Building a Large Bridge

Robert Heller

July 20, 2000 (Final-DRAFT)

Abstract

This article describes how I built a large H0 scale thru-truss double-track bridge to span over a 4' window. The bridge is a Warren Truss that is 400 scale feet long and it is made entirely out of styreen structure shapes.

Contents

Figure 1: Bridge, side view.

Figure 2: Bridge, floor view.

1 Introduction

I am in the process of working on a large H0 layout. Unfortunately, I don't have room for a conventional table-top level layout. Instead I will be building my layout about a foot or so below my ceiling, with much of the layout going along the walls as a narrow shelf type layout. At one point along the wall, I have a 4' wide window. I did not want to put support woodwork in front of the window, so I need to deal with a 4' clear span – AHA: What a great place for a bridge. So I set about designing and building a long bridge. See Figures 1 and 2.

2 Bridge Design

I designed a 20 panel Warren Truss bridge. To help figure the angles, spacings, and other measurements, I wrote a Tcl/Tk script to help me. This script is listed in Appendix A.

Figure 3: Floor Construction Jig.

3 Floor Construction

3.1 Jig

To help build the floor, I had a local carpenter friend make a jig to help keep the floor stringers aligned and properly positioned while gluing them to the floor beams, see Figure 3.

3.2 Floor Panel Parts

Each floor panel consists of a pair of floor beams (54" I beams (5/8" styreen I beams in H0 scale)), 4 stringers (38" I beams (7/16" styreen I beams in H0 scale), and 4 pieces of bracing angles – I used 3/4" molded styreen Warren Truss's. Each Floor beam as a pair of 3" (1/32" in H0 scale) angles attached to the webs at the ends to form a wider mounting area. The stringers also have 3" angles at the ends, as well as a 3" seat angle. The stringers are also notched to allow for the floor beam flanges, as shown in Figure 4.

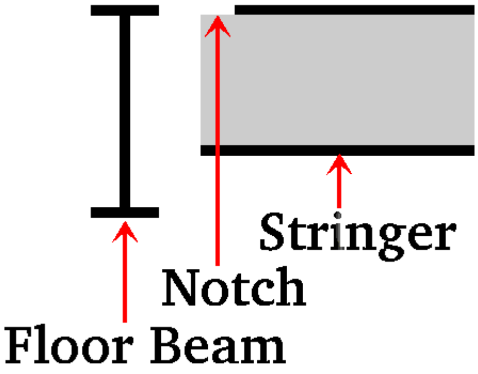

Figure 4: Stringer Notch

Figure 5: Floor, Step 1

3.3 Step-By-Step Floor Panel Construction

1. First the last completed panel is placed in the jig as shown in Figure 5.

2. Then the four stringers are glued on using the channels in the jig. The stringer notches go on the bottom (the bridge floor it built upside down). A small rubber band is used to secure and clamp the pieces until the glue dries as shown in Figures 6 and 7.

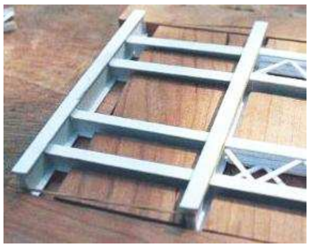

Figure 6: Floor, Step 2, view a

Figure 7: Floor, Step 2, view b

3. While the glue on the stringers dries, the stringer braces are prepared. The braces are formed from Plastruct's 3/4" styreen Warren Truss elements. These are molded as large, open I beams. They need to be converted to something like a large channel by cutting off the flanges on one side as shown below. Then they need a slight length trim. I used a razor saw to cut sections at 1-1/2 'cycle' intervals, cutting in the center of the 'V', but this length is about 3/16" too long to fit between the 3" stringer mounting angles (and between the flanges of the pair of floor beams). I trimmed about 3/32" off each end of the trusses, as shown in Figures 8, 9, and 10.

4. Once the stringer braces have been prepared they can be installed. First the "top"

Figure 8: Floor, Step 3: triming truss flanges

Figure 9: Floor, Step 3: sanding knife cuts

Figure 10: Floor, Step 3: triming ends of trusses

Figure 11: Floor, Step 4: dropping the first top brace in place

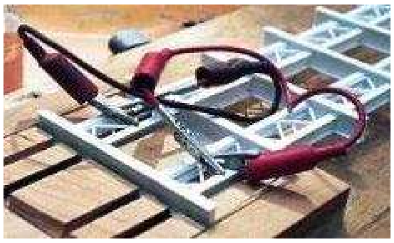

Figure 12: Floor, Step 4: clamping the bottom braces

braces are dropped into place, then the “bottom” braces are popped in. I used aligator clips to clamp the bottom braces (letting gravity hold the top braces in place as the glue dried, as shown in Figures 11 and 12.

3.4 Mostly Completed Floor

The floor with 16 of 20 panels completed is shown in Figure13.

Figure 13: 16 of 20 panels completed

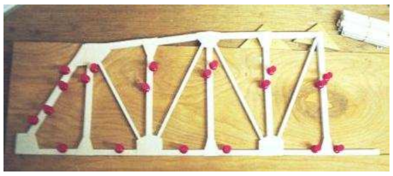

Figure 14: Side Truss, about to add a pair of panels

4 Side Truss Construction

4.1 Jig

The side construction jig is just two long pieces of 1/4” plywood, one wide and one narrow, glued one on top of the other. This forms a long, straight “curb” to build the side section. Also needed are a pair of small right, 45 degree triangles, about 1-1/2” to 2” on a side, with a pair of small holes (to clear push pins) in each. Side construction also requires a good supply of push pins.

4.2 Side Truss Parts and Materials

The sides are made with 21” H columns for the vertical and horizontal runs (1/4” in H0) and 21” I beams (1/4” in H0) for the diagonal braces. Plus a supply of 1” (.01” in H0) sheet for gussets.

4.3 Side Truss Construction

The side trusses are built in a similar step and repeat fashion as the floor. Each major step builds two panels. Starting with a completed panel step as shown in Figure 14.

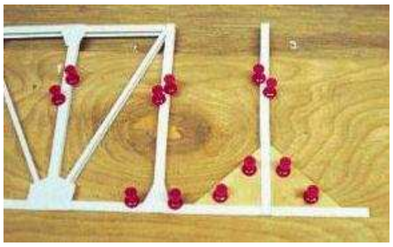

Figure 15: Side Truss, first column added

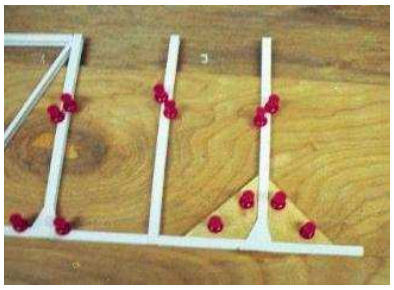

Figure 16: Side Truss, second column added

1. First a column is added. The small triangles are used to make sure the column is plumb. Push pins are used to hold the pieces in place. This is shown in Figure 15.
2. Now the second column is added. A non-brace type gussett is used to secure it, as shown in Figure 16.
3. One diagonal brace and one top beam is added and secured with a top brace type gussett, as shown in Figure 17.

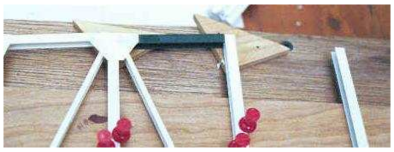

Figure 17: Side Truss, first top beam added

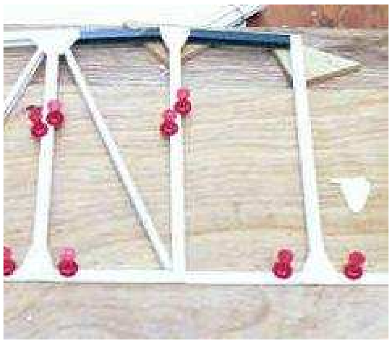

Figure 18: Side Truss, second top beam added

Figure 19: Floor plus side, side view

4. Now the second top beam is added, secured with a top non-brace type top gussett, as shown in Figure 18.

5. Finally, the second diagonal brace is added and secured with a bottom brace type gussett, as shown at the beginning of Subsection 4.3.

5 Assembly of the side trusses and the floor

The side trusses are glued to the side of the floor as shown in Figures 19 through 22.

And the second side truss, shown in Figures 23 through 25.

Figure 20: Floor plus side, end view

Figure 21: Floor plus side, detailed view 1

Figure 22: Floor plus side, detailed view 2
][nl clear=all

Figure 23: Floor plus 2 sides, side view

Figure 24: Floor plus 2 sides, detailed view 1

Figure 25: Floor plus 2 sides, detailed view 2
][nl clear=all

6 Laying the track

The next step is laying the track. I used normal code 83 flextrack, which I stiffened using CA on selected tie-rail points and to which I added guard rails made from code 70 rails. I contact cemented the track sections to the bridge floor as shown in Figures 26 and 27.

7 Top Bracing

Next, the top bracing is installed.

7.1 Portal braces

The portal bracing uses channels and small T braces, as shown in Figure 28.

7.2 Top beams and bracing

The rest of the top bracing and beam assembly is shown below. Note the use of a thread to ensure level placement of the cross channels. See Figures 29 and 30.

8 Completed Bridge

The completed bridge is shown in Figures 31, 32, and 33.

9 Bottom coat: “Rust Layer”

The bridge with its first coat of paint (Rust) is shown in Figures 34, 35, and 36.

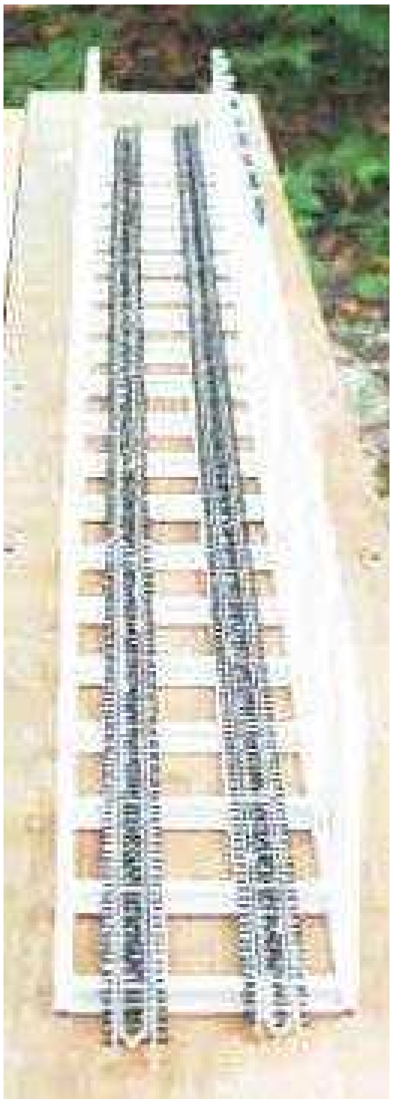

Figure 26: Track layed, end view

Figure 27: Track layed, oblique view

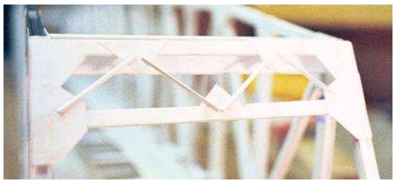

Figure 28: Portal bracing

Figure 29: Top bracing, view 1

Figure 30: Top bracing, view 2

10 Top coat: “ugly” green (Green Zinc Chromate)

The bridge with its second coat of paint (Green Zinc Chromate) is shown in Figures 37 through 40.

Figure 31: Completed bridge.

Figure 32: Completed bridge, top view

Figure 33: Completed bridge,Left End Closeup showing brace detail

Figure 34: Completed first coat (Rust)

Figure 35: Completed first coat (Rust)

Figure 36: Completed first coat (Rust)

Figure 37: Completed second coat (Green)

Figure 38: Completed second coat (Green)

Figure 39: Completed second coat (Green)

Figure 40: Completed second coat (Green)

A Bridge.tcl

```
#!/usr/bin/wish -f
# Program: Bridge
# Tcl version: 7.6 (Tcl/Tk/XF)
# Tk version: 4.2
# XF version: 4.0
#

# module inclusion
global env
global xfLoadPath
global xfLoadInfo
set xfLoadInfo 0
if {[info exists env(XF_LOAD_PATH)]} {
  if {[string first $env(XF_LOAD_PATH) /home/sven/xf] == -1} {
    set xfLoadPath $env(XF_LOAD_PATH):/home/sven/xf
  } {
    set xfLoadPath /home/sven/xf
  }
} {
  set xfLoadPath /home/sven/xf
}

global argc
global argv
set tmpArgv ""
for {set counter 0} {$counter < $argc} {incr counter 1} {
  case [string tolower [lindex $argv $counter]] in {
    {-xfloadpath} {
      incr counter 1
      set xfLoadPath "[lindex $argv $counter]:$xfLoadPath"
    }
    {-xfstartup} {
      incr counter 1
      source [lindex $argv $counter]
    }
    {-xfbindfile} {
      incr counter 1
      set env(XF_BIND_FILE) "[lindex $argv $counter]"
    }
    {-xfcolorfile} {
      incr counter 1
      set env(XF_COLOR_FILE) "[lindex $argv $counter]"
    }
```

```
    {-xfcursorfile} {
      incr counter 1
      set env(XF_CURSOR_FILE) "[lindex $argv $counter]"
    }
    {-xffontfile} {
      incr counter 1
      set env(XF_FONT_FILE) "[lindex $argv $counter]"
    }
    {-xfmodelmono} {
      tk colormodel . monochrome
    }
    {-xfmodelcolor} {
      tk colormodel . color
    }
    {-xfloading} {
      set xfLoadInfo 1
    }
    {-xfnoloading} {
      set xfLoadInfo 0
    }
    {default} {
      lappend tmpArgv [lindex $argv $counter]
    }
  }
}
set argv $tmpArgv
set argc [llength $tmpArgv]
unset counter
unset tmpArgv


# procedure to show window .
proc ShowWindow. {args} {# xf ignore me 7

  # Window manager configurations
  wm positionfrom . user
  wm sizefrom . ""
  wm maxsize . 1137 870
  wm minsize . 1 1
  wm protocol . WM_DELETE_WINDOW {XFProcError {Application win-
dows can not be destroyed.
Please use the "Current widget path:" to show/hide windows.}}
  wm title . {Bridge.tcl}
```

```
# build widget .frame0
frame .frame0 \
  -borderwidth {2} \
  -relief {raised}

# build widget .frame0.menubutton1
menubutton .frame0.menubutton1 \
  -menu {.frame0.menubutton1.m} \
  -padx {5} \
  -pady {4} \
  -text {File} \
  -underline {0}

# build widget .frame0.menubutton1.m
menu .frame0.menubutton1.m \
  -tearoff {0}
.frame0.menubutton1.m add command \
  -label {New} \
  -state {disabled} \
  -underline {0}
.frame0.menubutton1.m add command \
  -label {Open...} \
  -state {disabled} \
  -underline {0}
.frame0.menubutton1.m add separator
.frame0.menubutton1.m add command \
  -label {Save} \
  -state {disabled} \
  -underline {0}
.frame0.menubutton1.m add command \
  -label {Save as...} \
  -state {disabled} \
  -underline {5}
.frame0.menubutton1.m add separator
.frame0.menubutton1.m add command \
  -command {DoPrint} \
  -label {Print...} \
  -underline {0}
.frame0.menubutton1.m add separator
.frame0.menubutton1.m add command \
  -command {CareFullExit} \
  -label {Exit} \
  -underline {1}

# build widget .frame0.menubutton99
```

```
menubutton .frame0.menubutton99 \
  -menu {.frame0.menubutton99.m} \
  -padx {5} \
  -pady {4} \
  -text {Help} \
  -underline {0}

# build widget .frame0.menubutton99.m
menu .frame0.menubutton99.m \
  -tearoff {0}

# build widget .frame0.menubutton0
menubutton .frame0.menubutton0 \
  -menu {.frame0.menubutton0.m} \
  -padx {5} \
  -pady {4} \
  -text {Edit} \
  -underline {0}

# build widget .frame0.menubutton0.m
menu .frame0.menubutton0.m \
  -tearoff {0}

# build widget .frame0.menubutton2
menubutton .frame0.menubutton2 \
  -menu {.frame0.menubutton2.m} \
  -padx {5} \
  -pady {4} \
  -text {View} \
  -underline {0}

# build widget .frame0.menubutton2.m
menu .frame0.menubutton2.m \
  -tearoff {0}
.frame0.menubutton2.m add radiobutton \
  -label {1:1} \
  -state {active} \
  -value {1} \
  -variable {ScalingFactor} \
  -command {ReShow}
.frame0.menubutton2.m add radiobutton \
  -label {1:2} \
  -value {.5} \
  -variable {ScalingFactor} \
  -command {ReShow}
```

```
.frame0.menubutton2.m add radiobutton \
  -label {1:4} \
  -value {.25} \
  -variable {ScalingFactor} \
  -command {ReShow}
.frame0.menubutton2.m add radiobutton \
  -label {1:8} \
  -value {.125} \
  -variable {ScalingFactor} \
  -command {ReShow}
.frame0.menubutton2.m add radiobutton \
  -label {1:16} \
  -value {.0625} \
  -variable {ScalingFactor} \
  -command {ReShow}
.frame0.menubutton2.m add radiobutton \
  -label {1:32} \
  -value {.03125} \
  -variable {ScalingFactor} \
  -command {ReShow}

# build widget .frame0.menubutton3
menubutton .frame0.menubutton3 \
  -menu {.frame0.menubutton3.m} \
  -padx {5} \
  -pady {4} \
  -text {Show} \
  -underline {0}

# build widget .frame0.menubutton3.m
menu .frame0.menubutton3.m \
  -tearoff {0}
.frame0.menubutton3.m add radiobutton \
  -command {DoShow FloorU} \
  -label {Under Floor} \
  -value {FloorU} \
  -variable {NowShowing}
.frame0.menubutton3.m add radiobutton \
  -command {DoShow Floor} \
  -label {Floor} \
  -value {Floor} \
  -variable {NowShowing}
.frame0.menubutton3.m add radiobutton \
  -command {DoShow Side} \
  -label {Side} \
```

```
  -value {Side} \
  -variable {NowShowing}
.frame0.menubutton3.m add radiobutton \
  -command {DoShow Top} \
  -label {Top} \
  -value {Top} \
  -variable {NowShowing}
.frame0.menubutton3.m add radiobutton \
  -command {DoShow Details} \
  -label {Details} \
  -value {Details} \
  -variable {NowShowing}

# build widget .frame2
frame .frame2 \
  -relief {raised}

# build widget .frame2.scrollbar3
scrollbar .frame2.scrollbar3 \
  -command {.frame2.canvas2 xview} \
  -orient {horizontal} \
  -relief {raised}

# build widget .frame2.scrollbar1
scrollbar .frame2.scrollbar1 \
  -command {.frame2.canvas2 yview} \
  -relief {raised}

# build widget .frame2.canvas2
canvas .frame2.canvas2 \
  -height {480} \
  -relief {raised} \
  -scrollregion {0c 0c 20c 20c} \
  -width {640} \
  -xscrollcommand {.frame2.scrollbar3 set} \
  -yscrollcommand {.frame2.scrollbar1 set}

# pack master .frame0
pack configure .frame0.menubutton1 \
  -side left
pack configure .frame0.menubutton0 \
  -side left
pack configure .frame0.menubutton2 \
  -side left
pack configure .frame0.menubutton3 \
```

```
    -side left
  pack configure .frame0.menubutton99 \
    -side right

  # pack master .frame2
  pack configure .frame2.scrollbar1 \
    -fill y \
    -side right
  pack configure .frame2.canvas2 \
    -expand 1 \
    -fill both
  pack configure .frame2.scrollbar3 \
    -fill x

  # pack master .
  pack configure .frame0 \
    -fill x
  pack configure .frame2 \
    -fill both -expand 1

  # build canvas items .frame2.canvas2



  if {"[info procs XFEdit]" != ""} {
    catch "XFMiscBindWidgetTree ."
    after 2 "catch {XFEditSetShowWindows}"
  }
}


# User defined procedures


# Procedure: CareFullExit
proc CareFullExit {} {
  if {[YesNoBox "Really Exit?"]} {
    exit
  }
}

# Procedure: BrowsePrintFileName
proc BrowsePrintFileName {} {
  global PrintFileName
  global fsBox
```

```
  set fsBox(pattern) {*.ps}
  set fileName [FSBox "Print File Name:" "$PrintFileName"]
  if {"$fileName" == {}} {return}
  set PrintFileName "$fileName"
  update
}

# Procedure: ComputeNumberOfPages
proc ComputeNumberOfPages {} {
  global NowShowing
  set What $NowShowing
  set theCanvas [SN BridgeDisplay]
  set sr [$theCanvas cget -scrollregion]
  global ScalingFactor
  global BridgeWidthInches
  global BridgeLengthInches
  global ViewWidths
  global ViewHeights
  global H0Scale
  set VW [expr $ViewWidths($What) * $H0Scale * $ScalingFactor]i
  set VH [expr $ViewHeights($What) * $H0Scale * $ScalingFactor]i
  set vrx  [$theCanvas create rectangle 0 0 $VW $VH]
  set vrbb [$theCanvas bbox $vrx]
  $theCanvas delete $vrx
#  puts "*** vrbb = $vrbb"
  set inx [$theCanvas create rectangle 0 0 1i 1i]
  set inbb [$theCanvas bbox $inx]
  $theCanvas delete $inx
  set XInches [lindex $inbb 2]
  set YInches [lindex $inbb 3]
  set XSizeInches [expr (ceil(double([lindex $vrbb 2]) / $XInches)) + 2.0]
  set YSizeInches [expr (ceil(double([lindex $vrbb 3]) / $YInches)) + 2.0]
#  puts "*** XSizeInches = $XSizeInches, YSizeInches = $YSizeInches"
  set sr [$theCanvas cget -scrollregion]
  set srx  [$theCanvas create rectangle 0 0 [lindex $sr 2] [lindex $sr 3]]
  set srbb [$theCanvas bbox $srx]
  $theCanvas delete $srx
#  puts "*** srbb = $srbb"
  set SXSizeInches [expr (ceil(double([lindex $srbb 2]) / $XInches))]
  set SYSizeInches [expr (ceil(double([lindex $srbb 3]) / $YInches))]
  if {$XSizeInches > $SXSizeInches} {set XSizeInches $SXSizeInches}
  if {$YSizeInches > $SYSizeInches} {set YSizeInches $SYSizeInches}
  set numberOfPages [expr ceil($XSizeInches / 9.0) * ceil($YSizeInches / 6.0)]
  return [expr int(ceil($numberOfPages))]
}
```

```
# Procedure: ComputeScrollRegion
proc ComputeScrollRegion {} {
  global NowShowing
  global ViewWidths
  global ViewHeights
  global ScalingFactor
  global H0Scale
  set theCanvas [SN BridgeDisplay]

  set displayWidth [expr (96.0 + $ViewWidths($NowShowing)) * $H0Scale * \
 $ScalingFactor]i
  set displayHeight [expr (96.0 + $ViewHeights($NowShowing)) * $H0Scale * \
  $ScalingFactor]i
  set CW [$theCanvas cget -width]
  set CH [$theCanvas cget -height]
  set point [$theCanvas create rectangle $displayWidth $displayHeight \
 $displayWidth $displayHeight]
  set bbox [$theCanvas bbox $point]
  $theCanvas delete $point
  if {[lindex $bbox 0] < $CW} {set displayWidth $CW}
  if {[lindex $bbox 1] < $CH} {set displayHeight $CH}
  return [list 0i 0i $displayWidth $displayHeight]
}

# Procedure: DoBarLengths
proc DoBarLengths {} {
  global GirderHeights
  global BarLengths
  set BarLengths {}
  set index 0
  foreach h $GirderHeights {
    if {$index == 0 || $index == 18} {
      lappend BarLengths [expr sqrt(($h * $h) + 400.0)]
    } elseif {[expr $index % 2] == 0} {
      set bl [expr sqrt(($h * $h) + 400.0)]
      lappend BarLengths $bl $bl
    }
    incr index
  }
}

# Procedure: DoPrint
proc DoPrint {} {
# .printForm
```

```
# The above line makes pasting MUCH easier for me.
# It contains the pathname of the cutted widget.
# Tcl version: 7.6 (Tcl/Tk/XF)
# Tk version: 4.2
# XF version: 4.0
#

  # build widget .printForm
  if {"[info procs XFEdit]" != ""} {
    catch "XFDestroy .printForm"
  } {
    catch "destroy .printForm"
  }
  toplevel .printForm  \
    -relief {raised}

  # Window manager configurations
  wm positionfrom .printForm ""
  wm sizefrom .printForm ""
  wm maxsize .printForm 1280 1024
  wm minsize .printForm 1 1
  wm title .printForm {Print view...}
  wm transient .printForm .
  wm geometry .printForm "+512+384"


  # build widget .printForm.frame3
  frame .printForm.frame3 \
    -height {30} \
    -width {30}

  # build widget .printForm.frame3.label9
  label .printForm.frame3.label9 \
    -text {Print To:}

  # build widget .printForm.frame3.frame10
  frame .printForm.frame3.frame10 \
    -height {30} \
    -width {30}

  # build widget .printForm.frame3.frame10.radiobutton12
  radiobutton .printForm.frame3.frame10.radiobutton12 \
    -text {Printer: } \
    -value {Printer} \
    -variable {PrintTo}
```

```
# build widget .printForm.frame3.frame10.entry13
entry .printForm.frame3.frame10.entry13 \
  -textvariable {PrinterCommand}

# build widget .printForm.frame3.frame11
frame .printForm.frame3.frame11 \
  -height {30} \
  -width {30}

# build widget .printForm.frame3.frame11.radiobutton14
radiobutton .printForm.frame3.frame11.radiobutton14 \
  -text {File:} \
  -value {File} \
  -variable {PrintTo}

# build widget .printForm.frame3.frame11.entry15
entry .printForm.frame3.frame11.entry15 \
  -textvariable {PrintFileName}

# build widget .printForm.frame3.frame11.button16
button .printForm.frame3.frame11.button16 \
  -command {BrowsePrintFileName} \
  -padx {11} \
  -pady {4} \
  -text {Browse}

# build widget .printForm.frame19
frame .printForm.frame19


# build widget .printForm.frame19.label0
label .printForm.frame19.label0 \
  -text {Number of Pages:}

# build widget .printForm.frame19.label1
label .printForm.frame19.label1 \
  -relief {sunken} \
  -text [ComputeNumberOfPages]

# build widget .printForm.frame5
frame .printForm.frame5 \
  -height {30} \
  -width {30}
```

```
# build widget .printForm.frame5.button2
button .printForm.frame5.button2 \
  -command {PrintIt} \
  -padx {11} \
  -pady {4} \
  -text {Print}

# build widget .printForm.frame5.button3
button .printForm.frame5.button3 \
  -command {if {"[info procs XFEdit]" != ""} {
  catch "XFDestroy .printForm"
} {
  catch "destroy .printForm"
}} \
  -padx {11} \
  -pady {4} \
  -text {Cancel}

# pack master .printForm.frame3
pack configure .printForm.frame3.label9 \
  -expand 1 \
  -fill x
pack configure .printForm.frame3.frame10 \
  -expand 1 \
  -fill x
pack configure .printForm.frame3.frame11 \
  -expand 1 \
  -fill x

# pack master .printForm.frame3.frame10
pack configure .printForm.frame3.frame10.radiobutton12 \
  -side left
pack configure .printForm.frame3.frame10.entry13 \
  -expand 1 \
  -fill x \
  -side left

# pack master .printForm.frame3.frame11
pack configure .printForm.frame3.frame11.radiobutton14 \
  -side left
pack configure .printForm.frame3.frame11.entry15 \
  -expand 1 \
  -fill x \
  -side left
pack configure .printForm.frame3.frame11.button16 \
```

```
    -side left

  # pack master .printForm.frame19
  pack configure .printForm.frame19.label0 \
    -side left
  pack configure .printForm.frame19.label1 \
    -expand 1 \
    -fill x \
    -side left

  # pack master .printForm.frame5
  pack configure .printForm.frame5.button2 \
    -expand 1 \
    -side left
  pack configure .printForm.frame5.button3 \
    -expand 1 \
    -side left

  # pack master .printForm
  pack configure .printForm.frame3 \
    -expand 1 \
    -fill both
  pack configure .printForm.frame19 \
    -expand 1 \
    -fill both
  pack configure .printForm.frame5 \
    -expand 1 \
    -fill both
# end of widget tree

  update idletask
  grab .printForm
  tkwait window .printForm
}

# Procedure: DoShow
proc DoShow { What} {
  set theCanvas [SN BridgeDisplay]
  $theCanvas delete all
  $theCanvas config -scrollregion [ComputeScrollRegion]
  set sr [$theCanvas cget -scrollregion]
  global ScalingFactor
  global BridgeWidthInches
  global BridgeLengthInches
  global ViewWidths
```

```
  global ViewHeights
  global H0Scale
  set VW [expr $ViewWidths($What) * $H0Scale * $ScalingFactor]i
  set VH [expr $ViewHeights($What) * $H0Scale * $ScalingFactor]i
  set srx  [$theCanvas create rectangle 0 0 [lindex $sr 2] [lindex $sr 3]]
  set srbb [$theCanvas bbox $srx]
  $theCanvas delete $srx
#  puts "*** srbb = $srbb"
  set vrx  [$theCanvas create rectangle 0 0 $VW $VH]
  set vrbb [$theCanvas bbox $vrx]
  $theCanvas delete $vrx
#  puts "*** vrbb = $vrbb"
  set XDiff [expr [lindex $srbb 2] - [lindex $vrbb 2]]
  set YDiff [expr [lindex $srbb 3] - [lindex $vrbb 3]]
#  puts "*** XDiff = $XDiff, YDiff = $YDiff"
  set XOffsetPix [expr $XDiff / 2.0]
  set YOffsetPix [expr $YDiff / 2.0]
  set inx [$theCanvas create rectangle 0 0 1i 1i]
  set inbb [$theCanvas bbox $inx]
  $theCanvas delete $inx
  set XInches [lindex $inbb 2]
  set YInches [lindex $inbb 3]
  set TwoYPixInInches [expr 2.0 / $YInches]
  set XOffset [expr $XOffsetPix / $XInches]
  set YOffset [expr $YOffsetPix / $YInches]
#  puts "*** XOffset = $XOffset, YOffset = $YOffset"


  switch -exact $What {
    FloorU  {
set i42 [expr 42.0 * $H0Scale * $ScalingFactor]
set i21 [expr 21.0 * $H0Scale * $ScalingFactor]
DoShow Floor
set flbWidth [expr 22.0 * $H0Scale * $ScalingFactor]
set flbLength [expr ($BridgeWidthInches - 42) * $H0Scale * \
    $ScalingFactor]
set floorbraceWidth [expr 6.0 * $H0Scale * $ScalingFactor]
for {set i 0} {$i < 20} {incr i} {
  set flbCenterLine [expr 12.0 * $i * 20 * $H0Scale * \
  $ScalingFactor]
  set NextflbCenterLine [expr 12.0 * ($i + 1) * 20 * \
  $H0Scale * $ScalingFactor]
  $theCanvas create line \
[expr $XOffset + $flbCenterLine + $flbWidth]i \
[expr $YOffset + $i21]i \
```

```
[expr $XOffset + $NextflbCenterLine - $flbWidth]i \
[expr $YOffset + $i21 + $flbLength]i \
-width [expr $floorbraceWidth]i -fill darkgreen
  $theCanvas create line \
[expr $XOffset + $flbCenterLine + $flbWidth]i \
[expr $YOffset + $i21 + $flbLength]i \
[expr $XOffset + $NextflbCenterLine - $flbWidth]i \
[expr $YOffset + $i21]i \
-width [expr $floorbraceWidth]i -fill darkgreen
  $theCanvas create rectangle \
[expr $XOffset + $flbCenterLine + $flbWidth]i \
[expr $YOffset]i \
[expr $XOffset + $NextflbCenterLine - $flbWidth]i \
[expr $YOffset + $i21]i \
-fill darkgreen -outline darkgreen
  $theCanvas create rectangle \
[expr $XOffset + $flbCenterLine + $flbWidth]i \
[expr $YOffset + $flbLength + $i21]i \
[expr $XOffset + $NextflbCenterLine - $flbWidth]i \
[expr $YOffset + $flbLength + $i42]i \
-fill darkgreen -outline darkgreen
  UnderFloorBraceGussets $theCanvas $flbLength $flbCenterLine $NextflbCenter-
Line $i $XOffset $YOffset
}
set flbCenterLine [expr 12.0 * 20 * 20 * $HOScale * $ScalingFactor]
LastUnderFloorBraceGussets $theCanvas $flbLength $flbCenterLine $XOff-
set $YOffset
    }
    Floor   {
set i42 [expr 42.0 * $HOScale * $ScalingFactor]
set i21 [expr 21.0 * $HOScale * $ScalingFactor]
set flbWidth [expr 22.0 * $HOScale * $ScalingFactor]
set flbLength [expr ($BridgeWidthInches - 42) * $HOScale * $ScalingFactor]
set t1center [expr $flbLength * 0.25]
set t2center [expr $flbLength * 0.75]
set stringSpacing [expr 6.5 * 12.0 * $HOScale * $ScalingFactor]
set stringSpacing2 [expr $stringSpacing / 2.0]
set stringer1 [expr $t1center - $stringSpacing2]
set stringer2 [expr $t1center + $stringSpacing2]
set stringer3 [expr $t2center - $stringSpacing2]
set stringer4 [expr $t2center + $stringSpacing2]
set stringerWidth [expr 14.0 * $HOScale * $ScalingFactor]
set stringerBraceWidth [expr 3.0 * $HOScale * $ScalingFactor]
set sbracelen [expr (12.0 * 20.0 * $HOScale * $ScalingFactor) - $flbWidth]
set sbracewid [expr $stringSpacing - $stringerWidth]
```

```
set sbracedisp [expr $sbracelen / 3.0]
for {set i 0} {$i <= 20} {incr i} {
  set flbCenterLine [expr 12.0 * $i * 20 * $H0Scale * $ScalingFactor]
  set item [$theCanvas create rectangle \
[expr $XOffset + $flbCenterLine - ($flbWidth / 2.0)]i \
[expr $YOffset + $i21]i \
[expr $XOffset + $flbCenterLine + ($flbWidth / 2.0)]i \
[expr $YOffset + $i21 + $flbLength]i \
-fill darkgreen -outline darkgreen]
#   puts "*** Floor beam at $i: flbCenterLine = $flbCenter-
Line, bbox = [$theCanvas bbox $item]"
  if {$i == 20} {continue}
  set NextflbCenterLine [expr 12.0 * ($i + 1) * 20 * $H0Scale * \
$ScalingFactor]
  set item1 [$theCanvas create rectangle \
[expr $XOffset + $flbCenterLine + ($flbWidth / 2.0)]i \
[expr $YOffset + $i21 + $stringer1 - ($stringerWidth / 2.0)]i \
[expr $XOffset + $NextflbCenterLine - ($flbWidth / 2.0)]i \
[expr $YOffset + $i21 + $stringer1 + ($stringerWidth / 2.0)]i \
-fill darkgreen -outline darkgreen]
  set item2 [$theCanvas create rectangle \
[expr $XOffset + $flbCenterLine + ($flbWidth / 2.0)]i \
[expr $YOffset + $i21 + $stringer2 - ($stringerWidth / 2.0)]i \
[expr $XOffset + $NextflbCenterLine - ($flbWidth / 2.0)]i \
[expr $YOffset + $i21 + $stringer2 + ($stringerWidth / 2.0)]i \
-fill darkgreen -outline darkgreen]
  set sbstartX [expr $XOffset + $flbCenterLine + ($flbWidth / 2.0)]
  set sbstartY1 [expr $YOffset + $i21 + $stringer1 + ($stringerWidth / 2.0)]
  set sbstartY2 [expr $YOffset + $i21 + $stringer2 - ($stringerWidth / 2.0)]
  for {set b 0} {$b < 3} {incr b} {
    $theCanvas create line \
[expr $sbstartX + ($sbracedisp * $b)]i [expr $sbstartY1]i \
[expr $sbstartX + ($sbracedisp * ($b + 1))]i [expr $sbstartY2]i \
-width [expr $stringerBraceWidth]i -fill darkgreen
    $theCanvas create line \
[expr $sbstartX + ($sbracedisp * $b)]i [expr $sbstartY2]i \
[expr $sbstartX + ($sbracedisp * ($b + 1))]i [expr $sbstartY1]i \
-width [expr $stringerBraceWidth]i -fill darkgreen
    StringerGusset $theCanvas $b  1 [expr $sbstartX + ($sbracedisp * $b)] \
    $sbstartY1
    StringerGusset $theCanvas $b -1 [expr $sbstartX + ($sbracedisp * $b)] \
    $sbstartY2
  }
  StringerGusset $theCanvas 3  1 [expr $sbstartX + ($sbracedisp * 3)] \
 $sbstartY1
```

```
  StringerGusset $theCanvas 3 -1 [expr $sbstartX + ($sbracedisp * 3)] \
 $sbstartY2
  set item3 [$theCanvas create rectangle \
[expr $XOffset + $flbCenterLine + ($flbWidth / 2.0)]i \
[expr $YOffset + $i21 + $stringer3 - ($stringerWidth / 2.0)]i \
[expr $XOffset + $NextflbCenterLine - ($flbWidth / 2.0)]i \
[expr $YOffset + $i21 + $stringer3 + ($stringerWidth / 2.0)]i \
-fill darkgreen -outline darkgreen]
  set item4 [$theCanvas create rectangle \
[expr $XOffset + $flbCenterLine + ($flbWidth / 2.0)]i \
[expr $YOffset + $i21 + $stringer4 - ($stringerWidth / 2.0)]i \
[expr $XOffset + $NextflbCenterLine - ($flbWidth / 2.0)]i \
[expr $YOffset + $i21 + $stringer4 + ($stringerWidth / 2.0)]i \
-fill darkgreen -outline darkgreen]
  set sbstartY1 [expr $YOffset + $i21 + $stringer3 + ($stringerWidth / 2.0)]
  set sbstartY2 [expr $YOffset + $i21 + $stringer4 - ($stringerWidth / 2.0)]
  for {set b 0} {$b < 3} {incr b} {
    $theCanvas create line \
[expr $sbstartX + ($sbracedisp * $b)]i [expr $sbstartY1]i \
[expr $sbstartX + ($sbracedisp * ($b + 1))]i [expr $sbstartY2]i \
-width [expr $stringerBraceWidth]i -fill darkgreen
    $theCanvas create line \
[expr $sbstartX + ($sbracedisp * $b)]i [expr $sbstartY2]i \
[expr $sbstartX + ($sbracedisp * ($b + 1))]i [expr $sbstartY1]i \
-width [expr $stringerBraceWidth]i -fill darkgreen
    StringerGusset $theCanvas $b  1 [expr $sbstartX + ($sbracedisp * $b)] \
    $sbstartY1
    StringerGusset $theCanvas $b -1 [expr $sbstartX + ($sbracedisp * $b)] \
    $sbstartY2
  }
  StringerGusset $theCanvas 3  1 [expr $sbstartX + ($sbracedisp * 3)] \
 $sbstartY1
  StringerGusset $theCanvas 3 -1 [expr $sbstartX + ($sbracedisp * 3)] \
 $sbstartY2
}
$theCanvas raise StringerGusset
    }
    Side    {
global GirderHeights
# puts "*** {Side}: GirderHeights = $GirderHeights"
set startX [expr $H0Scale * $ScalingFactor + $XOffset]
set prevX $startX
set deltaX [expr 12.0 * 20 * $H0Scale * $ScalingFactor]
set bottomY [expr $YOffset + (12.0 * 50.0 * $H0Scale * $ScalingFactor)]
set prevY $bottomY
```

```
set GWidth [expr 21.0 * $H0Scale * $ScalingFactor]
set IWidth [expr (21.0 / 2.88) * $H0Scale * $ScalingFactor]
set HWidth [expr (21.0 / 1.5) * $H0Scale * $ScalingFactor]
set HWidth_2 [expr $HWidth * .5]

set down 1
foreach h $GirderHeights {
  set scaleHeight [expr 12.0 * $h * $H0Scale * $ScalingFactor]
#   puts "*** scaleHeight = $scaleHeight, prevX = $prevX, prevY = $prevY, \
deltaX = $deltaX, bottomY = $bottomY"
  $theCanvas create line [expr $prevX]i [expr $prevY]i \
 [expr $prevX + $deltaX]i \
 [expr $bottomY - $scaleHeight]i \
 -width [expr $HWidth]i -fill darkgreen
  $theCanvas create line [expr $prevX + $deltaX]i \
 [expr $bottomY]i \
    [expr $prevX + $deltaX]i \
 [expr $bottomY - $scaleHeight]i \
 -width [expr $HWidth]i -fill darkgreen
  if {$down == 1} {
    $theCanvas create line [expr $prevX + $deltaX]i \
        [expr $bottomY - $scaleHeight]i \
   [expr $prevX + $deltaX + $deltaX]i \
   [expr $bottomY]i \
   -width [expr $IWidth]i -fill darkgreen
    set down -1
    PostOnlyBottomSideGusset $theCanvas [expr $bottomY + $HWidth - \
      $TwoYPixInInches] \
        [expr $prevX + $deltaX]
  } elseif {$down == -1} {
    set down 0
  } else {
    $theCanvas create line [expr $prevX]i \
   [expr $bottomY]i \
   [expr $prevX + $deltaX]i \
   [expr $bottomY - $scaleHeight]i \
   -width [expr $IWidth]i -fill darkgreen
    $theCanvas create line [expr $prevX + $deltaX]i \
        [expr $bottomY - $scaleHeight]i \
   [expr $prevX + $deltaX + $deltaX]i \
   [expr $bottomY]i \
   -width [expr $IWidth]i -fill darkgreen
    TwoBarBottomSideGusset $theCanvas [expr $bottomY + $HWidth - $TwoYPixIn-
Inches] [expr $prevX]
    PostOnlyBottomSideGusset $theCanvas [expr $bottomY + $HWidth - \
```

```
      $TwoYPixInInches] \
[expr $prevX + $deltaX]
    set down -1
   }
   $theCanvas create line [expr $prevX]i [expr $bottomY + $HWidth_2]i \
     [expr $prevX + $deltaX]i \
  [expr $bottomY + $HWidth_2]i \
  -width [expr $HWidth]i -fill darkgreen
   set prevY [expr $bottomY - $scaleHeight]
   set prevX [expr $prevX + $deltaX]
}
$theCanvas create line [expr $prevX]i [expr $prevY]i \
       [expr $prevX + $deltaX]i \
       [expr $bottomY]i \
       -width [expr $HWidth]i -fill darkgreen
$theCanvas create line [expr $prevX]i [expr $bottomY + $HWidth_2]i \
       [expr $prevX + $deltaX]i \
       [expr $bottomY + $HWidth_2]i \
       -width [expr $HWidth]i -fill darkgreen
set tanA [expr double([lindex $GirderHeights 0] + \
      (((21.0 / 1.5) / 2.0) / 12.0)) / 20.0]
set A [expr atan($tanA)]
set B [expr asin(1.0) - $A]
set tanB [expr tan($B)]
set w 21.0
set w_2 [expr $w * .5]
set w_2_2 [expr $w_2 * $w_2]
set z [expr $tanB * $w_2]
set z_2 [expr $z * $z]
set h1 [expr sqrt($z_2 + $w_2_2)]
set ht [expr $w + $h1]
set dxin [expr $ht / $tanA]
set XXOff [expr $dxin * $H0Scale * $ScalingFactor]
LeftBottomSideGusset $theCanvas [expr $bottomY + $HWidth - $TwoYPixIn-
Inches] [expr $startX - $XXOff]
RightBottomSideGusset $theCanvas [expr $bottomY + $HWidth - $TwoYPixIn-
Inches] [expr $prevX + $deltaX + $XXOff]
$theCanvas raise BottomSideGusset
for {set index 1} {$index < 18} {incr index} {
  set ph [lindex $GirderHeights [expr $index - 1]]
  set ch [lindex $GirderHeights $index]
  set ch_scaled [expr 12.0 * $ch * $H0Scale * $ScalingFactor]
  set nh [lindex $GirderHeights [expr $index + 1]]
  set XCenter [expr $startX + $deltaX + ($deltaX * $index)]
  set YCenter [expr $bottomY - $ch_scaled - $HWidth_2]
```

```
  set pTanA [expr double($ch - $ph) / 20.0]
  set nTanA [expr double($ch - $nh) / 20.0]
  if {[expr $index % 2] == 1} {
    PostOnlyTopSideGusset $theCanvas $XCenter $YCenter $pTanA $nTanA
  } else {
    TwoBarTopSideGusset $theCanvas $XCenter $YCenter $pTanA $nTanA
  }
}
set ph 0
set ch [lindex $GirderHeights 0]
set ch_scaled [expr 12.0 * $ch * $H0Scale * $ScalingFactor]
set nh [lindex $GirderHeights 1]
set XCenter [expr $startX + $deltaX]
set YCenter [expr $bottomY - $ch_scaled - $HWidth_2]
set pTanA [expr double($ch - $ph) / 20.0]
set nTanA [expr double($ch - $nh) / 20.0]
LeftTopSideGusset $theCanvas $XCenter $YCenter $pTanA $nTanA
set XCenter [expr $startX + ($deltaX * 19)]
RightTopSideGusset $theCanvas $XCenter $YCenter $nTanA $pTanA
    }
    Top     {}
    Details {}
  }
}

# Procedure: DoTopGirderLengths
proc DoTopGirderLengths {} {
  global GirderHeights
  global TopGirderLengths
  set TopGirderLengths {}
  set prev 0
  foreach h $GirderHeights {
#
#  |\
# h| \
#  |  \
#  ----
#   20
#
    set dheight [expr $h - $prev]
    set hh [expr sqrt(($dheight * $dheight) + 400.0)]
    lappend TopGirderLengths $hh
    set prev $h
  }
  lappend TopGirderLengths [expr sqrt(($prev * $prev) + 400.0)]
```

```
}

# Procedure: FSBox
proc FSBox { {fsBoxMessage "Select file:"} {fsBoxFileName ""} {fsBoxAc-
tionOk ""} {fsBoxActionCancel ""}} {
# xf ignore me 5
##########
# Procedure: FSBox
# Description: show file selector box
# Arguments: fsBoxMessage - the text to display
#            fsBoxFileName - a file name that should be selected
#            fsBoxActionOk - the action that should be performed on ok
#            fsBoxActionCancel - the action that should be performed on cancel
# Returns: the filename that was selected, or nothing
# Sideeffects: none
##########
#
# global fsBox(activeBackground) - active background color
# global fsBox(activeForeground) - active foreground color
# global fsBox(background) - background color
# global fsBox(font) - text font
# global fsBox(foreground) - foreground color
# global fsBox(extensions) - scan directory for extensions
# global fsBox(scrollActiveForeground) - scrollbar active background color
# global fsBox(scrollBackground) - scrollbar background color
# global fsBox(scrollForeground) - scrollbar foreground color
# global fsBox(scrollSide) - side where scrollbar is located

  global fsBox

  set tmpButtonOpt ""
  set tmpFrameOpt ""
  set tmpMessageOpt ""
  set tmpScaleOpt ""
  set tmpScrollOpt ""
  if {"$fsBox(activeBackground)" != ""} {
    append tmpButtonOpt "-activebackground \"$fsBox(activeBackground)\" "
  }
  if {"$fsBox(activeForeground)" != ""} {
    append tmpButtonOpt "-activeforeground \"$fsBox(activeForeground)\" "
  }
  if {"$fsBox(background)" != ""} {
    append tmpButtonOpt "-background \"$fsBox(background)\" "
    append tmpFrameOpt "-background \"$fsBox(background)\" "
    append tmpMessageOpt "-background \"$fsBox(background)\" "
```

```
}
if {"$fsBox(font)" != ""} {
  append tmpButtonOpt "-font \"$fsBox(font)\" "
  append tmpMessageOpt "-font \"$fsBox(font)\" "
}
if {"$fsBox(foreground)" != ""} {
  append tmpButtonOpt "-foreground \"$fsBox(foreground)\" "
  append tmpMessageOpt "-foreground \"$fsBox(foreground)\" "
}
if {"$fsBox(scrollActiveForeground)" != ""} {
  append tmpScrollOpt \
 "-activeforeground \"$fsBox(scrollActiveForeground)\" "
}
if {"$fsBox(scrollBackground)" != ""} {
  append tmpScrollOpt "-background \"$fsBox(scrollBackground)\" "
}
if {"$fsBox(scrollForeground)" != ""} {
  append tmpScrollOpt "-foreground \"$fsBox(scrollForeground)\" "
}

if {[file exists [file tail $fsBoxFileName]] &&
    [IsAFile [file tail $fsBoxFileName]]} {
  set fsBox(name) [file tail $fsBoxFileName]
} {
  set fsBox(name) ""
}
if {[file exists $fsBoxFileName] && [IsADir $fsBoxFileName]} {
  set fsBox(path) $fsBoxFileName
} {
  if {"[file dirname $fsBoxFileName]" != "."} {
    set fsBox(path) [file dirname $fsBoxFileName]
  }
}
if {$fsBox(showPixmap)} {
  set fsBox(path) [string trimleft $fsBox(path) @]
}
if {"$fsBox(path)" != "" && [file exists $fsBox(path)] &&
    [IsADir $fsBox(path)]} {
  set fsBox(internalPath) $fsBox(path)
} {
  if {"$fsBox(internalPath)" == "" ||
      ![file exists $fsBox(internalPath)]} {
    set fsBox(internalPath) [pwd]
  }
}
```

```
# build widget structure

# start build of toplevel
if {"[info commands XFDestroy]" != ""} {
  catch {XFDestroy .fsBox}
} {
  catch {destroy .fsBox}
}
toplevel .fsBox  -borderwidth 0
catch ".fsBox config $tmpFrameOpt"
wm geometry .fsBox 350x300
wm title .fsBox {File select box}
wm maxsize .fsBox 1000 1000
wm minsize .fsBox 100 100
# end build of toplevel

label .fsBox.message1  -anchor c  -relief raised  -text "$fsBoxMessage"
catch ".fsBox.message1 config $tmpMessageOpt"

frame .fsBox.frame1  -borderwidth 0  -relief raised
catch ".fsBox.frame1 config $tmpFrameOpt"

button .fsBox.frame1.ok  -text "OK"  -command "
    global fsBox
    set fsBox(name) \[.fsBox.file.file get\]
    if {$fsBox(showPixmap)} {
      set fsBox(path) @\[.fsBox.path.path get\]
    } {
      set fsBox(path) \[.fsBox.path.path get\]
    }
    set fsBox(internalPath) \[.fsBox.path.path get\]
    $fsBoxActionOk
    if {\"\[info commands XFDestroy\]\" != \"\"} {
      catch {XFDestroy .fsBox}
    } {
      catch {destroy .fsBox}
    }"
catch ".fsBox.frame1.ok config $tmpButtonOpt"

button .fsBox.frame1.rescan  -text "Rescan"  -command {
    global fsBox
    FSBoxFSShow [.fsBox.path.path get]  [.fsBox.pattern.pattern get] \
$fsBox(all)}
catch ".fsBox.frame1.rescan config $tmpButtonOpt"
```

```
  button .fsBox.frame1.cancel  -text "Cancel"  -command "
      global fsBox
      set fsBox(name) {}
      set fsBox(path) {}
      $fsBoxActionCancel
      if {\"\[info commands XFDestroy\]\" != \"\"} {
        catch {XFDestroy .fsBox}
      } {
        catch {destroy .fsBox}
      }"
  catch ".fsBox.frame1.cancel config $tmpButtonOpt"

  if {$fsBox(showPixmap)} {
    frame .fsBox.frame2  -borderwidth 0  -relief raised
    catch ".fsBox.frame2 config $tmpFrameOpt"

    scrollbar .fsBox.frame2.scrollbar3  -command {.fs-
Box.frame2.canvas2 xview}  -orient {horizontal}  -relief {raised}
    catch ".fsBox.frame2.scrollbar3 config $tmpScrollOpt"

    scrollbar .fsBox.frame2.scrollbar1  -command {.fs-
Box.frame2.canvas2 yview}  -relief {raised}
    catch ".fsBox.frame2.scrollbar1 config $tmpScrollOpt"

    canvas .fsBox.frame2.canvas2  -confine {true}  -relief {raised}  -
scrollregion {0c 0c 20c 20c}  -width {100}  -xscrollcommand {.fs-
Box.frame2.scrollbar3 set}  -yscrollcommand {.fsBox.frame2.scrollbar1 set}
    catch ".fsBox.frame2.canvas2 config $tmpFrameOpt"

    .fsBox.frame2.canvas2 addtag currentBitmap withtag [.fs-
Box.frame2.canvas2 create bitmap 5 5 -anchor nw]
  }

  frame .fsBox.path  -borderwidth 0  -relief raised
  catch ".fsBox.path config $tmpFrameOpt"

  frame .fsBox.path.paths  -borderwidth 2  -relief raised
  catch ".fsBox.path.paths config $tmpFrameOpt"

  menubutton .fsBox.path.paths.paths  -borderwidth 0  -menu ".fs-
Box.path.paths.paths.menu"  -relief flat  -text "Pathname:"
  catch ".fsBox.path.paths.paths config $tmpButtonOpt"

  menu .fsBox.path.paths.paths.menu -tearoff 0
  catch ".fsBox.path.paths.paths.menu config $tmpButtonOpt"
```

```
  .fsBox.path.paths.paths.menu add command  \
       -label "[string trimright $fsBox(internalPath) {/@}]" \
       -command "
       global fsBox
       FSBoxFSShow \[.fsBox.path.path get\]  \[.fsBox.pattern.pattern get\] \
   \$fsBox(all)
       .fsBox.path.path delete 0 end
       .fsBox.path.path insert 0 [string trimright $fsBox(internalPath) {/@}]"

  entry .fsBox.path.path  -relief raised
  catch ".fsBox.path.path config $tmpMessageOpt"

  if {![IsADir $fsBox(internalPath)]} {
    set $fsBox(internalPath) [pwd]
  }
  .fsBox.path.path insert 0 $fsBox(internalPath)

  frame .fsBox.pattern  -borderwidth 0  -relief raised
  catch ".fsBox.pattern config $tmpFrameOpt"

  frame .fsBox.pattern.patterns  -borderwidth 2  -relief raised
  catch ".fsBox.pattern.patterns config $tmpFrameOpt"

  menubutton .fsBox.pattern.patterns.patterns  -borderwidth 0  \
-menu ".fsBox.pattern.patterns.patterns.menu"  \
-relief flat  \
-text "Selection pattern:"
  catch ".fsBox.pattern.patterns.patterns config $tmpButtonOpt"

  menu .fsBox.pattern.patterns.patterns.menu -tearoff 0
  catch ".fsBox.pattern.patterns.patterns.menu config $tmpButtonOpt"

  .fsBox.pattern.patterns.patterns.menu add checkbutton  \
-label "Scan extensions"  -variable fsBox(extensions)  -command {
      global fsBox
      FSBoxFSShow [.fsBox.path.path get]  [.fsBox.pattern.pattern get] \
$fsBox(all)}

  entry .fsBox.pattern.pattern  -relief raised
  catch ".fsBox.pattern.pattern config $tmpMessageOpt"

  .fsBox.pattern.pattern insert 0 $fsBox(pattern)

  frame .fsBox.files  -borderwidth 0  -relief raised
```

```
  catch ".fsBox.files config $tmpFrameOpt"

  scrollbar .fsBox.files.vscroll  -relief raised  \
-command ".fsBox.files.files yview"
  catch ".fsBox.files.vscroll config $tmpScrollOpt"

  scrollbar .fsBox.files.hscroll  -orient horiz  -relief raised  \
-command ".fsBox.files.files xview"
  catch ".fsBox.files.hscroll config $tmpScrollOpt"

  listbox .fsBox.files.files  -exportselection false  -relief raised  \
-xscrollcommand ".fsBox.files.hscroll set"  \
-yscrollcommand ".fsBox.files.vscroll set"
  catch ".fsBox.files.files config $tmpMessageOpt"

  frame .fsBox.file  -borderwidth 0  -relief raised
  catch ".fsBox.file config $tmpFrameOpt"

  label .fsBox.file.labelfile  -relief raised  -text "Filename:"
  catch ".fsBox.file.labelfile config $tmpMessageOpt"

  entry .fsBox.file.file  -relief raised
  catch ".fsBox.file.file config $tmpMessageOpt"

  .fsBox.file.file delete 0 end
  .fsBox.file.file insert 0 $fsBox(name)

  checkbutton .fsBox.pattern.all  -offvalue 0  -onvalue 1  \
-text "Show all files"  -variable fsBox(all)  -command {
      global fsBox
      FSBoxFSShow [.fsBox.path.path get]  [.fsBox.pattern.pattern get] \
  $fsBox(all)}
  catch ".fsBox.pattern.all config $tmpButtonOpt"

  FSBoxFSShow $fsBox(internalPath) $fsBox(pattern) $fsBox(all)

  # bindings
  bindtags .fsBox.files.files {.fsBox.files.files "" "" "" }
  bind .fsBox.files.files <Double-Button-1> "
    FSBoxFSFileSelectDouble %W $fsBox(showPixmap) \{$fsBoxActionOk\} %y"
  bind .fsBox.files.files <ButtonPress-1> "
    FSBoxFSFileSelect %W $fsBox(showPixmap) %y"
  bind .fsBox.files.files <Button1-Motion> "
    FSBoxFSFileSelect %W $fsBox(showPixmap) %y"
  bind .fsBox.files.files <Shift-Button1-Motion> "
```

```
    FSBoxFSFileSelect %W $fsBox(showPixmap) %y"
  bind .fsBox.files.files <Shift-ButtonPress-1> "
    FSBoxFSFileSelect %W $fsBox(showPixmap) %y"

  bind .fsBox.path.path <Tab> {
    FSBoxFSNameComplete path}
  bind .fsBox.path.path <Return> {
    global fsBox
    FSBoxFSShow [.fsBox.path.path get]  [.fsBox.pattern.pattern get] \
$fsBox(all)
    FSBoxFSInsertPath
    .fsBox.file.file icursor end
    focus .fsBox.file.file}
  catch "bind .fsBox.path.path <Up> {}"
  bind .fsBox.path.path <Down> {
    .fsBox.file.file icursor end
    focus .fsBox.file.file}

  bind .fsBox.file.file <Tab> {
    FSBoxFSNameComplete file}
  bind .fsBox.file.file <Return> "
    global fsBox
    set fsBox(name) \[.fsBox.file.file get\]
    if {$fsBox(showPixmap)} {
      set fsBox(path) @\[.fsBox.path.path get\]
    } {
      set fsBox(path) \[.fsBox.path.path get\]
    }
    set fsBox(internalPath) \[.fsBox.path.path get\]
    $fsBoxActionOk
    if {\"\[info commands XFDestroy\]\" != \"\"} {
      catch {XFDestroy .fsBox}
    } {
      catch {destroy .fsBox}
    }"
  bind .fsBox.file.file <Up> {
    .fsBox.path.path icursor end
    focus .fsBox.path.path}
  bind .fsBox.file.file <Down> {
    .fsBox.pattern.pattern icursor end
    focus .fsBox.pattern.pattern}

  bind .fsBox.pattern.pattern <Return> {
    global fsBox
    FSBoxFSShow [.fsBox.path.path get]  [.fsBox.pattern.pattern get] \
```

```
$fsBox(all)}
  bind .fsBox.pattern.pattern <Up> {
    .fsBox.file.file icursor end
    focus .fsBox.file.file}
  catch "bind .fsBox.pattern.pattern <Down> {}"

  # packing
  pack append .fsBox.files  .fsBox.files.vscroll "$fsBox(scrollSide) filly"  \
.fsBox.files.hscroll {bottom fillx}  \
.fsBox.files.files {left fill expand}
  pack append .fsBox.file  .fsBox.file.labelfile {left}  \
.fsBox.file.file {left fill expand}
  pack append .fsBox.frame1  .fsBox.frame1.ok {left fill expand}  \
.fsBox.frame1.rescan {left fill expand}  \
.fsBox.frame1.cancel {left fill expand}
  pack append .fsBox.path.paths  .fsBox.path.paths.paths {left}
  pack append .fsBox.pattern.patterns  .fsBox.pattern.patterns.patterns {left}
  pack append .fsBox.path  .fsBox.path.paths {left}  .fs-
Box.path.path {left fill expand}
  pack append .fsBox.pattern  .fsBox.pattern.patterns {left}  .fs-
Box.pattern.all {right fill}  .fsBox.pattern.pattern {left fill expand}
  if {$fsBox(showPixmap)} {
    pack append .fsBox.frame2  .fsBox.frame2.scrollbar1 {left filly}  .fs-
Box.frame2.canvas2 {top expand fill}  .fsBox.frame2.scrollbar3 {top fillx}

    pack append .fsBox  .fsBox.message1 {top fill}  .fsBox.frame1 {bot-
tom fill}  .fsBox.pattern {bottom fill}  .fsBox.file {bot-
tom fill}  .fsBox.path {bottom fill}  .fsBox.frame2 {right fill}  .fs-
Box.files {left fill expand}
  } {
    pack append .fsBox  .fsBox.message1 {top fill}  .fsBox.frame1 {bot-
tom fill}  .fsBox.pattern {bottom fill}  .fsBox.file {bottom fill}  .fs-
Box.path {bottom fill}  .fsBox.files {left fill expand}
  }

  if {"$fsBoxActionOk" == "" && "$fsBoxActionCancel" == ""} {
    # wait for the box to be destroyed
    update idletask
    grab .fsBox
    tkwait window .fsBox

    if {"[string trim $fsBox(path)]" != "" ||
        "[string trim $fsBox(name)]" != ""} {
      if {"[string trimleft [string trim $fsBox(name)] /]" == ""} {
        return [string trimright [string trim $fsBox(path)] /]
```

```
      } {
        return [string trimright [string trim $fsBox(path)] /]/[string trim-
left [string trim $fsBox(name)] /]
      }
    }
  }
}


# Procedure: FSBoxBindSelectOne
proc FSBoxBindSelectOne { fsBoxW fsBoxY} {
# xf ignore me 6

  set fsBoxNearest [$fsBoxW nearest $fsBoxY]
  if {$fsBoxNearest >= 0} {
    $fsBoxW select clear 0 end
    $fsBoxW select set $fsBoxNearest
  }
}


# Procedure: FSBoxFSFileSelect
proc FSBoxFSFileSelect { fsBoxW fsBoxShowPixmap fsBoxY} {
# xf ignore me 6
  global fsBox

  FSBoxBindSelectOne $fsBoxW $fsBoxY
  set fsBoxNearest [$fsBoxW nearest $fsBoxY]
  if {$fsBoxNearest >= 0} {
    set fsBoxTmpEntry [$fsBoxW get $fsBoxNearest]
    if {"[string index $fsBoxTmpEntry  [expr [string length $fsBoxTmpEntry]-
1]]" == "/" ||
        "[string index $fsBoxTmpEntry  [expr [string length $fsBoxTmpEntry]-
1]]" == "@"} {
      set fsBoxFileName [string range $fsBoxTmpEntry 0  \
[expr [string length $fsBoxTmpEntry]-2]]
      if {![IsADir [string trimright $fsBox(internalPath)/$fsBoxFileName \
   @]] &&
          ![IsASymlink [string trimright $fsBox(internalPath)/$fsBoxFileName \
       @]]} {
        set fsBoxFileName $fsBoxTmpEntry
      }
    } {
      if {"[string index $fsBoxTmpEntry  [expr [string length $fsBoxTmpEntry]-
1]]" == "*"} {
```

```
        set fsBoxFileName [string range $fsBoxTmpEntry 0  \
[expr [string length $fsBoxTmpEntry]-2]]
        if {![file executable $fsBox(internalPath)/$fsBoxFileName]} {
          set fsBoxFileName $fsBoxTmpEntry
        }
      } {
        set fsBoxFileName $fsBoxTmpEntry
      }
    }
    if {![IsADir [string trimright $fsBox(internalPath)/$fsBoxFileName @]]} {
      set fsBox(name) $fsBoxFileName
      .fsBox.file.file delete 0 end
      .fsBox.file.file insert 0 $fsBox(name)
      if {$fsBoxShowPixmap} {
        catch ".fsBox.frame2.canvas2 itemconfigure currentBitmap -
bitmap \"@$fsBox(internalPath)/$fsBox(name)\""
      }
    }
  }
}


# Procedure: FSBoxFSFileSelectDouble
proc FSBoxFSFileSelectDouble { fsBoxW fsBoxShowPixmap fsBoxAction fsBoxY} {
# xf ignore me 6
  global fsBox

  FSBoxBindSelectOne $fsBoxW $fsBoxY
  set fsBoxNearest [$fsBoxW nearest $fsBoxY]
  if {$fsBoxNearest >= 0} {
    set fsBoxTmpEntry [$fsBoxW get $fsBoxNearest]
    if {"$fsBoxTmpEntry" == "../"} {
      set fsBoxTmpEntry [string trimright [string trim $fs-
Box(internalPath)] "@/"]
      if {"$fsBoxTmpEntry" == ""} {
        return
      }
      FSBoxFSShow [file dirname $fsBoxTmpEntry]  [.fs-
Box.pattern.pattern get] \
  $fsBox(all)
      .fsBox.path.path delete 0 end
      .fsBox.path.path insert 0 $fsBox(internalPath)
    } {
      if {"[string index $fsBoxTmpEntry  [expr [string length $fsBoxTmpEntry]-
1]]" == "/" ||
```

```
          "[string index $fsBoxTmpEntry  [expr [string length $fsBoxTmpEntry]-
1]]" == "@"} {
        set fsBoxFileName [string range $fsBoxTmpEntry 0 \
  [expr [string length $fsBoxTmpEntry]-2]]
        if {![IsADir [string trimright $fsBox(internalPath)/$fsBoxFileName \
     @]] &&
            ![IsASymlink [string trimright \
 $fsBox(internalPath)/$fsBoxFileName @]]} {
          set fsBoxFileName $fsBoxTmpEntry
        }
      } {
        if {"[string index $fsBoxTmpEntry  [expr [string length $fsBoxTmpEn-
try]-1]]"\
 == "*"} {
          set fsBoxFileName [string range $fsBoxTmpEntry 0 \
    [expr [string length $fsBoxTmpEntry]-2]]
          if {![file executable $fsBox(internalPath)/$fsBoxFileName]} {
            set fsBoxFileName $fsBoxTmpEntry
          }
        } {
          set fsBoxFileName $fsBoxTmpEntry
        }
      }
      if {[IsADir [string trimright $fsBox(internalPath)/$fsBoxFileName @]]} {
        set fsBox(internalPath) "[string trimright $fs-
Box(internalPath) {/@}]/$fsBoxFileName"
        FSBoxFSShow $fsBox(internalPath)  [.fsBox.pattern.pattern get] \
    $fsBox(all)
        .fsBox.path.path delete 0 end
        .fsBox.path.path insert 0 $fsBox(internalPath)
      } {
        set fsBox(name) $fsBoxFileName
        if {$fsBoxShowPixmap} {
          set fsBox(path) @$fsBox(internalPath)
        } {
          set fsBox(path) $fsBox(internalPath)
        }
        if {"$fsBoxAction" != ""} {
          eval "global fsBox; $fsBoxAction"
        }
        if {"[info commands XFDestroy]" != ""} {
          catch {XFDestroy .fsBox}
        } {
          catch {destroy .fsBox}
        }
```

```
      }
    }
  }
}


# Procedure: FSBoxFSInsertPath
proc FSBoxFSInsertPath {} {
# xf ignore me 6
  global fsBox

  set fsBoxLast [.fsBox.path.paths.paths.menu index last]
  set fsBoxNewEntry [string trimright [.fsBox.path.path get] "/@"]
  for {set fsBoxCounter 0} {$fsBoxCounter <= $fsBoxLast} {incr fsBox-
Counter 1} {
    if {"$fsBoxNewEntry" ==  "[lindex [.fsBox.path.paths.paths.menu entrycon-
figure  $fsBoxCounter -label] 4]"} {
      return
    }
  }
  if {$fsBoxLast < 9} {
    .fsBox.path.paths.paths.menu add command  -label "$fsBoxNewEntry"  -
command "
        global fsBox
        FSBoxFSShow $fsBoxNewEntry  \[.fsBox.pattern.pattern get\] \
   \$fsBox(all)
        .fsBox.path.path delete 0 end
        .fsBox.path.path insert 0 $fsBoxNewEntry"
  } {
    for {set fsBoxCounter 0} {$fsBoxCounter < $fsBoxLast} {incr fsBox-
Counter 1} {
      .fsBox.path.paths.paths.menu entryconfigure  $fsBoxCounter -label  [lin-
dex [.fsBox.path.paths.paths.menu entryconfigure  [expr $fsBoxCounter+1] -
label] 4]
      .fsBox.path.paths.paths.menu entryconfigure $fsBoxCounter  -command "
          global fsBox
          FSBoxFSShow [lindex [.fsBox.path.paths.paths.menu \
entryconfigure  \
        [expr $fsBoxCounter+1] -label] 4] \
     \[.fsBox.pattern.pattern get\] \$fsBox(all)
          .fsBox.path.path delete 0 end
          .fsBox.path.path insert 0 [lindex  [.fsBox.path.paths.paths.menu en-
tryconfigure  [expr $fsBoxCounter+1] -label] 4]"
    }
    .fsBox.path.paths.paths.menu entryconfigure $fsBoxLast  -
```

```
label "$fsBoxNewEntry"
    .fsBox.path.paths.paths.menu entryconfigure $fsBoxCounter  -command "
        global fsBox
        FSBoxFSShow \[.fsBox.path.path get\]  \[.fsBox.pattern.pattern get\] \
    \$fsBox(all)
        .fsBox.path.path delete 0 end
        .fsBox.path.path insert 0 $fsBoxNewEntry"
  }
}


# Procedure: FSBoxFSNameComplete
proc FSBoxFSNameComplete { fsBoxType} {
# xf ignore me 6
  global fsBox

  set fsBoxNewFile ""
  if {"$fsBoxType" == "path"} {
    set fsBoxDirName [file dirname [.fsBox.path.path get]]
    set fsBoxFileName [file tail [.fsBox.path.path get]]
  } {
    set fsBoxDirName [file dirname [.fsBox.path.path get]/]
    set fsBoxFileName [file tail [.fsBox.file.file get]]
  }

  set fsBoxNewFile ""
  if {[IsADir [string trimright $fsBoxDirName @]]} {
    catch "glob -nocomplain $fsBoxDirName/${fsBoxFileName}*" fsBoxResult
    foreach fsBoxCounter $fsBoxResult {
      if {"$fsBoxNewFile" == ""} {
        set fsBoxNewFile [file tail $fsBoxCounter]
      } {
        if {"[string index [file tail $fsBoxCounter] 0]" !=
            "[string index $fsBoxNewFile 0]"} {
          set fsBoxNewFile ""
          break
        }
        set fsBoxCounter1 0
        set fsBoxTmpFile1 $fsBoxNewFile
        set fsBoxTmpFile2 [file tail $fsBoxCounter]
        set fsBoxLength1 [string length $fsBoxTmpFile1]
        set fsBoxLength2 [string length $fsBoxTmpFile2]
        set fsBoxNewFile ""
        if {$fsBoxLength1 > $fsBoxLength2} {
          set fsBoxLength1 $fsBoxLength2
```

```
        }
        while {$fsBoxCounter1 < $fsBoxLength1} {
          if {"[string index $fsBoxTmpFile1 $fsBoxCounter1]" ==  "[string in-
dex $fsBoxTmpFile2 $fsBoxCounter1]"} {
            append fsBoxNewFile [string index $fsBoxTmpFile1 $fsBoxCounter1]
          } {
            break
          }
          incr fsBoxCounter1 1
        }
      }
    }
  }
  if {"$fsBoxNewFile" != ""} {
    if {[IsADir [string trimright $fsBoxDirName/$fsBoxNewFile @]] ||
        ![IsAFile [string trimright $fsBoxDirName/$fsBoxNewFile @]]} {
      if {[IsADir [string trimright $fsBoxDirName/$fsBoxNewFile @]]} {
        if {"$fsBoxDirName" == "/"} {
          .fsBox.path.path delete 0 end
          .fsBox.path.path insert 0 "/[string trimright [string trim $fs-
BoxNewFile /] @]/"
        } {
          .fsBox.path.path delete 0 end
          .fsBox.path.path insert 0 "[string trimright $fs-
BoxDirName /]/[string trimright [string trim $fsBoxNewFile /] @]/"
        }
        FSBoxFSShow [.fsBox.path.path get]  [.fsBox.pattern.pattern get] \
    $fsBox(all)
        FSBoxFSInsertPath
      } {
        .fsBox.path.path delete 0 end
        .fsBox.path.path insert 0 "[string trimright $fs-
BoxDirName /]/[string trimright [string trim $fsBoxNewFile /] @]"
      }
    } {
      .fsBox.path.path delete 0 end
      .fsBox.path.path insert 0 "[string trimright $fsBoxDirName {@/}]/"
      .fsBox.file.file delete 0 end
      .fsBox.file.file insert 0 $fsBoxNewFile
      .fsBox.file.file icursor end
      focus .fsBox.file.file
    }
  }
}
```

```
# Procedure: FSBoxFSShow
proc FSBoxFSShow { fsBoxPath fsBoxPattern fsBoxAll} {
# xf ignore me 6
  global fsBox

  set tmpButtonOpt ""
  if {"$fsBox(activeBackground)" != ""} {
    append tmpButtonOpt "-activebackground \"$fsBox(activeBackground)\" "
  }
  if {"$fsBox(activeForeground)" != ""} {
    append tmpButtonOpt "-activeforeground \"$fsBox(activeForeground)\" "
  }
  if {"$fsBox(background)" != ""} {
    append tmpButtonOpt "-background \"$fsBox(background)\" "
  }
  if {"$fsBox(font)" != ""} {
    append tmpButtonOpt "-font \"$fsBox(font)\" "
  }
  if {"$fsBox(foreground)" != ""} {
    append tmpButtonOpt "-foreground \"$fsBox(foreground)\" "
  }

  set fsBox(pattern) $fsBoxPattern
  if {[file exists $fsBoxPath] && [file readable $fsBoxPath] &&
      [IsADir $fsBoxPath]} {
    set fsBox(internalPath) $fsBoxPath
  } {
    if {[file exists $fsBoxPath] && [file readable $fsBoxPath] &&
        [IsAFile $fsBoxPath]} {
      set fsBox(internalPath) [file dirname $fsBoxPath]
      .fsBox.file.file delete 0 end
      .fsBox.file.file insert 0 [file tail $fsBoxPath]
      set fsBoxPath $fsBox(internalPath)
    } {
      while {"$fsBoxPath" != "" && "$fsBoxPath" != "/" &&
             ![file isdirectory $fsBoxPath]} {
        set fsBox(internalPath) [file dirname $fsBoxPath]
         set fsBoxPath $fsBox(internalPath)
      }
    }
  }
  if {"$fsBoxPath" == ""} {
    set fsBoxPath "/"
    set fsBox(internalPath) "/"
```

```
}
.fsBox.path.path delete 0 end
.fsBox.path.path insert 0 $fsBox(internalPath)

if {[.fsBox.files.files size] > 0} {
  .fsBox.files.files delete 0 end
}
if {$fsBoxAll} {
  if {[catch "Ls -F -a $fsBoxPath" fsBoxResult]} {
    puts stderr "$fsBoxResult"
  }
} {
  if {[catch "Ls -F $fsBoxPath" fsBoxResult]} {
    puts stderr "$fsBoxResult"
  }
}
set fsBoxElementList [lsort $fsBoxResult]

foreach fsBoxCounter [winfo children .fsBox.pattern.patterns.patterns] {
  if {[string length [info commands XFDestroy]] > 0} {
    catch {XFDestroy $fsBoxCounter}
  } {
    catch {destroy $fsBoxCounter}
  }
}
menu .fsBox.pattern.patterns.patterns.menu
catch ".fsBox.pattern.patterns.patterns.menu config $tmpButtonOpt"

if {$fsBox(extensions)} {
  .fsBox.pattern.patterns.patterns.menu add command  -label "*"  -command {
      global fsBox
      set fsBox(pattern) "*"
      .fsBox.pattern.pattern delete 0 end
      .fsBox.pattern.pattern insert 0 $fsBox(pattern)
      FSBoxFSShow [.fsBox.path.path get] $fsBox(pattern)  $fsBox(all)}
}

if {"$fsBoxPath" != "/"} {
  .fsBox.files.files insert end "../"
}
foreach fsBoxCounter $fsBoxElementList {
  if {[string match $fsBoxPattern $fsBoxCounter] ||
      [IsADir [string trimright $fsBoxPath/$fsBoxCounter "/@"]]} {
    if {"$fsBoxCounter" != "../" &&
        "$fsBoxCounter" != "./"} {
```

```
        .fsBox.files.files insert end $fsBoxCounter
      }
    }

    if {$fsBox(extensions)} {
      catch "file rootname $fsBoxCounter" fsBoxRootName
      catch "file extension $fsBoxCounter" fsBoxExtension
      set fsBoxExtension [string trimright $fsBoxExtension "/*@"]
      if {"$fsBoxExtension" != "" && "$fsBoxRootName" != ""} {
        set fsBoxInsert 1
        set fsBoxLast [.fsBox.pattern.patterns.patterns.menu index last]
        for {set fsBoxCounter1 0} {$fsBoxCounter1 <= $fsBoxLast} {incr fsBox-
Counter1 1} {
          if {"*$fsBoxExtension" ==  \
      "[lindex [.fsBox.pattern.patterns.patterns.menu entryconfigure  \
      $fsBoxCounter1 \
      -label] 4]"} {
            set fsBoxInsert 0
          }
        }
if {$fsBoxInsert} {
          .fsBox.pattern.patterns.patterns.menu add command  -label "*$fsBox-
Extension"  -command "
              global fsBox
              set fsBox(pattern) \"*$fsBoxExtension\"
              .fsBox.pattern.pattern delete 0 end
              .fsBox.pattern.pattern insert 0 \$fsBox(pattern)
              FSBoxFSShow \[.fsBox.path.path get\] \$fsBox(pattern) \
  \$fsBox(all)"
        }
      }
    }
  }
  if {$fsBox(extensions)} {
    .fsBox.pattern.patterns.patterns.menu add separator
  }
  if {$fsBox(extensions) ||
      "[.fsBox.pattern.patterns.patterns.menu index last]" == "none"} {
    .fsBox.pattern.patterns.patterns.menu add checkbutton  -label "Scan exten-
sions"  -variable "fsBox(extensions)"  -command {
        global fsBox
        FSBoxFSShow [.fsBox.path.path get]  [.fsBox.pattern.pattern get] \
    $fsBox(all)}
  }
}
```

```
# Procedure: IsADir
proc IsADir { pathName} {
# xf ignore me 5
##########
# Procedure: IsADir
# Description: check if name is a directory (including symbolic links)
# Arguments: pathName - the path to check
# Returns: 1 if its a directory, otherwise 0
# Sideeffects: none
##########

  if {[file isdirectory $pathName]} {
    return 1
  } {
    catch "file type $pathName" fileType
    if {"$fileType" == "link"} {
      if {[catch "file readlink $pathName" linkName]} {
        return 0
      }
      catch "file type $linkName" fileType
      while {"$fileType" == "link"} {
        if {[catch "file readlink $linkName" linkName]} {
          return 0
        }
        catch "file type $linkName" fileType
      }
      return [file isdirectory $linkName]
    }
  }
  return 0
}


# Procedure: IsAFile
proc IsAFile { fileName} {
# xf ignore me 5
##########
# Procedure: IsAFile
# Description: check if filename is a file (including symbolic links)
# Arguments: fileName - the filename to check
# Returns: 1 if its a file, otherwise 0
# Sideeffects: none
##########
```

```
  if {[file isfile $fileName]} {
    return 1
  } {
    catch "file type $fileName" fileType
    if {"$fileType" == "link"} {
      if {[catch "file readlink $fileName" linkName]} {
        return 0
      }
      catch "file type $linkName" fileType
      while {"$fileType" == "link"} {
        if {[catch "file readlink $linkName" linkName]} {
          return 0
        }
        catch "file type $linkName" fileType
      }
      return [file isfile $linkName]
    }
  }
  return 0
}


# Procedure: IsASymlink
proc IsASymlink { fileName} {
# xf ignore me 5
##########
# Procedure: IsASymlink
# Description: check if filename is a symbolic link
# Arguments: fileName - the path/filename to check
# Returns: none
# Sideeffects: none
##########

  catch "file type $fileName" fileType
  if {"$fileType" == "link"} {
    return 1
  }
  return 0
}


# Procedure: Ls
if {"[info procs Ls]" == ""} {
proc Ls { args} {
```

```
# xf ignore me 7
##########
# Procedure: Ls
# Description: emulate UNIX ls
# Arguments: like UNIX (switches authorized -F -a [-a is ignored])
# Returns: directory list
# Sideeffects: none
##########
   set last [lindex $args end]
   if {[lsearch $last -* ] >= 0 || $last == "" } {
     set path "*"
   } else {
     set path $last/*
   }
   set lst [glob $path]
   set lst1 ""
   if {[lsearch -exact $args "-F"] >= 0} then {set car "/"} else {set car ""}
   foreach f $lst {
      if {[file isdirectory $f]}  {
set f [file tail $f]$car
      } else {
set f [file tail $f]
      }
      lappend lst1 $f
   }
   return [lsort $lst1]
}
}



# Procedure: PrintIt
proc PrintIt {} {
  global NowShowing
  set What $NowShowing
  set theCanvas [SN BridgeDisplay]
  set sr [$theCanvas cget -scrollregion]
  global ScalingFactor
  global BridgeWidthInches
  global BridgeLengthInches
  global ViewWidths
  global ViewHeights
  global H0Scale
  set VW [expr $ViewWidths($What) * $H0Scale * $ScalingFactor]i
  set VH [expr $ViewHeights($What) * $H0Scale * $ScalingFactor]i
```

```
  set srx  [$theCanvas create rectangle 0 0 [lindex $sr 2] [lindex $sr 3]]
  set srbb [$theCanvas bbox $srx]
  $theCanvas delete $srx
#  puts "*** srbb = $srbb"
  set vrx  [$theCanvas create rectangle 0 0 $VW $VH]
  set vrbb [$theCanvas bbox $vrx]
  $theCanvas delete $vrx
#  puts "*** vrbb = $vrbb"
  set XDiff [expr [lindex $srbb 2] - [lindex $vrbb 2]]
  set YDiff [expr [lindex $srbb 3] - [lindex $vrbb 3]]
#  puts "*** XDiff = $XDiff, YDiff = $YDiff"
  set XOffsetPix [expr $XDiff / 2.0]
  set YOffsetPix [expr $YDiff / 2.0]
  set inx [$theCanvas create rectangle 0 0 1i 1i]
  set inbb [$theCanvas bbox $inx]
  $theCanvas delete $inx
  set XInches [lindex $inbb 2]
  set YInches [lindex $inbb 3]
  set XOffset [expr ($XOffsetPix / $XInches) - 1.0]
  set YOffset [expr ($YOffsetPix / $YInches) - 1.0]
#  puts "*** XOffset = $XOffset, YOffset = $YOffset"
  set XSizeInches [expr (ceil(double([lindex $vrbb 2]) / $XInches)) + 2.0]
  set YSizeInches [expr (ceil(double([lindex $vrbb 3]) / $YInches)) + 2.0]
#  puts "*** XSizeInches = $XSizeInches, YSizeInches = $YSizeInches"
  set SXSizeInches [expr (ceil(double([lindex $srbb 2]) / $XInches))]
  set SYSizeInches [expr (ceil(double([lindex $srbb 3]) / $YInches))]

  if {$XOffset < 0 || $XSizeInches > $SXSizeInches} {
    set XOffset 0.0
    set XSizeInches $SXSizeInches
  }
  if {$YOffset < 0 || $YSizeInches > $SYSizeInches} {
    set YOffset 0.0
    set YSizeInches $SYSizeInches
  }

  if {"[info procs XFEdit]" != ""} {
    catch "XFDestroy .printForm"
  } {
    catch "destroy .printForm"
  }

  set NOP [expr int(ceil(ceil($XSizeInches / 9.0) * \
 ceil($YSizeInches / 6.0)))]
```

```
  set oldCursor "[. cget -cursor]"
  . config -cursor watch

  global PrintTo
  set prFile stderr

  if {$NOP == 1} {
    if {"$PrintTo" == "Printer"} {
      global PrinterCommand
      set fileName "|$PrinterCommand"
    } elseif {"$PrintTo" == "File"} {
      global PrintFileName
      if {[file exists $PrintFileName]} {
        if {![YesNoBox "Overwrite existing $PrintFileName?"]} {
  . config -cursor "$oldCursor"
  return
}
      }
      set fileName "$PrintFileName"
    } else {
      tkerror "Interal error - don't know where to print to!"
      . config -cursor "$oldCursor"
      return
    }
    if {[catch "$theCanvas postscript -file \"$fileName\" \
-height [lindex $vrbb 3] -width [lindex $vrbb 2] \
-pageanchor center \
-rotate 1 \
-x $XOffsetPix -y $YOffsetPix" error]} {
      tkerror "Could not execute postscript command: $theCanvas postscript -
file \"$fileName\" ...: $error"
    }
    . config -cursor "$oldCursor"
    return
  }

# .printingWaitBox
# The above line makes pasting MUCH easier for me.
# It contains the pathname of the cutted widget.
# Tcl version: 7.6 (Tcl/Tk/XF)
# Tk version: 4.2
# XF version: 4.0
#

  # build widget .printingWaitBox
```

```
  if {"[info procs XFEdit]" != ""} {
    catch "XFDestroy .printingWaitBox"
  } {
    catch "destroy .printingWaitBox"
  }
  toplevel .printingWaitBox

  # Window manager configurations
  wm positionfrom .printingWaitBox ""
  wm sizefrom .printingWaitBox ""
  wm maxsize .printingWaitBox 1137 870
  wm minsize .printingWaitBox 1 1
  wm protocol .printingWaitBox WM_DELETE_WINDOW {}
  wm title .printingWaitBox {Printing in progress...}
  wm transient .printingWaitBox .
  wm geometry .printingWaitBox "+512+384"


  # build widget .printingWaitBox.message1
  message .printingWaitBox.message1 \
    -aspect {1500} \
    -padx {5} \
    -pady {2} \
    -text "Printing Page 1 of $NOP"

  # pack master .printingWaitBox
  pack configure .printingWaitBox.message1
# end of widget tree

  update
  grab .printingWaitBox

  if {"$PrintTo" == "Printer"} {
    global PrinterCommand
    if {[catch "open \"|$PrinterCommand\" w" prFile]} {
      tkerror "Could not open pipe to $PrinterCommand: $prFile"
      . config -cursor "$oldCursor"
      return
    }
  } elseif {"$PrintTo" == "File"} {
    global PrintFileName
    if {[file exists $PrintFileName]} {
      if {![YesNoBox "Overwrite existing $PrintFileName?"]} {
. config -cursor "$oldCursor"
return
```

```
      }
    }
    if {[catch "open \"$PrintFileName\" w" prFile]} {
      tkerror "Could not $PrintFileName for output: $prFile"
      . config -cursor "$oldCursor"
      return
    }
  } else {
    tkerror "Interal error - don't know where to print to!"
    . config -cursor "$oldCursor"
    return
  }

  puts $prFile "%!PS-Adobe-2.0"
  puts $prFile "%%Creator: Bridge.tcl Copyright 1997 Robert Heller"
  puts $prFile "%%Title: Bridge: $What"
  puts -nonewline $prFile "%%CreationDate: "
  global tcl_version
  if {$tcl_version >= 7.6} {
    puts $prFile "[clock format [clock seconds]]"
  } else {
    puts $prFile "[exec date]"
  }
  puts $prFile "%%Pages: $NOP"
  puts $prFile "%%EndComments"
  puts $prFile "/EncapDict 200 dict def EncapDict begin"
  puts $prFile "/showpage {} def /erasepage {} def /copypage {} def end"
  puts $prFile "/BeginInclude {0 setgray 0 setlinecap 1 setlinewidth"
  puts $prFile "0 setlinejoin 10 setmiterlimit \[\] 0 setdash"
  puts $prFile "/languagelevel where {"
  puts $prFile "  pop"
  puts $prFile "  languagelevel 2 ge {"
  puts $prFile "    false setoverprint"
  puts $prFile "    false setstrokeadjust"
  puts $prFile "  } if"
  puts $prFile "} if"
  puts $prFile "newpath"
  puts $prFile "save EncapDict begin} def"
  puts $prFile "/EndInclude {restore end} def"
  puts $prFile "%%EndProlog"
  set pageNo 1
  $theCanvas delete PageBox
  set XPixIn [expr 1.0 / double($XInches)]
  set YPixIn [expr 1.0 / double($YInches)]
  for {set yoff 0.0} {$yoff < $YSizeInches} {set yoff [expr $yoff + 6.0]} {
```

```
    set height 6.0
    if {[expr $yoff + $height] > $YSizeInches} {
      set height [expr $YSizeInches - $yoff]
    }
    for {set xoff 0.0} {$xoff < $XSizeInches} {set xoff [expr $xoff + 9.0]} {
      puts $prFile "%%Page: $pageNo $pageNo"
      puts $prFile "BeginInclude"
      set width 9.0
      if {[expr $xoff + $width] > $XSizeInches} {
set width [expr $XSizeInches - $xoff]
      }
      set xoff1 [expr ($xoff + $XOffset) - $XPixIn]
      set yoff1 [expr ($yoff + $YOffset) - $YPixIn]
      set height1 [expr $height + $YPixIn + $YPixIn]
      set width1 [expr $width + $XPixIn + $XPixIn]
      if {$xoff1 < 0} {set xoff1 0.0}
      if {$yoff1 < 0} {set yoff1 0.0}
      if {[expr $height1 + $yoff1] > $SYSizeInches} {set height1 [expr $SY-
SizeInches - $yoff1]}
      if {[expr $width1 + $xoff1] > $SXSizeInches} {set width1 [expr $SX-
SizeInches - $xoff1]}
#      puts "*** yoff = $yoff, xoff = $xoff, height = $height, \
 width = $width,\n*** xoff1 = $xoff1, yoff1 = $yoff1, \
 height1 = $height1, width1 = $width1"
      $theCanvas create rectangle [expr $xoff + $XOff-
set]i [expr $yoff + $YOffset]i \
         [expr $xoff + $XOffset + $width]i [expr $yoff + $YOff-
set + $height]i \
  -fill {} -outline black -tags {PageBox}
      .printingWaitBox.message1 config -text "Printing Page $pageNo of $NOP"
      update
      set eps "[$theCanvas postscript -height [expr $height1]i -
width [expr $width1]i \
      -x [expr $xoff1]i -y [expr $yoff1]i \
      -pageanchor nw -rotate 1 -pagex 1i -pagey 1i]"
      $theCanvas delete PageBox
      set EOC [string first "%%BeginProlog\n" "$eps"]
      set EOF [expr [string first "%%EOF\n" "$eps"] - 1]
      puts $prFile "[StripPSComments [string range $eps $EOC $EOF]]"
      puts $prFile "EndInclude showpage"
      incr pageNo
    }
  }
  puts $prFile "%%EOF"
  close $prFile
```

```
  if {"[info procs XFEdit]" != ""} {
    catch "XFDestroy .printingWaitBox"
  } {
    catch "destroy .printingWaitBox"
  }
  . config -cursor "$oldCursor"
}

# Procedure: StripPSComments
proc StripPSComments {PSString} {
  set result {}
  foreach l [split "$PSString" "\n"] {
#    puts stderr "*** StripPSComments: l = $l"
    set i [string first "%" "$l$"]
    if {$i == 0} {
#      puts stderr "*** replaced with newline"
      set result "$result\n"
    } elseif {$i > 0 && [regexp {(^.*[^\\])(%.*$)} "$l" whole prefix com-
ment]} {
#      puts stderr "*** replaced with $prefix"
      set result "$result$prefix\n"
    } else {
      set result "$result$l\n"
    }
  }
  return "$result"
}

# Procedure: ReShow
proc ReShow {} {
  global NowShowing
  DoShow $NowShowing
}

# Procedure: StringerGusset
proc StringerGusset {theCanvas b leftRight XCenter YCenter} {
  global H0Scale
  global ScalingFactor
  set i9 [expr 9.0 * $H0Scale * $ScalingFactor]
  set i6 [expr 6.0 * $H0Scale * $ScalingFactor]
  set i3 [expr 3.0 * $H0Scale * $ScalingFactor]
  set i4_5 [expr 4.5 * $H0Scale * $ScalingFactor]
  set i1_5 [expr 1.5 * $H0Scale * $ScalingFactor]
  if {($b == 0 || $b == 3)&& $leftRight == 1} {
    $theCanvas create rectangle \
```

```
[expr $XCenter - $i4_5]i [expr $YCenter - $i6]i \
[expr $XCenter + $i4_5]i [expr $YCenter + $i3]i \
-fill green -outline black -tags {StringerGusset}
  } elseif {($b == 0 || $b == 3)&& $leftRight == -1} {
    $theCanvas create rectangle \
[expr $XCenter - $i4_5]i [expr $YCenter - $i3]i \
[expr $XCenter + $i4_5]i [expr $YCenter + $i6]i \
-fill green -outline black -tags {StringerGusset}
  } elseif {$leftRight == 1} {
    $theCanvas create rectangle \
[expr $XCenter - $i9]i [expr $YCenter - $i6]i \
[expr $XCenter + $i9]i [expr $YCenter + $i1_5]i \
-fill green -outline black -tags {StringerGusset}
  } elseif {$leftRight == -1} {
    $theCanvas create rectangle \
[expr $XCenter - $i9]i [expr $YCenter - $i3]i \
[expr $XCenter + $i9]i [expr $YCenter + $i6]i \
-fill green -outline black -tags {StringerGusset}
  }
}


# Procedure: UnderFloorBraceGussets
proc UnderFloorBraceGussets {theCanvas flbLength flbCenterLine NextflbCenter-
Line i XOffset YOffset} {
  global H0Scale
  global ScalingFactor
  set i42 [expr 42.0 * $H0Scale * $ScalingFactor]
  set i21 [expr 21.0 * $H0Scale * $ScalingFactor]
  set i21_2 [expr $i21 * .5]
  set i21_15 [expr $i21 * 1.5]
#  set i63 [expr 63.0 * $H0Scale * $ScalingFactor]
  if {$i == 0} {
    set GLeft [expr $flbCenterLine - $i21_2]
    UnderFloorBraceGussetsA $theCanvas $GLeft $XOffset $YOffset
    UnderFloorBraceGussetsB $theCanvas $GLeft $XOffset [expr $YOff-
set + $i42 + $flbLength]
  } else {
    set GLeft [expr $flbCenterLine - $i21_15]
    UnderFloorBraceGussetsC $theCanvas $GLeft $XOffset $YOffset
    UnderFloorBraceGussetsD $theCanvas $GLeft $XOffset [expr $YOff-
set + $i42 + $flbLength]
  }
#  set i8 [expr 8.0 * $H0Scale * $ScalingFactor]
#  set i4_25 [expr 4.25 * $H0Scale * $ScalingFactor]
```

```
#  set i4 [expr 4.0 * $H0Scale * $ScalingFactor]
#  set i6 [expr 6.0 * $H0Scale * $ScalingFactor]
  set YBCenter [expr $YOffset + $i21 + ($flbLength / 2.0)]
  set XBCenter [expr $XOffset + (($NextflbCenterLine - $flbCenter-
Line ) / 2.0) + $flbCenterLine]
  UnderFloorCenterBraceGusset $theCanvas $YBCenter $XBCenter
}

# Procedure: UnderFloorBraceGussetsA
proc UnderFloorBraceGussetsA {theCanvas GLeft XOffset YOffset} {
  global H0Scale
  global ScalingFactor
  set i42 [expr 42.0 * $H0Scale * $ScalingFactor]
  set i21 [expr 21.0 * $H0Scale * $ScalingFactor]
  set i21_2 [expr $i21 * .5]
  set i21_15 [expr $i21 * 1.5]
  set i63 [expr 63.0 * $H0Scale * $ScalingFactor]
  $theCanvas create polygon \
[expr $GLeft + $XOffset]i [expr $YOffset]i \
[expr $GLeft + $i42 + $XOffset]i [expr $YOffset]i \
[expr $GLeft + $i42 + $XOffset]i [expr $YOffset + $i21]i \
[expr $GLeft + $i21 + $XOffset]i [expr $YOffset + $i42]i \
[expr $GLeft + $XOffset]i [expr $YOffset + $i42]i \
[expr $GLeft + $XOffset]i [expr $YOffset]i \
-fill green -outline black
}

# Procedure: UnderFloorBraceGussetsB
proc UnderFloorBraceGussetsB {theCanvas GLeft XOffset YOffset} {
  global H0Scale
  global ScalingFactor
  set i42 [expr 42.0 * $H0Scale * $ScalingFactor]
  set i21 [expr 21.0 * $H0Scale * $ScalingFactor]
  set i21_2 [expr $i21 * .5]
  set i21_15 [expr $i21 * 1.5]
  set i63 [expr 63.0 * $H0Scale * $ScalingFactor]
  $theCanvas create polygon \
[expr $GLeft + $XOffset]i [expr $YOffset]i \
[expr $GLeft + $i42 + $XOffset]i [expr $YOffset]i \
[expr $GLeft + $i42 + $XOffset]i [expr $YOffset - $i21]i \
[expr $GLeft + $i21 + $XOffset]i [expr $YOffset - $i42]i \
[expr $GLeft + $XOffset]i [expr $YOffset - $i42]i \
[expr $GLeft + $XOffset]i [expr $YOffset]i \
-fill green -outline black
}
```

```
# Procedure: UnderFloorBraceGussetsC
proc UnderFloorBraceGussetsC {theCanvas GLeft XOffset YOffset} {
  global H0Scale
  global ScalingFactor
  set i42 [expr 42.0 * $H0Scale * $ScalingFactor]
  set i21 [expr 21.0 * $H0Scale * $ScalingFactor]
  set i21_2 [expr $i21 * .5]
  set i21_15 [expr $i21 * 1.5]
  set i63 [expr 63.0 * $H0Scale * $ScalingFactor]
  $theCanvas create polygon \
[expr $GLeft + $XOffset]i [expr $YOffset]i \
[expr $GLeft + $i63 + $XOffset]i [expr $YOffset]i \
[expr $GLeft + $i63 + $XOffset]i [expr $YOffset + $i21]i \
[expr $GLeft + $i42 + $XOffset]i [expr $YOffset + $i42]i \
[expr $GLeft + $i21 + $XOffset]i [expr $YOffset + $i42]i \
[expr $GLeft + $XOffset]i [expr $YOffset + $i21]i \
[expr $GLeft + $XOffset]i [expr $YOffset]i \
-fill green -outline black
}

# Procedure: UnderFloorBraceGussetsD
proc UnderFloorBraceGussetsD {theCanvas GLeft XOffset YOffset} {
  global H0Scale
  global ScalingFactor
  set i42 [expr 42.0 * $H0Scale * $ScalingFactor]
  set i21 [expr 21.0 * $H0Scale * $ScalingFactor]
  set i21_2 [expr $i21 * .5]
  set i21_15 [expr $i21 * 1.5]
  set i63 [expr 63.0 * $H0Scale * $ScalingFactor]
  $theCanvas create polygon \
[expr $GLeft + $XOffset]i [expr $YOffset]i \
[expr $GLeft + $i63 + $XOffset]i [expr $YOffset]i \
[expr $GLeft + $i63 + $XOffset]i [expr $YOffset - $i21]i \
[expr $GLeft + $i42 + $XOffset]i [expr $YOffset - $i42]i \
[expr $GLeft + $i21 + $XOffset]i [expr $YOffset - $i42]i \
[expr $GLeft + $XOffset]i [expr $YOffset - $i21]i \
[expr $GLeft + $XOffset]i [expr $YOffset]i \
-fill green -outline black
}

# Procedure: UnderFloorCenterBraceGusset
proc UnderFloorCenterBraceGusset {theCanvas YBCenter XBCenter} {
  global H0Scale
  global ScalingFactor
```

```
  set i8 [expr 8.0 * $H0Scale * $ScalingFactor]
  set i4_25 [expr 4.25 * $H0Scale * $ScalingFactor]
  set i4 [expr 4.0 * $H0Scale * $ScalingFactor]
  set i6 [expr 6.0 * $H0Scale * $ScalingFactor]
  $theCanvas create polygon \
[expr $XBCenter - $i4]i          [expr $YBCenter - $i8 - $i4_25]i \
[expr $XBCenter + $i4]i          [expr $YBCenter - $i8 - $i4_25]i \
[expr $XBCenter + $i4 + $i4_25]i [expr $YBCenter - $i8]i \
[expr $XBCenter + $i4 + $i4_25]i [expr $YBCenter + $i8]i \
[expr $XBCenter + $i4]i          [expr $YBCenter + $i8 + $i4_25]i \
[expr $XBCenter - $i4]i          [expr $YBCenter + $i8 + $i4_25]i \
[expr $XBCenter - $i4 - $i4_25]i [expr $YBCenter + $i8]i \
[expr $XBCenter - $i4 - $i4_25]i [expr $YBCenter - $i8]i \
[expr $XBCenter - $i4]i          [expr $YBCenter - $i8 - $i4_25]i \
-fill green -outline black
}

# Procedure: LastUnderFloorBraceGussets
proc LastUnderFloorBraceGussets {theCanvas flbLength flbCenterLine XOff-
set YOffset} {
  global H0Scale
  global ScalingFactor
  set i42 [expr 42.0 * $H0Scale * $ScalingFactor]
  set i21 [expr 21.0 * $H0Scale * $ScalingFactor]
  set i21_2 [expr $i21 * .5]
  set GRight [expr $flbCenterLine + $i21_2]
  LastUnderFloorBraceGussetsA $theCanvas $GRight $XOffset $YOffset
  LastUnderFloorBraceGussetsB $theCanvas $GRight $XOffset [expr $YOff-
set + $i42 + $flbLength]
}

# Procedure: LastUnderFloorBraceGussetsA
proc LastUnderFloorBraceGussetsA {theCanvas GRight XOffset YOffset} {
  global H0Scale
  global ScalingFactor
  set i42 [expr 42.0 * $H0Scale * $ScalingFactor]
  set i21 [expr 21.0 * $H0Scale * $ScalingFactor]
  set i21_2 [expr $i21 * .5]
  $theCanvas create polygon \
[expr $GRight + $XOffset]i [expr $YOffset]i \
[expr $GRight - $i42 + $XOffset]i [expr $YOffset]i \
[expr $GRight - $i42 + $XOffset]i [expr $YOffset + $i21]i \
[expr $GRight - $i21 + $XOffset]i [expr $YOffset + $i42]i \
[expr $GRight + $XOffset]i [expr $YOffset + $i42]i \
[expr $GRight + $XOffset]i [expr $YOffset]i \
```

```
-fill green -outline black
}

# Procedure: LastUnderFloorBraceGussetsB
proc LastUnderFloorBraceGussetsB {theCanvas GRight XOffset YOffset} {
  global H0Scale
  global ScalingFactor
  set i42 [expr 42.0 * $H0Scale * $ScalingFactor]
  set i21 [expr 21.0 * $H0Scale * $ScalingFactor]
  set i21_2 [expr $i21 * .5]
  $theCanvas create polygon \
[expr $GRight + $XOffset]i [expr $YOffset]i \
[expr $GRight - $i42 + $XOffset]i [expr $YOffset]i \
[expr $GRight - $i42 + $XOffset]i [expr $YOffset - $i21]i \
[expr $GRight - $i21 + $XOffset]i [expr $YOffset - $i42]i \
[expr $GRight + $XOffset]i [expr $YOffset - $i42]i \
[expr $GRight + $XOffset]i [expr $YOffset]i \
-fill green -outline black
}

# Procedure: PostOnlyBottomSideGusset
proc PostOnlyBottomSideGusset {theCanvas bottomY CenterX} {
  global H0Scale
  global ScalingFactor
  set i42 [expr 42.0 * $H0Scale * $ScalingFactor]
  set i21 [expr 21.0 * $H0Scale * $ScalingFactor]
  set i11 [expr 11.0 * $H0Scale * $ScalingFactor]
  set i5F8 [expr ((5 * 12) + 8) * $H0Scale * $ScalingFactor]
  set i4F6 [expr ((4 * 12) + 6) * $H0Scale * $ScalingFactor]
  set i4F6_2 [expr $i4F6 * 0.5]
  $theCanvas create polygon \
[expr $CenterX - $i4F6_2]i [expr $bottomY]i \
[expr $CenterX - $i4F6_2]i [expr $bottomY - $i21]i \
[expr $CenterX - $i11]i [expr $bottomY - $i5F8]i \
[expr $CenterX + $i11]i [expr $bottomY - $i5F8]i \
[expr $CenterX + $i4F6_2]i [expr $bottomY - $i21]i \
[expr $CenterX + $i4F6_2]i [expr $bottomY]i \
[expr $CenterX - $i4F6_2]i [expr $bottomY]i \
-fill green -outline black -tags {BottomSideGusset}
}

# Procedure: PostOnlyTopSideGusset
proc PostOnlyTopSideGusset {theCanvas CenterX CenterY pTanA nTanA} {
  global H0Scale
  global ScalingFactor
```

```
  set i42 [expr 42.0 * $H0Scale * $ScalingFactor]
  set i21 [expr 21.0 * $H0Scale * $ScalingFactor]
  set i11 [expr 11.0 * $H0Scale * $ScalingFactor]
  set i5F8 [expr ((5 * 12) + 8) * $H0Scale * $ScalingFactor]
  set i4F6 [expr ((4 * 12) + 6) * $H0Scale * $ScalingFactor]
  set i4F6_2 [expr $i4F6 * 0.5]
  set YRight [expr ($nTanA * $i4F6_2) + $CenterY]
  set YLeft  [expr ($pTanA * $i4F6_2) + $CenterY]
  $theCanvas create polygon \
[expr $CenterX - $i4F6_2]i [expr $YLeft]i \
[expr $CenterX]i [expr $CenterY]i \
[expr $CenterX + $i4F6_2]i [expr $YRight]i \
[expr $CenterX + $i4F6_2]i [expr $YRight + $i21]i \
[expr $CenterX + $i11]i [expr $CenterY + $i5F8]i \
[expr $CenterX - $i11]i [expr $CenterY + $i5F8]i \
[expr $CenterX - $i4F6_2]i [expr $YLeft + $i21]i \
[expr $CenterX - $i4F6_2]i [expr $YLeft]i \
-fill green -outline black -tags {TopSideGusset}
}

# Procedure: TwoBarBottomSideGusset
proc TwoBarBottomSideGusset {theCanvas bottomY CenterX} {
  global H0Scale
  global ScalingFactor
  set i42 [expr 42.0 * $H0Scale * $ScalingFactor]
  set i21 [expr 21.0 * $H0Scale * $ScalingFactor]
  set i8F [expr (8 * 12) * $H0Scale * $ScalingFactor]
  set i8F_2 [expr $i8F * .5]
  set i6F [expr (6 * 12) * $H0Scale * $ScalingFactor]
  set i30 [expr 30.0 * $H0Scale * $ScalingFactor]
  set i30_2 [expr $i30 * .5]
  set i12 [expr 12.0 * $H0Scale * $ScalingFactor]
  $theCanvas create polygon \
[expr $CenterX - $i8F_2]i [expr $bottomY]i \
[expr $CenterX - $i8F_2]i [expr $bottomY - ($i6F - $i12)]i \
[expr $CenterX - $i30_2]i [expr $bottomY - $i6F]i \
[expr $CenterX + $i30_2]i [expr $bottomY - $i6F]i \
[expr $CenterX + $i8F_2]i [expr $bottomY - ($i6F - $i12)]i \
[expr $CenterX + $i8F_2]i [expr $bottomY]i \
[expr $CenterX - $i8F_2]i [expr $bottomY]i \
-fill green -outline black -tags {BottomSideGusset}

}

# Procedure: TwoBarTopSideGusset
```

```
proc TwoBarTopSideGusset {theCanvas CenterX CenterY pTanA nTanA} {
  global H0Scale
  global ScalingFactor
  set i42 [expr 42.0 * $H0Scale * $ScalingFactor]
  set i21 [expr 21.0 * $H0Scale * $ScalingFactor]
  set i8F [expr (8 * 12) * $H0Scale * $ScalingFactor]
  set i8F_2 [expr $i8F * .5]
  set i6F [expr (6 * 12) * $H0Scale * $ScalingFactor]
  set i30 [expr 30.0 * $H0Scale * $ScalingFactor]
  set i30_2 [expr $i30 * .5]
  set i12 [expr 12.0 * $H0Scale * $ScalingFactor]
  set YRight [expr ($nTanA * $i8F_2) + $CenterY]
  set YLeft  [expr ($pTanA * $i8F_2) + $CenterY]
  $theCanvas create polygon \
[expr $CenterX - $i8F_2]i [expr $YLeft]i \
[expr $CenterX]i [expr $CenterY]i \
[expr $CenterX + $i8F_2]i [expr $YRight]i \
[expr $CenterX + $i8F_2]i [expr $YRight + $i21]i \
[expr $CenterX + $i30_2]i [expr $CenterY + ($i6F - $i12)]i \
[expr $CenterX - $i30_2]i [expr $CenterY + ($i6F - $i12)]i \
[expr $CenterX - $i8F_2]i [expr $YLeft + $i21]i \
[expr $CenterX - $i8F_2]i [expr $YLeft]i \
-fill green -outline black -tags {TopSideGusset}
}

# Procedure: LeftBottomSideGusset
proc LeftBottomSideGusset {theCanvas bottomY LeftX} {
  global H0Scale
  global ScalingFactor
#  set i42 [expr 42.0 * $H0Scale * $ScalingFactor]
  set i21 [expr 21.0 * $H0Scale * $ScalingFactor]
  set i6F [expr (6 * 12) * $H0Scale * $ScalingFactor]
#  set i30 [expr 30.0 * $H0Scale * $ScalingFactor]
#  set i30_2 [expr $i30 * .5]
  set i12 [expr 12.0 * $H0Scale * $ScalingFactor]
  $theCanvas create polygon \
[expr $LeftX]i [expr $bottomY]i \
[expr $LeftX]i [expr $bottomY - $i6F]i \
[expr $LeftX + $i6F - $i12]i [expr $bottomY - $i6F]i \
[expr $LeftX + $i6F]i [expr $bottomY - $i21]i \
[expr $LeftX + $i6F]i [expr $bottomY]i \
[expr $LeftX]i [expr $bottomY]i \
-fill green -outline black -tags {BottomSideGusset}
}
```

```
# Procedure: RightBottomSideGusset
proc RightBottomSideGusset {theCanvas bottomY RightX} {
  global H0Scale
  global ScalingFactor
#  set i42 [expr 42.0 * $H0Scale * $ScalingFactor]
  set i21 [expr 21.0 * $H0Scale * $ScalingFactor]
  set i6F [expr (6 * 12) * $H0Scale * $ScalingFactor]
#  set i30 [expr 30.0 * $H0Scale * $ScalingFactor]
#  set i30_2 [expr $i30 * .5]
  set i12 [expr 12.0 * $H0Scale * $ScalingFactor]
  $theCanvas create polygon \
[expr $RightX]i [expr $bottomY]i \
[expr $RightX]i [expr $bottomY - $i6F]i \
[expr $RightX - $i6F + $i12]i [expr $bottomY - $i6F]i \
[expr $RightX - $i6F]i [expr $bottomY - $i21]i \
[expr $RightX - $i6F]i [expr $bottomY]i \
[expr $RightX]i [expr $bottomY]i \
-fill green -outline black -tags {BottomSideGusset}
}

# Procedure: LeftTopSideGusset
proc LeftTopSideGusset {theCanvas CenterX CenterY pTanA nTanA} {
  global H0Scale
  global ScalingFactor
#  set i42 [expr 42.0 * $H0Scale * $ScalingFactor]
  set i21 [expr 21.0 * $H0Scale * $ScalingFactor]
  set i21_2 [expr $i21 * 0.5]
  set i6F [expr (6 * 12) * $H0Scale * $ScalingFactor]
  set i6F_2 [expr $i6F * .5]
  set i30 [expr 30.0 * $H0Scale * $ScalingFactor]
  set i30_2 [expr $i30 * .5]
  set i12 [expr 12.0 * $H0Scale * $ScalingFactor]
  set i5  [expr 5.0 * $H0Scale * $ScalingFactor]
  set YRight [expr ($nTanA * $i6F_2) + $CenterY]
  set YLeft  [expr ($pTanA * $i6F_2) + $CenterY]
  $theCanvas create polygon \
[expr $CenterX - $i6F_2 - $i5]i [expr $YLeft]i \
[expr $CenterX - $i21_2]i [expr $CenterY]i \
[expr $CenterX + $i6F_2]i [expr $YRight]i \
[expr $CenterX + $i6F_2]i [expr $YRight + $i30]i \
[expr $CenterX + $i30_2]i [expr $YLeft]i \
[expr $CenterX - $i6F_2]i [expr $YLeft]i \
-fill green -outline black -tags {TopSideGusset}
}
```

```
# Procedure: RightTopSideGusset
proc RightTopSideGusset {theCanvas CenterX CenterY pTanA nTanA} {
#  puts stderr "*** RightTopSideGusset $theCanvas $CenterX $Cen-
terY $pTanA $nTanA"
  global H0Scale
  global ScalingFactor
#  set i42 [expr 42.0 * $H0Scale * $ScalingFactor]
  set i21 [expr 21.0 * $H0Scale * $ScalingFactor]
  set i21_2 [expr $i21 * 0.5]
  set i6F [expr (6 * 12) * $H0Scale * $ScalingFactor]
  set i6F_2 [expr $i6F * .5]
  set i30 [expr 30.0 * $H0Scale * $ScalingFactor]
  set i30_2 [expr $i30 * .5]
  set i12 [expr 12.0 * $H0Scale * $ScalingFactor]
  set i5  [expr 5.0 * $H0Scale * $ScalingFactor]
  set YRight [expr ($nTanA * $i6F_2) + $CenterY]
  set YLeft  [expr ($pTanA * $i6F_2) + $CenterY]
  $theCanvas create polygon \
[expr $CenterX + $i6F_2 + $i5]i [expr $YRight]i \
[expr $CenterX + $i21_2]i [expr $CenterY]i \
[expr $CenterX - $i6F_2]i [expr $YLeft]i \
[expr $CenterX - $i6F_2]i [expr $YLeft + $i30]i \
[expr $CenterX - $i30_2]i [expr $YRight]i \
[expr $CenterX + $i6F_2]i [expr $YRight]i \
-fill green -outline black -tags {TopSideGusset}
}

# Procedure: YesNoBox
proc YesNoBox { {yesNoBoxMessage "Yes/no message"} {yesNoBoxGeome-
try "350x150+512+384"}} {
# xf ignore me 5
##########
# Procedure: YesNoBox
# Description: show yesno box
# Arguments: {yesNoBoxMessage} - the text to display
#            {yesNoBoxGeometry} - the geometry for the window
# Returns: none
# Sideeffects: none
##########
#
# global yesNoBox(activeBackground) - active background color
# global yesNoBox(activeForeground) - active foreground color
# global yesNoBox(anchor) - anchor for message box
# global yesNoBox(background) - background color
# global yesNoBox(font) - message font
```

```
# global yesNoBox(foreground) - foreground color
# global yesNoBox(justify) - justify for message box
# global yesNoBox(afterNo) - destroy yes-no box after n seconds.
#                             The no button is activated
# global yesNoBox(afterYes) - destroy yes-no box after n seconds.
#                              The yes button is activated

  global yesNoBox

  set tmpButtonOpt ""
  set tmpFrameOpt ""
  set tmpMessageOpt ""
  if {"$yesNoBox(activeBackground)" != ""} {
    append tmpButtonOpt "-activebackground \"$yesNoBox(activeBackground)\" "
  }
  if {"$yesNoBox(activeForeground)" != ""} {
    append tmpButtonOpt "-activeforeground \"$yesNoBox(activeForeground)\" "
  }
  if {"$yesNoBox(background)" != ""} {
    append tmpButtonOpt "-background \"$yesNoBox(background)\" "
    append tmpFrameOpt "-background \"$yesNoBox(background)\" "
    append tmpMessageOpt "-background \"$yesNoBox(background)\" "
  }
  if {"$yesNoBox(font)" != ""} {
    append tmpButtonOpt "-font \"$yesNoBox(font)\" "
    append tmpMessageOpt "-font \"$yesNoBox(font)\" "
  }
  if {"$yesNoBox(foreground)" != ""} {
    append tmpButtonOpt "-foreground \"$yesNoBox(foreground)\" "
    append tmpMessageOpt "-foreground \"$yesNoBox(foreground)\" "
  }

  # start build of toplevel
  if {"[info commands XFDestroy]" != ""} {
    catch {XFDestroy .yesNoBox}
  } {
    catch {destroy .yesNoBox}
  }
  toplevel .yesNoBox  -borderwidth 0
  catch ".yesNoBox config $tmpFrameOpt"
  if {[catch "wm geometry .yesNoBox $yesNoBoxGeometry"]} {
    wm geometry .yesNoBox 350x150+512+384
  }
  wm title .yesNoBox {Alert box}
  wm maxsize .yesNoBox 1000 1000
```

```
  wm minsize .yesNoBox 100 100
  wm transient .yesNoBox .
  # end build of toplevel

  message .yesNoBox.message1  -anchor "$yesNoBox(anchor)"  -
justify "$yesNoBox(justify)"  -relief raised  -text "$yesNoBoxMessage"
  catch ".yesNoBox.message1 config $tmpMessageOpt"

  set xfTmpWidth  [string range $yesNoBoxGeometry 0 \
  [expr [string first x $yesNoBoxGeometry]-1]]
  if {"$xfTmpWidth" != ""} {
    # set message size
    catch ".yesNoBox.message1 configure  -width [expr $xfTmpWidth-10]"
  } {
    .yesNoBox.message1 configure  -aspect 1500
  }

  frame .yesNoBox.frame1  -borderwidth 0  -relief raised
  catch ".yesNoBox.frame1 config $tmpFrameOpt"

  button .yesNoBox.frame1.button0  -text "Yes"  -command "
      global yesNoBox
      set yesNoBox(button) 1
      if {\"\[info commands XFDestroy\]\" != \"\"} {
        catch {XFDestroy .yesNoBox}
      } {
        catch {destroy .yesNoBox}
      }"
  catch ".yesNoBox.frame1.button0 config $tmpButtonOpt"

  button .yesNoBox.frame1.button1  -text "No"  -command "
      global yesNoBox
      set yesNoBox(button) 0
      if {\"\[info commands XFDestroy\]\" != \"\"} {
        catch {XFDestroy .yesNoBox}
      } {
        catch {destroy .yesNoBox}
      }"
  catch ".yesNoBox.frame1.button1 config $tmpButtonOpt"

  pack append .yesNoBox.frame1  \
.yesNoBox.frame1.button0 {left fillx expand}  \
.yesNoBox.frame1.button1 {left fillx expand}

  # packing
```

```
  pack append .yesNoBox  .yesNoBox.frame1 {bottom fill}  \
.yesNoBox.message1 {top fill expand}

  if {$yesNoBox(afterYes) != 0} {
    after [expr $yesNoBox(afterYes)*1000]  \
"catch \".yesNoBox.frame1.button0 invoke\""
  }
  if {$yesNoBox(afterNo) != 0} {
    after [expr $yesNoBox(afterNo)*1000]  \
"catch \".yesNoBox.frame1.button1 invoke\""
  }

  # wait for the box to be destroyed
  update idletask
  grab .yesNoBox
  tkwait window .yesNoBox

  return $yesNoBox(button)
}


# User defined images


# Internal procedures


# Procedure: Alias
if {"[info procs Alias]" == ""} {
proc Alias { args} {
# xf ignore me 7
##########
# Procedure: Alias
# Description: establish an alias for a procedure
# Arguments: args - no argument means that a list of all aliases
#                   is returned. Otherwise the first parameter is
#                   the alias name, and the second parameter is
#                   the procedure that is aliased.
# Returns: nothing, the command that is bound to the alias or a
#          list of all aliases - command pairs.
# Sideeffects: internalAliasList is updated, and the alias
#              proc is inserted
##########
  global internalAliasList
```

```
  if {[llength $args] == 0} {
    return $internalAliasList
  } {
    if {[llength $args] == 1} {
      set xfTmpIndex [lsearch $internalAliasList "[lindex $args 0] *"]
      if {$xfTmpIndex != -1} {
        return [lindex [lindex $internalAliasList $xfTmpIndex] 1]
      }
    } {
      if {[llength $args] == 2} {
        eval "proc [lindex $args 0] {args} {#xf ignore me 4
return \[eval \"[lindex $args 1] \$args\"\]}"
        set xfTmpIndex [lsearch $internalAliasList "[lindex $args 0] *"]
        if {$xfTmpIndex != -1} {
          set internalAliasList [lreplace $internalAliasList $xfTmpIn-
dex $xfTmpIndex "[lindex $args 0] [lindex $args 1]"]
        } {
          lappend internalAliasList "[lindex $args 0] [lindex $args 1]"
        }
      } {
        error "Alias: wrong number or args: $args"
      }
    }
  }
}
}


# Procedure: Cat
if {"[info procs Cat]" == ""} {
proc Cat { filename} {
# xf ignore me 7
##########
# Procedure: Cat
# Description: emulate UNIX cat for one file
# Arguments: filename
# Returns: file contents
# Sideeffects: none
##########
global tcl_platform
if {$tcl_platform(platform) == "unix"} {exec cat $filename} {
   set fileid [open $filename "r"]
   set data [read $fileid]
   close $fileid
   return $data
```

```
}
}
}


# Procedure: Chmod
if {"[info procs Chmod]" == ""} {
proc Chmod { mode file} {
# xf ignore me 7
##########
# Procedure: Chmod
# Description: ignore UNIX chmod under DOS
# Arguments: file1 file2/directory
# Returns: nothing
# Sideeffects: none
##########
global tcl_platform
if {$tcl_platform(platform) == "unix"} {eval exec chmod $mode $file} {
   regsub -all {/} $file1 {\\\\} file1
   regsub -all {/} $file2 {\\\\} file2
   eval exec command.com /c copy /y $file1 $file2 >@stderr
}
}
}


# Procedure: Cp
if {"[info procs Cp]" == ""} {
proc Cp { file1 file2} {
# xf ignore me 7
##########
# Procedure: Cp
# Description: emulate UNIX cp with DOS COPY
# Arguments: file1 file2/directory
# Returns: nothing
# Sideeffects: none
##########
global tcl_platform
if {$tcl_platform(platform) == "unix"} {eval exec cp $file1 $file2} {
   regsub -all {/} $file1 {\\\\} file1
   regsub -all {/} $file2 {\\\\} file2
   eval exec command.com /c copy /y $file1 $file2 >@stderr
}
}
}
```

```
# Procedure: GetSelection
if {"[info procs GetSelection]" == ""} {
proc GetSelection {} {
# xf ignore me 7
##########
# Procedure: GetSelection
# Description: get current selection
# Arguments: none
# Returns: none
# Sideeffects: none
##########

  # the save way
  set xfSelection ""
  catch "selection get" xfSelection
  if {"$xfSelection" == "selection doesn't exist or form \"STRING\" not de-
fined"} {
    return ""
  } {
    return $xfSelection
  }
}
}


# Procedure: Ls
if {"[info procs Ls]" == ""} {
proc Ls { args} {
# xf ignore me 7
##########
# Procedure: Ls
# Description: emulate UNIX ls
# Arguments: like UNIX (switches authorized -F -a [-a is ignored])
# Returns: directory list
# Sideeffects: none
##########
global tcl_platform
if {$tcl_platform(platform) == "unix"} {eval exec ls $args} {
   set last [lindex $args end]
   if {[lsearch $last -* ] >= 0 || $last == "" } {
set path "*"
   } else {
set path $last/*
```

```
   }
   set lst [glob $path]
   set lst1 ""
   if {[lsearch -exact $args "-F"] >= 0} then {set car "/"} else {set car ""}
   foreach f $lst {
      if {[file isdirectory $f]}  {
set f [file tail $f]$car
      } else {
set f [file tail $f]
      }
      lappend lst1 $f
   }
   return [lsort $lst1]
}
}
}


# Procedure: NoFunction
if {"[info procs NoFunction]" == ""} {
proc NoFunction { args} {
# xf ignore me 7
##########
# Procedure: NoFunction
# Description: do nothing (especially with scales and scrollbars)
# Arguments: args - a number of ignored parameters
# Returns: none
# Sideeffects: none
##########
}
}


# Procedure: Rm
if {"[info procs Rm]" == ""} {
proc Rm { filename} {
# xf ignore me 7
##########
# Procedure: Rm
# Description: emulate UNIX rm with DOS DEL
# Arguments: filename(s)
# Returns: nothing
# Sideeffects: none
##########
global tcl_platform
```

```
if {$tcl_platform(platform) == "unix"} {exec rm -f $filename} {
   regsub -all {/} $filename {\\\\} filename
   eval exec command.com /c del $filename >@stderr
}
}
}


# Procedure: SN
if {"[info procs SN]" == ""} {
proc SN { {xfName ""}} {
# xf ignore me 7
##########
# Procedure: SN
# Description: map a symbolic name to the widget path
# Arguments: xfName
# Returns: the symbolic name
# Sideeffects: none
##########

  SymbolicName $xfName
}
}


# Procedure: SymbolicName
if {"[info procs SymbolicName]" == ""} {
proc SymbolicName { {xfName ""}} {
# xf ignore me 7
##########
# Procedure: SymbolicName
# Description: map a symbolic name to the widget path
# Arguments: xfName
# Returns: the symbolic name
# Sideeffects: none
##########

  global symbolicName

  if {"$xfName" != ""} {
    set xfArrayName ""
    append xfArrayName symbolicName ( $xfName )
    if {![catch "set \"$xfArrayName\"" xfValue]} {
      return $xfValue
    } {
```

```
      if {"[info commands XFProcError]" != ""} {
        XFProcError "Unknown symbolic name:\n$xfName"
      } {
        puts stderr "XF error: unknown symbolic name:\n$xfName"
      }
    }
  }
  return ""
}
}


# Procedure: Unalias
if {"[info procs Unalias]" == ""} {
proc Unalias { aliasName} {
# xf ignore me 7
##########
# Procedure: Unalias
# Description: remove an alias for a procedure
# Arguments: aliasName - the alias name to remove
# Returns: none
# Sideeffects: internalAliasList is updated, and the alias
#              proc is removed
##########
  global internalAliasList

  set xfIndex [lsearch $internalAliasList "$aliasName *"]
  if {$xfIndex != -1} {
    rename $aliasName ""
    set internalAliasList [lreplace $internalAliasList $xfIndex $xfIndex]
  }
}
}



# application parsing procedure
proc XFLocalParseAppDefs {xfAppDefFile} {
  global xfAppDefaults tcl_platform

  # basically from: Michael Moore
  if {[file exists $xfAppDefFile] &&
      [file readable $xfAppDefFile] &&
      "[file type $xfAppDefFile]" == "link"} {
    catch "file type $xfAppDefFile" xfType
```

```
  while {"$xfType" == "link"} {
    if {[catch "file readlink $xfAppDefFile" xfAppDefFile]} {
      return
    }
    catch "file type $xfAppDefFile" xfType
  }
}
if {!("$xfAppDefFile" != "" &&
      [file exists $xfAppDefFile] &&
      [file readable $xfAppDefFile] &&
      "[file type $xfAppDefFile]" == "file")} {
  return
}
if {![catch "open $xfAppDefFile r" xfResult]} {
  set xfAppFileContents [read $xfResult]
  close $xfResult
  foreach line [split $xfAppFileContents "\n"] {
    # backup indicates how far to backup.  It applies to the
    # situation where a resource name ends in . and when it
    # ends in *.  In the second case you want to keep the *
    # in the widget name for pattern matching, but you want
    # to get rid of the . if it is the end of the name.
    set backup -2
    set line [string trim $line]
    if {[string index $line 0] == "#" || "$line" == ""} {
      # skip comments and empty lines
      continue
    }
    if {![string compare "windows" $tcl_platform(platform)]} {
      set list [split $line ";"]
    } {
      set list [split $line ":"]
    }
    set resource [string trim [lindex $list 0]]
    set i [string last "." $resource]
    set j [string last "*" $resource]
    if {$j > $i} {
      set i $j
      set backup -1
    }
    incr i
    set name [string range $resource $i end]
    incr i $backup
    set widname [string range $resource 0 $i]
    set value [string trim [lindex $list 1]]
```

```
      if {"$widname" != "" && "$widname" != "*"} {
        # insert the widget and resourcename to the application
        # defaults list.
        if {![info exists xfAppDefaults]} {
          set xfAppDefaults ""
        }
        lappend xfAppDefaults [list $widname [string tolower $name] $value]
      }
    }
  }
}

# application loading procedure
proc XFLocalLoadAppDefs {{xfClasses ""} {xfPriority "startupFile"} {xfAppDef-
File ""}} {
  global env tcl_platform

  if {![string compare "windows" $tcl_platform(platform)]} {
    set separator ";"
  } {
    set separator ":"
  }
  if {"$xfAppDefFile" == ""} {
    set xfFileList ""
    if {[info exists env(XUSERFILESEARCHPATH)]} {
      eval lappend xfFileList [split $env(XUSERFILESEARCHPATH) $separator]
    }
    if {[info exists env(XAPPLRESDIR)]} {
      eval lappend xfFileList [split $env(XAPPLRESDIR) $separator]
    }
    if {[info exists env(XFILESEARCHPATH)]} {
      eval lappend xfFileList [split $env(XFILESEARCHPATH) $separator]
    }
    append xfFileList " /usr/lib/X11/app-defaults"
    append xfFileList " /usr/X11/lib/X11/app-defaults"

    foreach xfCounter1 $xfClasses {
      foreach xfCounter2 $xfFileList {
        set xfPathName $xfCounter2
        if {[regsub -all "%N" "$xfPathName" "$xfCounter1" xfResult]} {
          set xfPathName $xfResult
        }
        if {[regsub -all "%T" "$xfPathName" "app-defaults" xfResult]} {
          set xfPathName $xfResult
        }
```

```
      if {[regsub -all "%S" "$xfPathName" "" xfResult]} {
        set xfPathName $xfResult
      }
      if {[regsub -all "%C" "$xfPathName" "" xfResult]} {
        set xfPathName $xfResult
      }
      if {[file exists $xfPathName] &&
          [file readable $xfPathName] &&
          ("[file type $xfPathName]" == "file" ||
           "[file type $xfPathName]" == "link")} {
        catch "option readfile $xfPathName $xfPriority"
        if {"[info commands XFParseAppDefs]" != ""} {
          XFParseAppDefs $xfPathName
        } {
          if {"[info commands XFLocalParseAppDefs]" != ""} {
            XFLocalParseAppDefs $xfPathName
          }
        }
      } {
        if {[file exists $xfCounter2/$xfCounter1] &&
            [file readable $xfCounter2/$xfCounter1] &&
            ("[file type $xfCounter2/$xfCounter1]" == "file" ||
             "[file type $xfCounter2/$xfCounter1]" == "link")} {
          catch "option readfile $xfCounter2/$xfCounter1 $xfPriority"
          if {"[info commands XFParseAppDefs]" != ""} {
            XFParseAppDefs $xfCounter2/$xfCounter1
          } {
            if {"[info commands XFLocalParseAppDefs]" != ""} {
              XFLocalParseAppDefs $xfCounter2/$xfCounter1
            }
          }
        }
      }
    }
  }
} {
  # load a specific application defaults file
  if {[file exists $xfAppDefFile] &&
      [file readable $xfAppDefFile] &&
      ("[file type $xfAppDefFile]" == "file" ||
       "[file type $xfAppDefFile]" == "link")} {
    catch "option readfile $xfAppDefFile $xfPriority"
    if {"[info commands XFParseAppDefs]" != ""} {
      XFParseAppDefs $xfAppDefFile
    } {
```

```
        if {"[info commands XFLocalParseAppDefs]" != ""} {
          XFLocalParseAppDefs $xfAppDefFile
        }
      }
    }
  }
}

# application setting procedure
proc XFLocalSetAppDefs {{xfWidgetPath "."}} {
  global xfAppDefaults

  if {![info exists xfAppDefaults]} {
    return
  }
  foreach xfCounter $xfAppDefaults {
    if {"$xfCounter" == ""} {
      break
    }
    set widname [lindex $xfCounter 0]
    if {[string match $widname ${xfWidgetPath}] ||
        [string match "${xfWidgetPath}*" $widname]} {
      set name [string tolower [lindex $xfCounter 1]]
      set value [lindex $xfCounter 2]
      # Now lets see how many tcl commands match the name
      # pattern specified.
      set widlist [info command $widname]
      if {"$widlist" != ""} {
        foreach widget $widlist {
          # make sure this command is a widget.
          if {![catch "winfo id $widget"] &&
              [string match "${xfWidgetPath}*" $widget]} {
            catch "$widget configure -$name $value"
          }
        }
      }
    }
  }
}

# initialize bindings for all widgets
proc XFInitAllBindings {} {
  # bindings
  bind all <Alt-Key> {
    tkTraverseToMenu %W %A
```

```
}
  bind all <Key-F10> {
    tkFirstMenu %W
}
  bind all <Key-Tab> {focus [tk_focusNext %W]}
  bind all <Shift-Key-Tab> {focus [tk_focusPrev %W]}
}

# prepare auto loading
global auto_path
global tk_library
global xfLoadPath
foreach xfElement [eval list [split $xfLoadPath :] $auto_path] {
  if {[file exists $xfElement/tclIndex]} {
    lappend auto_path $xfElement
  }
}

# initialize global variables
proc InitGlobals {} {
  global {NowShowing}
  set {NowShowing} {Side}
  global {ScalingFactor}
  set {ScalingFactor} {.125}
  global H0Scale
  set H0Scale [expr 1.0 / 87.0]
  global BridgeLengthInches
  set BridgeLengthInches [expr 12.0 * 21 * 20]
  global BridgeWidthInches
  set BridgeWidthInches [expr 12.0 * 40]
  global ViewWidths
  set ViewWidths(FloorU) [expr $BridgeLengthInches]
  set ViewWidths(Floor)  [expr $BridgeLengthInches]
  set ViewWidths(Side)   [expr $BridgeLengthInches]
  set ViewWidths(Top)    [expr $BridgeLengthInches]
  set ViewWidths(Details) [expr 12.0 * 20]
  global ViewHeights
  set ViewHeights(FloorU) [expr $BridgeWidthInches]
  set ViewHeights(Floor)  [expr $BridgeWidthInches]
  set ViewHeights(Side)   [expr 12.0 * 55]
  set ViewHeights(Top)    [expr $BridgeWidthInches]
  set ViewHeights(Details) [expr 12.0 * 20]
  global GirderHeights
  set GirderHeights {36 39 42 43 44 45 46 47 48 49
                     48 47 46 45 44 43 42 39 36}
```

```
global TopGirderLengths
set TopGirderLengths {}
global BarLengths
set BarLengths {}
global PrinterCommand
set PrinterCommand {lpr}
global PrintTo
set PrintTo {Printer}
global PrintFileName
set PrintFileName {Bridge.ps}
global {fsBox}
set {fsBox(activeBackground)} {}
set {fsBox(activeForeground)} {}
set {fsBox(all)} {0}
set {fsBox(background)} {}
set {fsBox(button)} {0}
set {fsBox(extensions)} {0}
set {fsBox(font)} {}
set {fsBox(foreground)} {}
set {fsBox(internalPath)} {}
set {fsBox(name)} {}
set {fsBox(path)} {.}
set {fsBox(pattern)} {*}
set {fsBox(scrollActiveForeground)} {}
set {fsBox(scrollBackground)} {}
set {fsBox(scrollForeground)} {}
set {fsBox(scrollSide)} {left}
set {fsBox(showPixmap)} {0}
global {yesNoBox}
set {yesNoBox(activeBackground)} {}
set {yesNoBox(activeForeground)} {}
set {yesNoBox(afterNo)} {0}
set {yesNoBox(afterYes)} {0}
set {yesNoBox(anchor)} {n}
set {yesNoBox(background)} {}
set {yesNoBox(button)} {1}
set {yesNoBox(font)} {*times-bold-r-normal*24*}
set {yesNoBox(foreground)} {}
set {yesNoBox(justify)} {center}

# please don't modify the following
# variables. They are needed by xf.
global {autoLoadList}
set {autoLoadList(Bridge.tcl)} {0}
global {internalAliasList}
```

```
  set {internalAliasList} {}
  global {moduleList}
  set {moduleList(Bridge.tcl)} {}
  global {preloadList}
  set {preloadList(xfInternal)} {}
  global {symbolicName}
  set {symbolicName(BridgeDisplay)} {.frame2.canvas2}
  set {symbolicName(root)} {.}
  global {xfWmSetPosition}
  set {xfWmSetPosition} {}
  global {xfWmSetSize}
  set {xfWmSetSize} {}
  global {xfAppDefToplevels}
  set {xfAppDefToplevels} {}
}

# Procedure: EndSrc
proc EndSrc {} {
  DoTopGirderLengths
  DoBarLengths
  ReShow
}


# initialize global variables
InitGlobals

# display/remove toplevel windows.
ShowWindow.

# load default bindings.
if {[info exists env(XF_BIND_FILE)] &&
    "[info procs XFShowHelp]" == ""} {
  source $env(XF_BIND_FILE)
}

# initialize bindings for all widgets.
XFInitAllBindings


# parse and apply application defaults.
XFLocalLoadAppDefs Bridge
XFLocalSetAppDefs

EndSrc
```

```
# eof
#
```