PLINE" epnt1 epnt2
  epnt3 epnt4 epnt1 "")
 ; cut length
 (setq minpt (getvar "EXTMIN"))
 (setq maxpt (getvar "EXTMAX"))
 (setq cutlen
  (fix (+ 0.5 (abs (- (nth 1 minpt)
  (nth 1 maxpt))))))
 ; id text
 (setq epnt1 (list
  (- (nth 0 spnt) 2.0)
  (nth 1 epnt3) 0.0))
 (command ".TEXT" epnt1
  (/ 1.0 4.0) "90"
  (strcat scriptid "/"
  (rtos clen 4 2) "/"
  (rtos cwid 4 2 ) "/"
  (rtos zedge 4 2 ) "/"
  (rtos zmid 4 2 ) "/"
  (rtos modellen 4 2 ) "/"
  (itoa numtimes) "/"
  (rtos bwid 4 2) "x"
  (rtos cutlen 4 2) "/")
 )
 ; reset layers
 (command ".LAYER" "SET" "0" "")
 (c:vt)
 (princ)
)
;-----------------------------------
```

For the laser cuts consider the variety of materials and thicknesses available. The thinner the material the more cuts will have to be made but the smoother the model will appear.

When cutting formcore a slight burn is visible on the edges and the foam shrinks a bit due to the heat; an interesting texture emerges.

When translucent or transparent plastics are cut, the generated curves are visible along the side of the model when assembled.

The negative of the section can be used to represent a carved out volume or it can be used as a mold for casting if one of the ends is closed.

Consider other approaches: does every section need to be cut, how can spacers between section be generated.

The laser cuts in this section were based on the section being computed, consider the approach that would be required if a solid model was created. How could the sections be extracted from a solid model?

**Specifying Multiple Curves**

The previous method discussed works for to select a small set of possible curves. When a larger number of fixed curves are needed, another method can be used. Each curve can be assigned a type and number, such as, S01, S02, C01, or C02; for sine 01 and 02, cosine 01 and 02. Variations to that coding scheme could be adding an A as a suffix fro absolute value, and adding AN as a suffix for the negative of the absolute value.

In the case of the curves discussed for the DOCURVE function, the coding scheme would include:

S01-S06, S01A-S06A, S01An-S06AN and
C01-C06, C01A-C06A, C01An-C06AN

The case of NONE or undefined is also included.

To use this function have first input the coded function name:

```
; get curve types
(setq lctype
 (getstring "Enter length curve type:"))
```

Then use the function name to compute the modified distance, for example:

```
; compute endpoint for length
(setq cdist1 (* linelen linefact))
(setq cdist2
 (* linelen (- 1.0 linefact)))
(setq newlen
 (+ cdist1
 (modcurve lctype cdist2 ang)))
```

Completed function MODCURVE:

```
;-----------------------------------
(defun modcurve ( ctype cdist cang /
 newcdist )
 (setq ctype (strcase ctype))
 ; NONE or undefined
 (setq newcdist cdist)
 ; 01 = SIN(x)
 (if (= (substr ctype 1 3) "S01")
  (setq newcdist
  (* (sin (dtr cang)) cdist)))
 ; 02 = nSIN(x)
 (if (= (substr ctype 1 3) "S02")
  (setq newcdist
  (* (* (+ 1.0 (/ (rem cang 360) 360.0))
  (sin (dtr cang))) cdist))  )
 ; 03 = SIN(nx)
 (if (= (substr ctype 1 3) "S03")
  (setq newcdist
  (* (sin (dtr (* (+ 1.0 (/
  (rem cang 360) 360.0)) cang))) cdist)))
 ; 04 = SIN(x)+SIN(2x)
 (if (= (substr ctype 1 3) "S04")
  (setq newcdist
  (* (+ (sin (dtr cang))
  (sin (dtr (* cang 2)))) cdist)))
 ; 05 = SIN(2x)+SIN(3x)
 (if (= (substr ctype 1 3) "S05")
  (setq newcdist
  (* (+ (sin (dtr (* cang 2)))
  (sin (dtr (* cang 3)))) cdist)))
 ; 06 = SIN(x)+SIN(2x)+SIN(3x)
 (if (= (substr ctype 1 3) "S06")
  (setq newcdist
  (* (+ (sin (dtr cang))
  (+ (sin (dtr (* cang 2)))
  (sin (dtr (* cang 3))))) cdist)))
 ; 01 = COS(x)
 (if (= (substr ctype 1 3) "C01")
  (setq newcdist
  (* (cos (dtr cang)) cdist)))
 ; 02 = nCOS(x)
 (if (= (substr ctype 1 3) "C02")
  (setq newcdist
  (* (* (+ 1.0 (/ (rem cang 360) 360.0))
  (cos (dtr cang))) cdist)))
 ; 03 = COS(nx)
 (if (= (substr ctype 1 3) "C03")
  (setq newcdist
  (* (cos (dtr (* (+ 1.0 (/
  (rem cang 360) 360.0)) cang))) cdist)))
 ; 04 = COS(x)+COS(2x)
 (if (= (substr ctype 1 3) "C04")
  (setq newcdist
  (* (+ (cos (dtr cang))
  (cos (dtr (* cang 2)))) cdist)))
 ; 05 = COS(2x)+COS(3x)
 (if (= (substr ctype 1 3) "C05")
  (setq newcdist
  (* (+ (cos (dtr (* cang 2)))
  (cos (dtr (* cang 3)))) cdist)))
 ; 06 = COS(x)+COS(2x)+COS(3x)
 (if (= (substr ctype 1 3) "C06")
  (setq newcdist
  (* (+ (cos (dtr cang))
  (+ (cos (dtr (* cang 2)))
  (cos (dtr (* cang 3))))) cdist)))
 ;
 ; check to make ABS
 (if (= (substr ctype 4 1) "A")
  (setq newcdist (abs newcdist))
 )
 ; check to make -ABS
 (if (= (substr ctype 4 2) "AN")
  (setq newcdist (* newcdist -1.0))
 )
 ; return value
 (+ newcdist 0)
)
;-----------------------------------
```

Using the same curve naming scheme add your own curve modifiers.

If the series of curves are not fixed and a great variety of them need to be explored, instead of selecting already specified curves, enter the curve expression directly as demonstrated in the previous function DOCURVE.

To use this method first input the expression as a string:

```
; get curve epression
(setq lcexp
 (getstring
  "Enter length curve epression:"))
```

Then evaluate the expression to compute the modified distance, for example:

```
; compute endpoint for length
(setq cdist1 (* linelen linefact))
(setq cdist2
 (* linelen (- 1.0 linefact)))
(setq newlen
 (+ cdist1
 (* cdist2 (eval (read lcexp)))))
```

The expression could be entered as any of the following:

```
(sin (dtr ang))
(* 2.0 (sin (dtr ang)))
(abs (* 2.0 (sin (dtr ang))))
(* (abs (* 2.0 (sin (dtr ang)))) -1)
```

Review the section on the DOCURVE function on how to construct these expressions.