ICOT Technical Report: TM-0066

TM-0066

作図支援プログラム
使用説明書

辻　順一郎

July, 1984

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03) 456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

作図支援プログラム
使用説明書

第3研究室

辻　順一郎

LISP Machine 上で　グラフィック機能，画面のハードコピー機能を利用した　簡易作図支援プログラムの使い方を記す。

本プログラムは　論文用図面，OHPシート等の作成を支援するツール・プログラムであり，メニュー，マウスを多用して簡単に図面を作成することができる。また作成した図面はデータ・ファイルとしてディスク上にセーブし　後日　編集しなおすことも可能である。

なお，本プログラムの仕様は　断りなく変更することがありますので　御了承下さい。

S.59.4.26

3研 J.Tsuji

LISP Machine作図支援プログラム

1．起　動

i) select CNTL-L かSystem Menu で新しいLisp Listener を作る
(Lisp Listener 1 は使用しない方がbetter)

ii) (load '>tsuji>ohp)を入力する。

注）1度ohp をロードしたあとは（ohp)のみで可。

2．メニュー項目

（1）CREATE　LINE

直線オブジェクトの作成。マウスによって始点、終点を指定する。
（マウスカーソル“X”の中心が指示点）

（2）CREATE　VERTICAL　LINE

垂直な直線オブジェクトの作成。マウスによって始点のｘ，ｙ座標、終点のｙ座標を指定する。

（3）CREATE　HORIZONTAL　LINE

水平な直線オブジェクトの作成。マウスによって始点のｘ，ｙ座標、終点のｘ座標を指定する。

（4）DELETE　LINE

直線オブジェクトの消去。マウスカーソルにより直線を指定する。

（5）CREATE　DASHED　LINE

破線オブジェクトの作成。マウスにより始点・終点を指定する。

（6）CREATE　VER.　DASHED　LINE

垂直な破線オブジェクトの作成。マウスにより始点のｘ，ｙ座標及び終点のｙ座標を指定する。

（7）CREATE HOR. DASHED LINE

水平な破線オブジェクトの作成。マウスにより始点のｘ，ｙ座標及び終点のｘ座標を指定する。

（8）DELETE DASHED LINE

破線オブジェクトの消去。マウスにより破線を指定する。

（9）CREATE BOX

矩形オブジェクトの作成。マウスにより矩形の左上端、下端を指定する。

（10）MOVE BOX

矩形オブジェクトの移動。マウスにより移動するオブジェクトを指定した後その新しい位置を指定する。

（11）DELETE BOX

矩形オブジェクトの消去。マウスにより消去する矩形オブジェクトを指定する。

（12）COPY BOX

すでに作成してある矩形オブジェクトと同じサイズを持つ矩形オブジェクトを作成する。マウスによりコピーする矩形オブジェクトを指定し、その後新しい矩形オブジェクトの位置を指定する。

（13）SET FILL FLAG

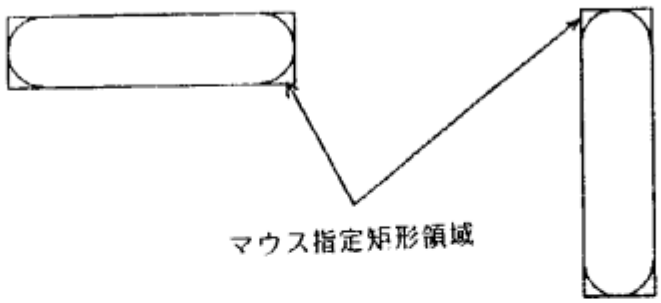
矩形オブジェクトのFILL FLAG をセットする。FILL FLAG がセットされているとその矩形オブジェクトの描画時にその内部をクリヤする。

マウスによって矩形オブジェクトを指定する。＋印のマウスカーソルによって矩形オブジェクトを指定したのち、マウスマーカがその矩形の枠に変化するので希望の矩形の場合左又は右のボタンをクリックする。希望の矩形オブジェクトと異なる場合、真中のボタンをクリックすると、他の矩形オブジェクトを示すので、希望の矩形になるまでくり返す。

（14）CREATE CIRCULAR BOX

角の丸まったボックス・オブジェクトの作成。マウスによって外接する矩形を指定する。

注）マウスにより指定された矩形領域が、横長の場合左右の辺がそれぞれ半円化され、縦長の場合、上下の辺がそれぞれ半円化される。（下図参照）



（15）MOVE CIRCULAR BOX

CIRCULAR-BOXの移動。マウスにより移動位置を指定する。

（16）DELETE CIRCULAR BOX

CIRCULAR-BOXの消去。マウスにより消去するCIRCULAR-BOXを指定する。

（17）COPY CIRCULAR BOX

CIRCULAR-BOXのコピー。マウスにより指定したCIRCULAR-BOXと同じサイズのCIRCULAR-BOXを作成し指定位置に移動する。

（18）CREATE STRING

文字列オブジェクトの作成。文字列入力用ウィンドウが表示され、キーボードから文字列を入力する。文字列入力後マウスで表示位置を指定する。

（19）MOVE STRING

文字列オブジェクトの移動。文字列オブジェクトの表示位置をマウスで指定する。

（20）DELETE STRING

文字列オブジェクトの消去。マウスによって消去する文字列を指定する。

（21）COPY STRING

文字列オブジェクトのコピー。すでに作成された文字列オブジェクトと同じ文字列、フォントを持つ文字列オブジェクトを作成する。マウスによりコピーする文字列オブジェクトと新しいオブジェクトの表示位置を指定する。

（22）SET FONT

文字列オイブジェクトのフォントの変更。マウスによりフォントを変更する文字列オブジェクトを指定し、その後表示されるフォントのリストの中からマウスでフォントを指定する。

（23）SET DEFAULT FONT

それ以降作成する文字列オブジェクトのデフォールト・フォントを指定する。

注）フォントはロード済フォント、">SYS>fonts>"中のフォント、">Tsuji>fonts>"中のフォントより選択が可能。

（24）HARDCOPY

現在表示されているシートのハードコピーを取る。

注）HARDCOPYの処理は別プロセス化してあるのですぐに次の作業（次のシートのHARDCOPYを含む）を行なってかまわない。

（25）REDISPLAY

現在表示されているシートの全オブジェクトを再表示する。

（26）EXIT

本プログラムの処理を終了し（起動ずみのHARDCOPY処理は継続）、Lisp Listenerに戻る。

（27）SAVE

現在表示されているシートのみをファイルにセーブする。セーブするファイル名をファイル名入力用ウィンドウに入力する。

（28）LOAD

現在表示されているシートの全オブジェクトをクリヤし、ファイルからオブジェクト群をロードする。

（29）SAVE MULTIPLE SHEETS

現在ロードされている（作成された）シート群をファイルにセーブする。

(30) LOAD MULTIPE SHEETS

ファイルからシート群をロードする。

注) 28, 29と30, 31ではセーブ、ロードの形式が異なるので注意して下さい。

(31) NEXT SHEET

現在表示中のシートの編集を終了し、次のシートに移る。

(32) PREVIOUS SHEET

1つ前のシートに戻る。

(33) CLEAR

現在表示中のシートの全オブジェクトを消去し、画面をクリヤする。

(34) CLEAR ALL SHEETS

全シート中のオブジェクトを消去する。

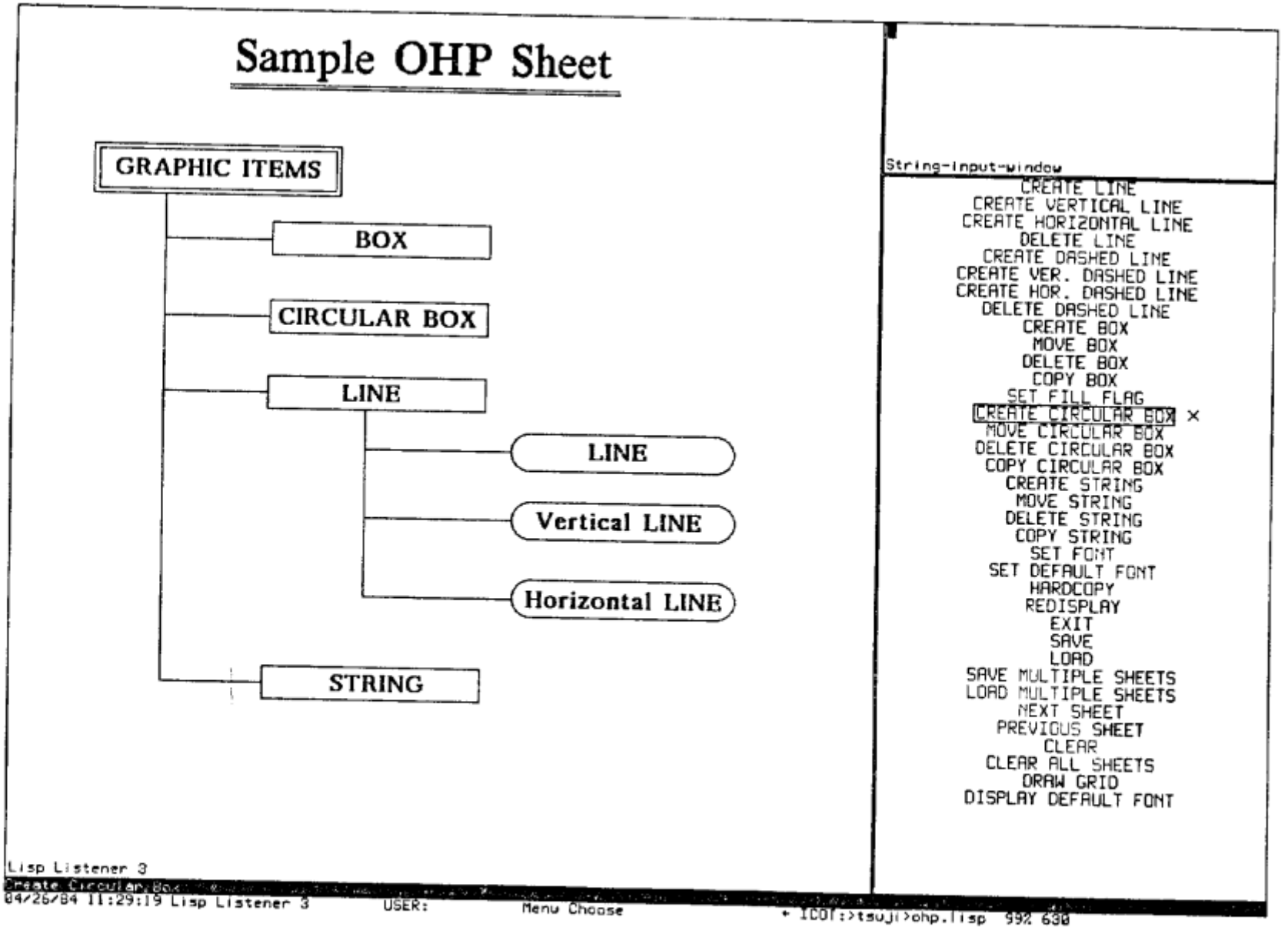
(35) DRAW GRID

各オブジェクトの位置ぎめのために格子を表示する。
再度本項を選択することにより格子は消去される。

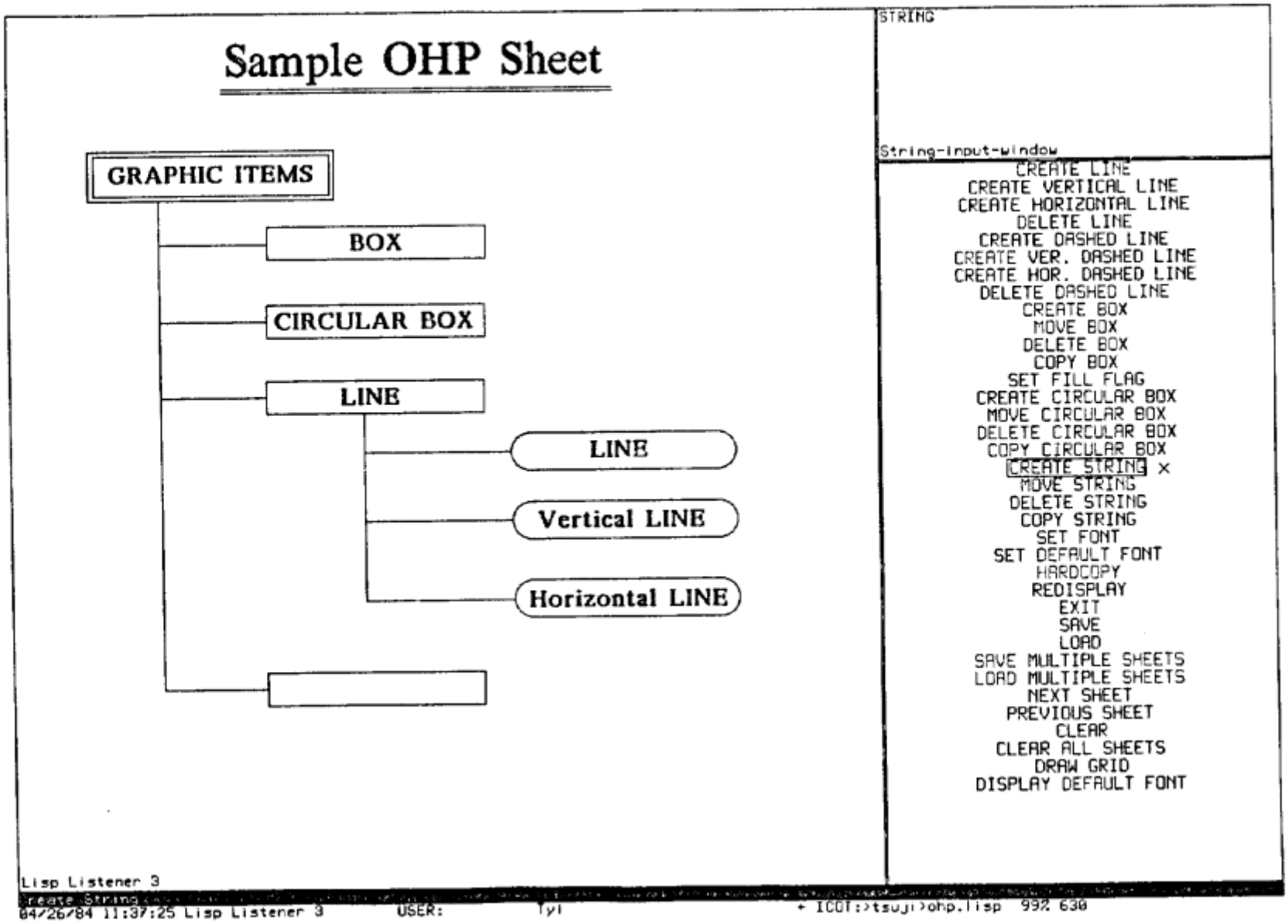
注) 格子はHARDCOPYには影響しない。
格子の間隔はHARDCOPY作成時に1cm間隔となるように設定されている。

(36) DISPLAY DEFAULT FONT

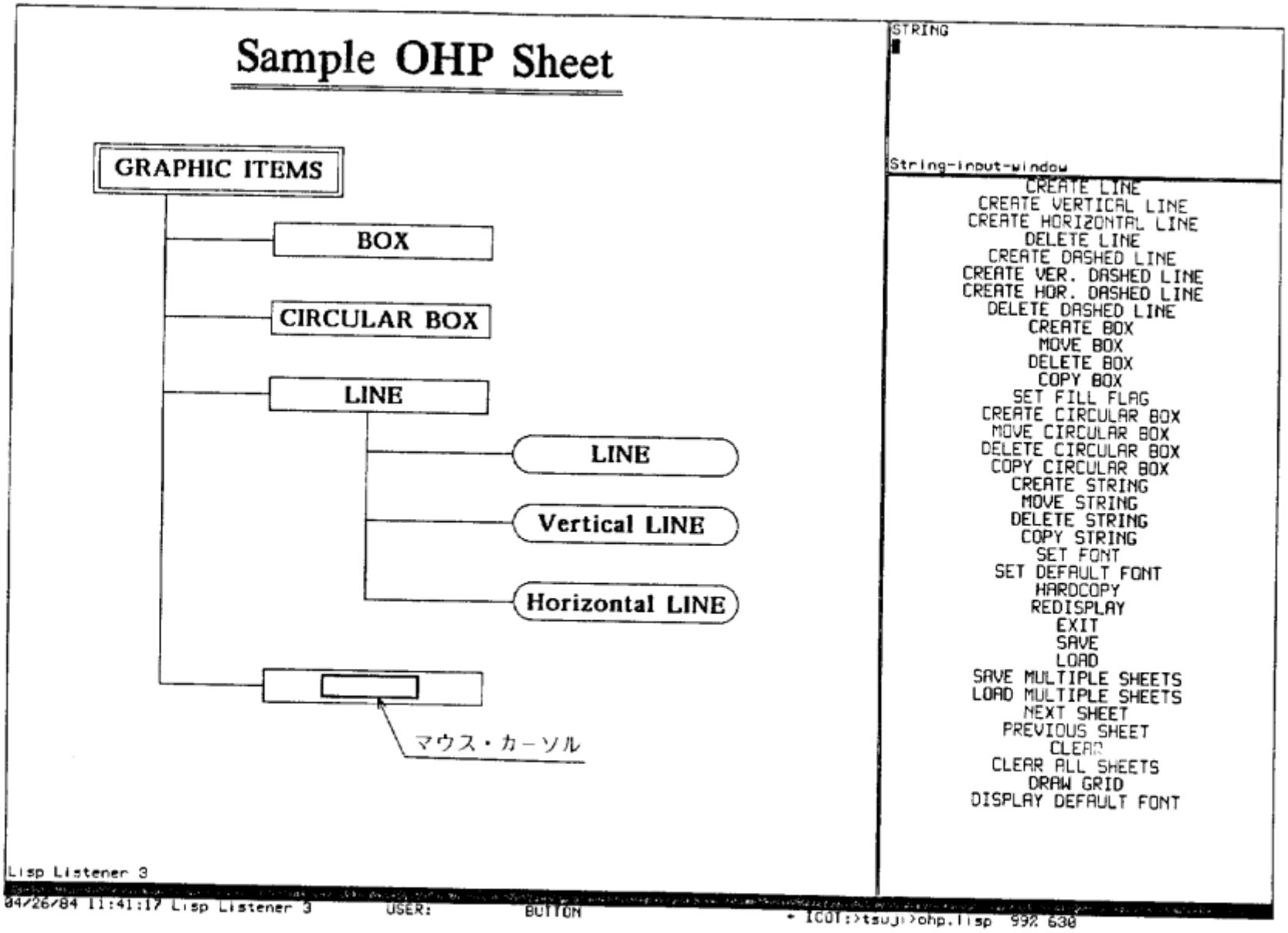
現在のデフォールト・フォントの全文字のイメージを表示する。



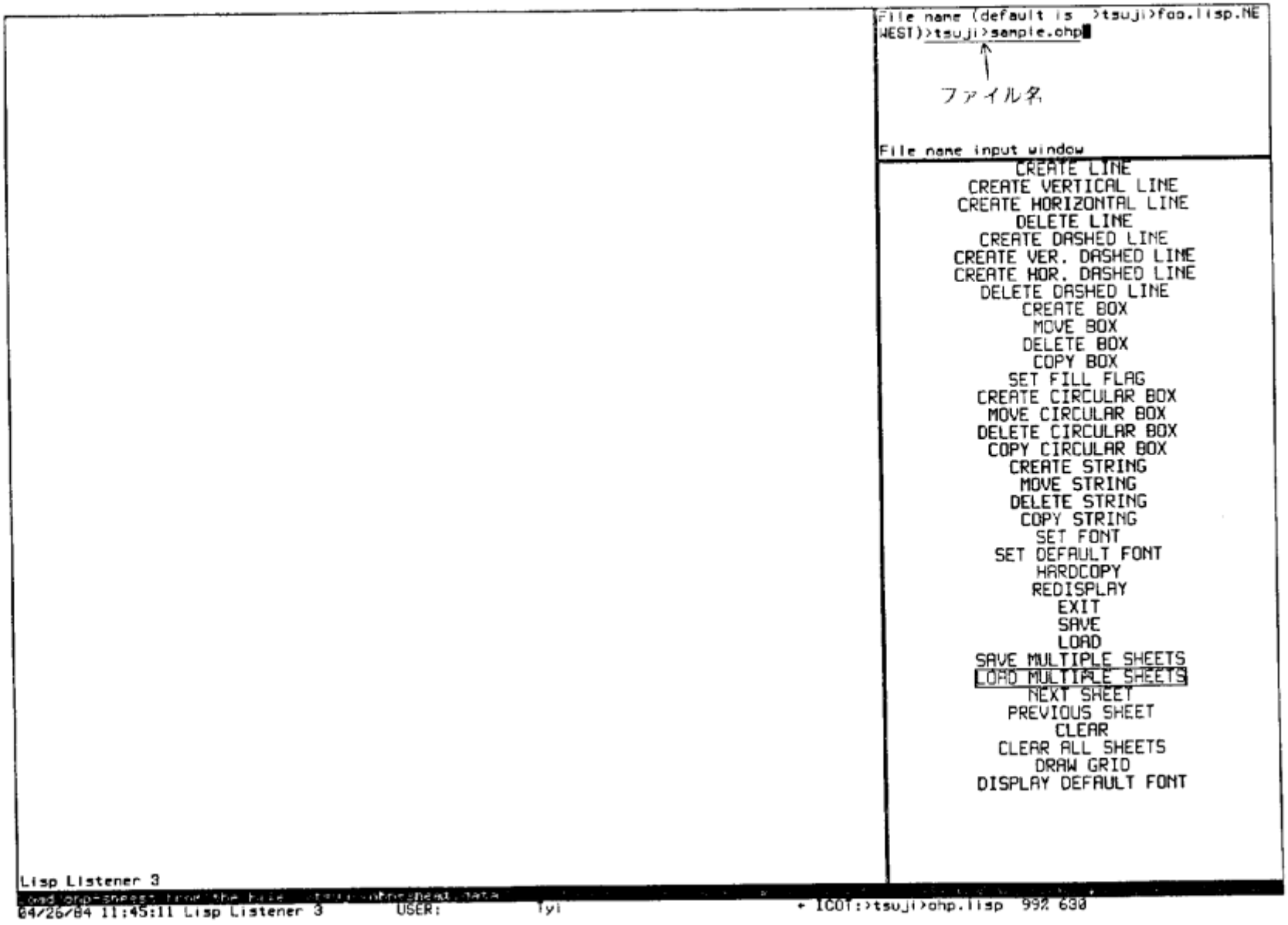
図－1　作図支援プログラム画面イメージ



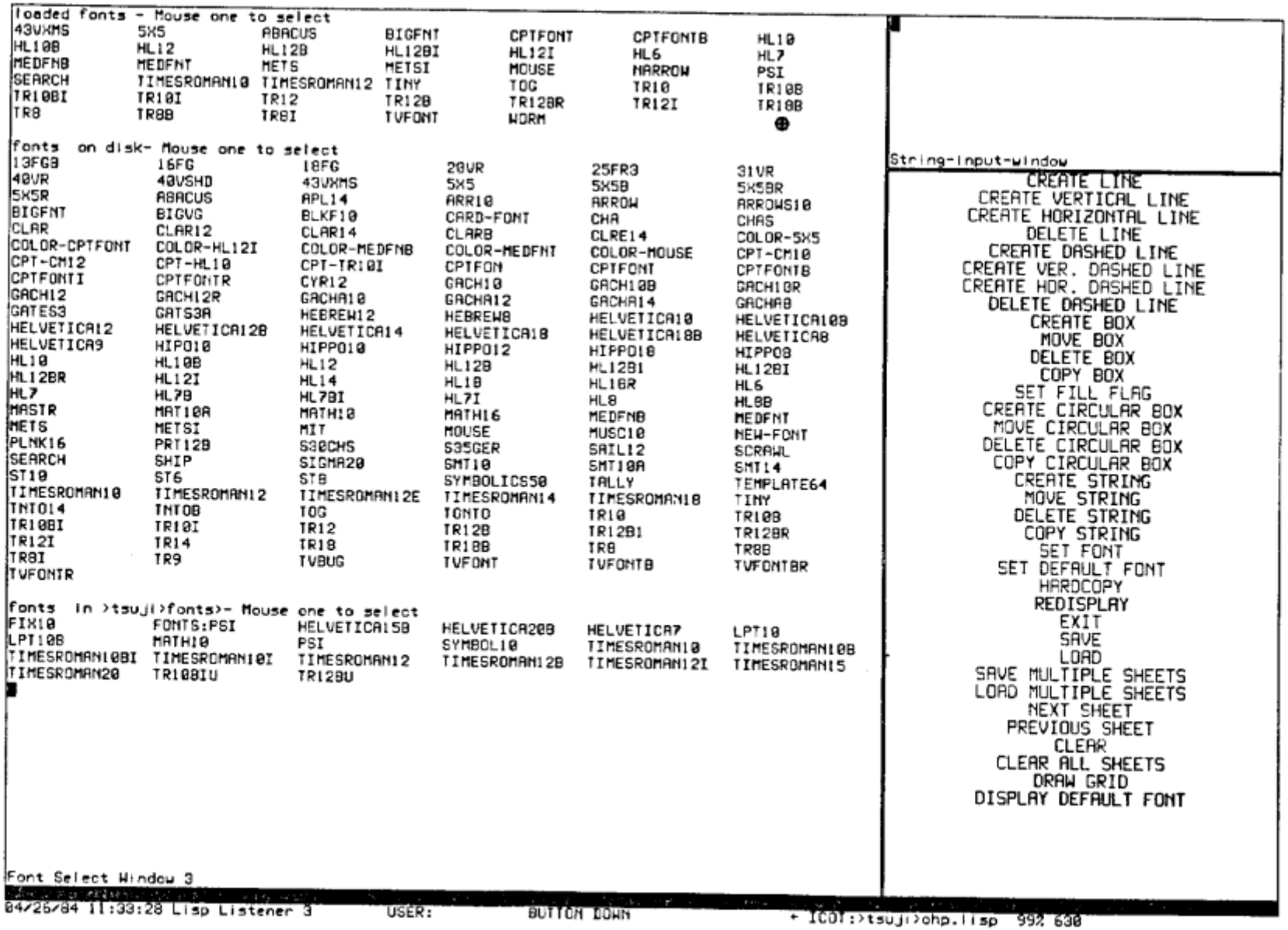
図－２　文字列オブジェクト（"ＳＴＲＩＮＧ"）の入力



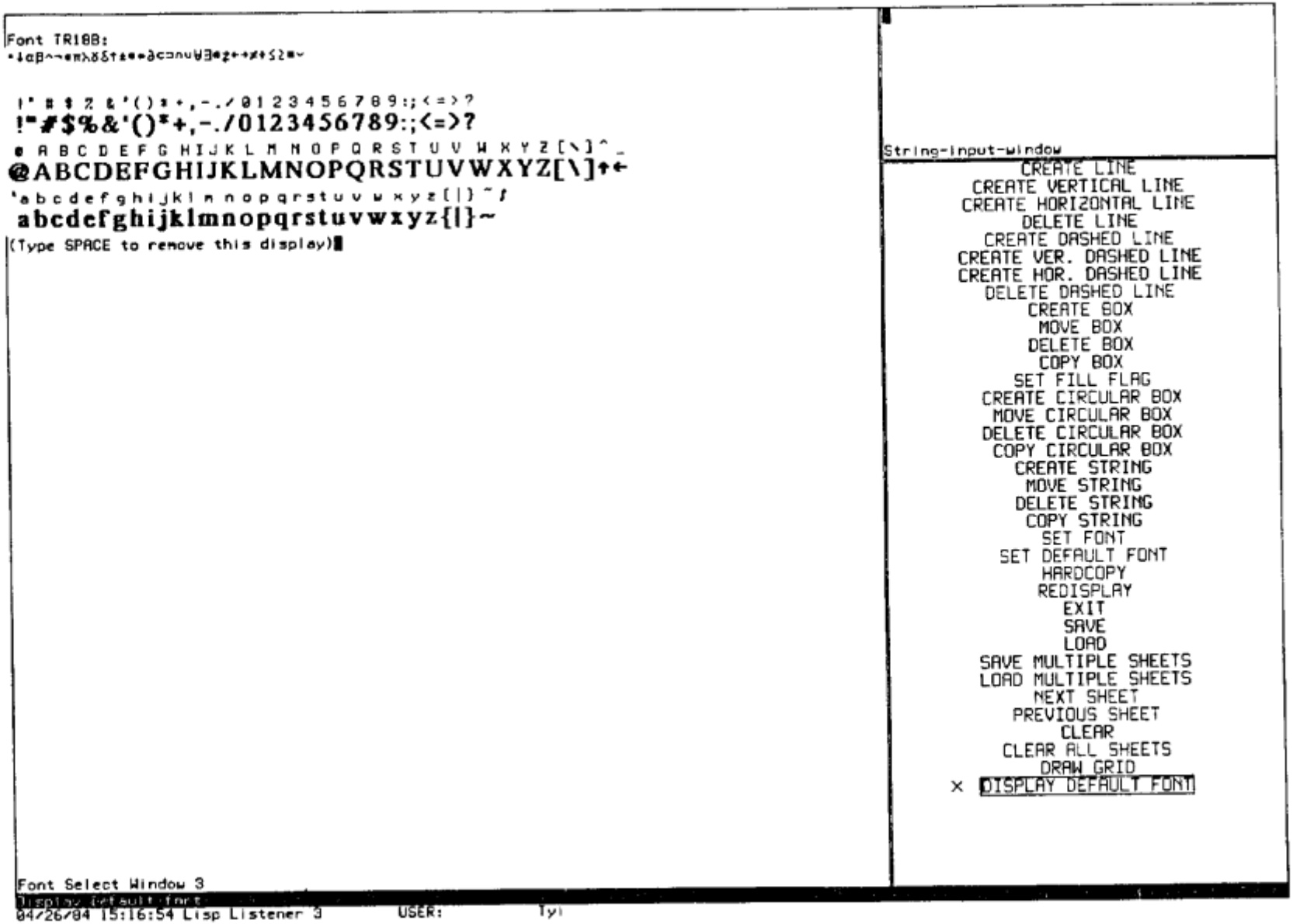
図－３　文字列オブジェクト（"ＳＴＲＩＮＧ"）の位置決め


File name (default is >tsuji>foo.lisp.NE
WEST)>tsuji>sample.ohp
ファイル名
File name input window
CREATE LINE
CREATE VERTICAL LINE
CREATE HORIZONTAL LINE
DELETE LINE
CREATE DASHED LINE
CREATE VER. DASHED LINE
CREATE HOR. DASHED LINE
DELETE DASHED LINE
CREATE BOX
MOVE BOX
DELETE BOX
COPY BOX
SET FILL FLAG
CREATE CIRCULAR BOX
MOVE CIRCULAR BOX
DELETE CIRCULAR BOX
COPY CIRCULAR BOX
CREATE STRING
MOVE STRING
DELETE STRING
COPY STRING
SET FONT
SET DEFAULT FONT
HARDCOPY
REDISPLAY
EXIT
SAVE
LOAD
SAVE MULTIPLE SHEETS
LOAD MULTIPLE SHEETS
NEXT SHEET
PREVIOUS SHEET
CLEAR
CLEAR ALL SHEETS
DRAW GRID
DISPLAY DEFAULT FONT
Lisp Listener 3
04/26/84 11:45:11 Lisp Listener 3 USER: Tyi
ICOT:>tsuji>ohp.lisp 99% 630

図－4　シート群のロード


loaded fonts - Mouse one to select
43VXMS 5X5 ABACUS BIGFNT CPTFONT CPTFONTB HL10
HL10B HL12 HL12B HL12BI HL12I HL6 HL7
MEDFNB MEDFNT METS METSI MOUSE NARROW PSI
SEARCH TIMESROMAN10 TIMESROMAN12 TINY TOG TR10 TR10B
TR10BI TR10I TR12 TR12B TR12BR TR12I TR18B
TR8 TR8B TR8I TVFONT WORM
fonts on disk- Mouse one to select
13FGB 16FG 18FG 20VR 25FR3 31VR
40VR 40VSHD 43VXMS 5X5 5X5B 5X5BR
5X5R ABACUS APL14 ARR10 ARROW ARROWS10
BIGFNT BIGVG BLKF10 CARD-FONT CHA CHAS
CLAR CLAR12 CLAR14 CLARB CLRE14 COLOR-5X5
COLOR-CPTFONT COLOR-HL12I COLOR-MEDFNB COLOR-MEDFNT COLOR-MOUSE CPT-CM10
CPT-CM12 CPT-HL10 CPT-TR10I CPTFON CPTFONT CPTFONTB
CPTFONTI CPTFONTR CYR12 GACH10 GACH10B GACH10R
GACH12 GACH12R GACHA10 GACHA12 GACHA14 GACHAB
GATES3 GATS3A HEBREW12 HEBREW8 HELVETICA10 HELVETICA10B
HELVETICA12 HELVETICA12B HELVETICA14 HELVETICA18 HELVETICA18B HELVETICA8
HELVETICA9 HIPO10 HIPPO10 HIPPO12 HIPPO18 HIPPO8
HL10 HL10B HL12 HL12B HL12BI HL12BI
HL12BR HL12I HL14 HL18 HL18R HL6
HL7 HL7B HL7BI HL7I HL8 HL8B
MASTR MAT10A MATHI0 MATHI6 MEDFNB MEDFNT
METS METSI MIT MOUSE MUSC10 NEW-FONT
PLNK16 PRT12B S30CHS S35GER SAIL12 SCRAWL
SEARCH SHIP SIGMA20 SMT10 SMT10A SMT14
ST10 ST6 ST8 SYMBOLICS50 TALLY TEMPLATE64
TIMESROMAN10 TIMESROMAN12 TIMESROMAN12E TIMESROMAN14 TIMESROMAN18 TINY
TNIO14 TNIOB TOG TONTO TR10 TR10B
TR10BI TR10I TR12 TR12B TR12BI TR12BR
TR12I TR14 TR18 TR18B TR8 TR8B
TR8I TR9 TVBUG TVFONT TVFONTB TVFONTBR
TVFONTR
fonts in >tsuji>fonts>- Mouse one to select
FIX10 FONTS:PSI HELVETICA15B HELVETICA20B HELVETICA7 LPT10
LPT10B MATHI0 PSI SYMBOL10 TIMESROMAN10 TIMESROMAN10B
TIMESROMAN10BI TIMESROMAN10I TIMESROMAN12 TIMESROMAN12B TIMESROMAN12I TIMESROMAN15
TIMESROMAN20 TR10BIU TR12BU
Font Select Window 3
String-Input-Window
CREATE LINE
CREATE VERTICAL LINE
CREATE HORIZONTAL LINE
DELETE LINE
CREATE DASHED LINE
CREATE VER. DASHED LINE
CREATE HOR. DASHED LINE
DELETE DASHED LINE
CREATE BOX
MOVE BOX
DELETE BOX
COPY BOX
SET FILL FLAG
CREATE CIRCULAR BOX
MOVE CIRCULAR BOX
DELETE CIRCULAR BOX
COPY CIRCULAR BOX
CREATE STRING
MOVE STRING
DELETE STRING
COPY STRING
SET FONT
SET DEFAULT FONT
HARDCOPY
REDISPLAY
EXIT
SAVE
LOAD
SAVE MULTIPLE SHEETS
LOAD MULTIPLE SHEETS
NEXT SHEET
PREVIOUS SHEET
CLEAR
CLEAR ALL SHEETS
DRAW GRID
DISPLAY DEFAULT FONT
04/26/84 11:33:28 Lisp Listener 3
USER:
BUTTON DOWN
ICOT:>tsuji>ohp.lisp 99% 630

図－5　フォントの指定



図－6　デフォールト・フォントの表示

PSI

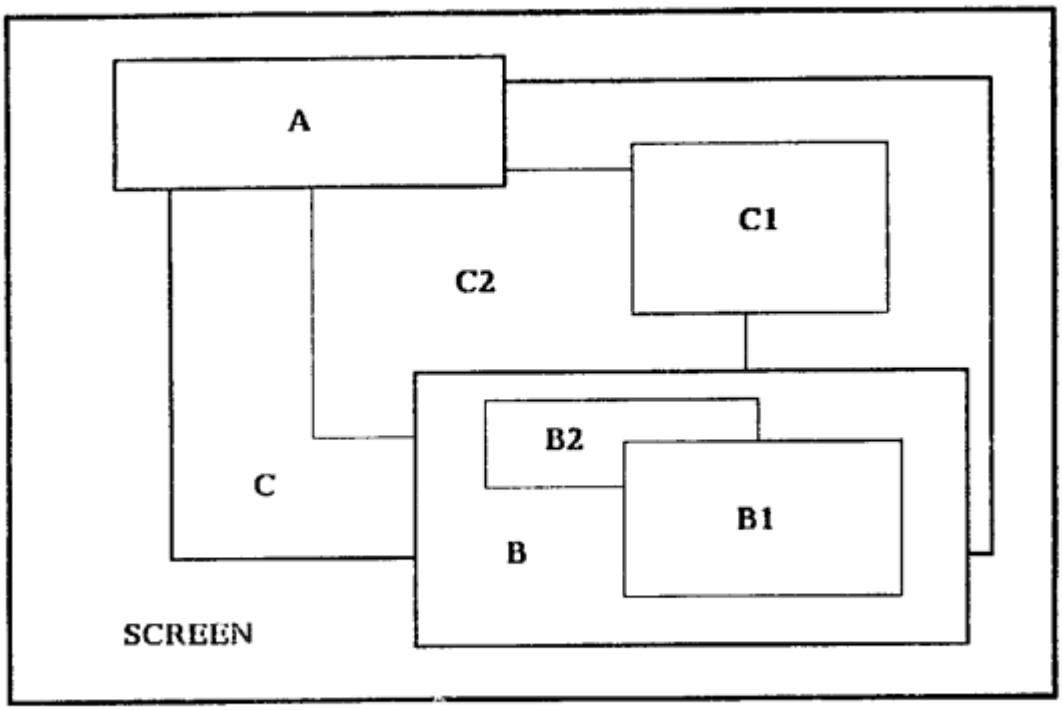
(Personal Sequential Inference machine)

Designed by M. Yokota
Font Created by J. Tsuji

図－7　作成シート出力例（１）

```
;; -*- Patch-File: Yes -*-
;;;;;;;;;;;;;;;;;;;;;; VARIABLES ;;;;;;;;;;;;;;;;;;;;;;

(DEFVAR GRAPHIC-ITEM-LIST
  '(
    ("CREATE LINE"          :VALUE C-LINE
      :DOCUMENTATION "Create Line")
    ("CREATE VERTICAL LINE"  :VALUE V-LINE
      :DOCUMENTATION "Create Vertical Line")
    ("CREATE HORIZONTAL LINE" :VALUE H-LINE
      :DOCUMENTATION "Create Horizontal Line")
    ("DELETE LINE"          :VALUE D-LINE
      :DOCUMENTATION "Delete Line")
    ("CREATE DASHED LINE"    :VALUE C-DASHED-LINE
      :DOCUMENTATION "Create Dashed Line")
    ("CREATE VER. DASHED LINE" :VALUE V-DASHED-LINE
      :DOCUMENTATION "Create Vertical Dashed Line")
    ("CREATE HOR. DASHED LINE" :VALUE H-DASHED-LINE
      :DOCUMENTATION "Create Horizontal Dashed Line")
    ("DELETE DASHED LINE"    :VALUE D-DASHED-LINE
      :DOCUMENTATION "Delete DAshed Line")
    ("CREATE BOX"           :VALUE CREATE-BOX
      :DOCUMENTAION  "Create Box")
    ("MOVE BOX"             :VALUE M-BOX
      :DOCUMENTATION "Move Box")
    ("DELETE BOX"           :VALUE D-BOX
      :DOCUMENTATION "Delete Box")
    ("COPY BOX"             :VALUE GB-COPY
      :DOCUMENTATION "Copy Bix")
    ("SET FILL FLAG"        :VALUE SET-FILL-FLAG
      :DOCUMENTATION "Set Fill Flag of the Box to clear the inside of the box")
    ("CREATE CIRCULAR BOX"   :VALUE create-c-box
      :DOCUMENTATION "Create Circular Box")
    ("MOVE CIRCULAR BOX"     :VALUE MOVE-C-BOX
      :DOCUMENTATION "Move Circular Box")
    ("DELETE CIRCULAR BOX"   :VALUE DELETE-C-BOX
      :DOCUMENTATION "Delete Circular Box")
    ("COPY CIRCULAR BOX"     :VALUE COPY-C-BOX
      :DOCUMENTATION "Copy Circular Box")
    ("CREATE STRING"        :VALUE C-STRING
      :DOCUMENTATION "Create String")
    ("MOVE STRING"          :VALUE M-STRING
      :DOCUMENTATION "Move String")
    ("DELETE STRING"        :VALUE D-STRING
      :DOCUMENTATION  "Delete String")
    ("COPY STRING"          :VALUE COPY-STRING
      :DOCUMENTATION "Copy The String")
    ("SET FONT"             :VALUE SET-FONT
      :DOCUMENTATION  "Set the Font of String")
    ("SET DEFAULT FONT"      :VALUE SET-DEFAULT-FONT
      :DOCUMENTATION "Change the default font")
    ("HARDCOPY"             :VALUE HC
      :DOCUMENTATION "Hardcopy the display image")
    ("REDISPLAY"            :VALUE REDISPLAY
      :DOCUMENTATION "Refresh Window and Redisplay all objects")
    ("EXIT"                 :VALUE GB-QUIT
      :DOCUMENTATION "Exit from the graphic tool")
    ("SAVE"                 :VALUE SAVE-OHP
      :DOCUMENTATION "Save the ohp-sheet to the File")
    ("LOAD"                 :VALUE LOAD-OHP
      :DOCUMENTATION "Load ohp-sheet from the File")
    ("SAVE MULTIPLE SHEETS"  :VALUE SAVE-OHP-SHEETS
      :DOCUMENTATION "Save the ohp-sheets to the File")
    ("LOAD MULTIPLE SHEETS"  :VALUE LOAD-OHP-SHEETS
      :DOCUMENTATION "Load ohp-sheest from the File")
    ("NEXT SHEET"           :VALUE NEXT
      :DOCUNENTATION "Edit Next Sheet")
    ("PREVIOUS SHEET"        :VALUE PREVIOUS
      :DOCUMENTATION "Edit Previous Sheet")
    ("CLEAR"                :VALUE G-CLEAR
      :DOCUMENTATION "Clear Window")
    ("CLEAR ALL SHEETS"      :VALUE ALL-CLEAR
      :DOCUMENTATION "Clear all sheets")
    ("DRAW GRID"            :VALUE DRAW-GRID
      :DOCUMENTATION "Draw Grid (1cm) on the window --- when hardcopy the grid is ignored")
    ("DISPLAY DEFAULT FONT"  :VALUE DISPLAY-DEFAULT-FONT
      :DOCUMENTATION "Display Default font")
    ))
(DEFVAR W)
(DEFVAR X)
(DEFVAR Y)
```

```
(defvar hc-window
  (tv:make-window 'tv:window-without-label
                  ':left 0.
                  ':top 0.
                  ':right 749.
                  ':borders 0.
                  ':bottom 749.
                  ':save-bits t
                  ':font-map '(fonts:tr18b)
                  ':blinker-p nil))

(defun ohp ()
  (send terminal-io ':set-edges 0 0 749. 749.)
  (send terminal-io ':set-font-map '(fonts:tr18b))
  (edit-box))

(load '>tsuji>ohp-system.bin)

(ohp)
```



A: selected

B: shown

B1: shown

B2: overlapped

C: overlapped

C1: exposed

C2: overlapped

図－8　作成シート出力例（２）

```
(DEFVAR XX)
(DEFVAR YY)
(DEFVAR BOX-EDGE-ALIST ())
(DEFVAR LINE-EDGE-ALIST ())
(DEFVAR DASHED-LINE-EDGE-ALIST ())
(DEFVAR STRING-EDGE-ALIST ())
(DEFVAR C-BOX-EDGE-ALIST ())
(DEFVAR BOX)
(DEFVAR LINE)
(DEFVAR DASHED-LINE)
(DEFVAR STRING)
(DEFVAR C-BOX)
(DEFVAR BOXES)
(DEFVAR LINES)
(DEFVAR DASHED-LINES)
(DEFVAR STRINGS)
(DEFVAR C-BOXES)
(DEFVAR OBJ)
(DEFVAR S-WINDOW)
(DEFVAR BOX-SAVE-LIST)
(DEFVAR C-BOX-SAVE-LIST)
(DEFVAR LINE-SAVE-LIST)
(DEFVAR DASHED-LINE-SAVE-LIST)
(DEFVAR STRING-SAVE-LIST)
(DEFVAR LOAD-LIST)
(DEFVAR FILE-WINDOW)
(DEFVAR SHEETS1)
(DEFVAR SHEETS2)
(DEFVAR F-WINDOW)
(DEFVAR FONT)
(DEFVAR DEFAULT-FONT)
(DEFVAR MOUSE-X)
(DEFVAR MOUSE-Y)
(DEFVAR S-I WINDOW)
(DEFVAR GRAPHIC-BOX-MENU)
(DEFVAR CHR)
(DEFVAR CHRS ())
(DEFVAR S-I-WINDOW)
(DEFVAR FONT-SELECT-WINDOW)
(DEFVAR PAI 3.141592653589)
(DEFVAR GRID-FLAG 0)
(DEFVAR FONT-NAME)

;;;;;;;;;;;;;;;;;;;; MISCELANIOUS ;;;;;;;;;;;;;;;;;;;;

(DEFUN G-CLEAR (W &REST IGNORE)
  (SEND W ':REFRESH)
  (SETQ BOX-EDGE-ALIST () STRING-EDGE-ALIST () LINE-EDGE-ALIST ()
        DASHED-LINE EDGE ALIST () C-BOX-EDGE-ALIST ())
  (SEND BOXES ':SET-OBJECTS-LIST ())
  (SEND STRINGS ':SET-OBJECTS-LIST ())
  (SEND LINES ':SET-OBJECTS-LIST ())
  (SEND DASHED-LINES ':SET-OBJECTS-LIST ())
  (SEND C-BOXES ':SET-OBJECTS-LIST ()))


(DEFUN ALL-CLEAR (W &REST IGNORE)
  (G-CLEAR W)
  (SETQ SHEETS1 () SHEETS2 ()))

(DEFUN GB-QUIT (W &REST IGNORE)
  (*THROW 'EXIT-BOX-EDITOR T))

(DEFUN REDISPLAY (W)
  (SEND W ':REFRESH)
  (SEND BOXES ':SHOW-OBJECTS)
  (SEND C-BOXES ':SHOW-OBJECTS)
  (SEND STRINGS ':SHOW OBJECTS)
  (SEND LINES ':SHOW-OBJECTS)
  (SEND DASHED-LINES ':SHOW-OBJECTS)
  (IF (EQ GRID-FLAG 1)
      (DRAW-GRID-INTERNAL W)))

(DEFUN DRAW-GRID (&OPTIONAL (W TERMINAL-IO))
  (SETQ GRID-FLAG (LOGXOR GRID-FLAG 1))
  (DRAW-GRID-INTERNAL W))

(DEFUN DRAW-GRID-INTERNAL (&OPTIONAL (W TERMINAL-IO))
  (DO I 47.4 (+ 47.4 I) (> I 749.)
    (SEND W ':DRAW-LINE (FIXR I) 0 (FIXR I) 749. TV:ALU-XOR))
  (DO I 47.4 (+ 47.4 I) (> I 749.)
```

```
  (SEND W ':DRAW-LINE 0 (FIXR I) 1088. (FIXR I) TV:ALU-XOR)))

(DEFUN DISPLAY-DEFAULT-FONT (W)
 (SEND FONT-SELECT-WINDOW ':SELECT)
 (SEND FONT-SELECT-WINDOW ':REFRESH)
 (FED:COM-DISPLAY-FONT DEFAULT-FONT FONT-SELECT-WINDOW)
 (TYI FONT SELECT-WINDOW)
 (SEND FONT-SELECT-WINDOW ':BURY))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; WINDOW RESOURCES ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(TV:DEFWINDOW-RESOURCE GRAPHIC-BOX-MENU () ;
 :MAKE-WINDOW (TV:MENU :ITEM-LIST GRAPHIC-ITEM-LIST
                       :SAVE-BITS T
                       :TOP 131.
                       :LEFT 750.
                       :BOTTOM 749.
                       :RIGHT 1088.)
 :REUSABLE-WHEN :DEEXPOSED)

(DEFFLAVOR FONT-SELECT-WINDOW () (TV:BASIC-MOUSE-SENSITIVE-ITEMS TV:WINDOW))

(SETQ FONT SELECT-WINDOW (TV:MAKE-WINDOW 'FONT-SELECT-WINDOW
                  ':TOP 0.
                  ':LEFT 0.
                  ':RIGHT 749.
                  ':BOTTOM 749.))

(SETQ FILE-WINDOW (TV:MAKE-WINDOW 'TV:WINDOW
                   ':LABEL (:STRING "FILE NAME INPUT WINDOW")
                   ':TOP 0.
                   ':LEFT 750.
                   ':BOTTOM 130.
                   ':RIGHT 1088.))

(DEFUN SELECT-FILE-WINDOW ()
    (SEND FILE WINDOW ':SELECT)
    (SEND FILE-WINDOW ':REFRESH)
    (SEND FILE-WINDOW ':STRING-OUT "File name (default is ")
    (SEND FILE WINDOW ':STRING-OUT
        (SEND (FS:DEFAULT-PATHNAME) ':STRING FOR-MINI))
    (SEND FILE-WINDOW ':STRING-OUT ")"))

(SETQ S-I-WINDOW (TV:MAKE-WINDOW 'TV:WINDOW
                   ':LABEL (:STRING "String-input-window")
                   ':TOP 0.
                   ':LEFT 750.
                   ':RIGHT 1088.
                   ':BOTTOM 130.))

(SETQ S-WINDOW (TV:MAKE-WINDOW 'TV:WINDOW
                   ':TOP 100.
                   ':LEFT 0.
                   ':BOTTOM 300.
                   ':RIGHT 1000.
                   ':FONT-MAP '(FONTS:TR18B)
                   ':DEEXPOSED-TYPEOUT-ACTION ':PERMIT
                   ':SAVE-BITS T))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; MAIN FUNCTION ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(DEFUN EDIT-BOX (&OPTIONAL (WINDOW TERMINAL-IO))
 (SETQ DEFAULT-FONT FONTS:TR18B)
 (SETQ MOUSE-X 830. MOUSE-Y 350.)
 (SETQ SHEETS1 () SHEETS2 ())
 (SETQ BOX-EDGE-ALIST ())
 (SEND BOXES ':SET-OBJECTS-LIST ())
 (SETQ C-BOX-EDGE-ALIST ())
 (SEND C-BOXES ':SET-OBJECTS-LIST ())
 (SETQ STRING-EDGE-ALIST ())
 (SEND STRINGS ':SET-OBJECTS-LIST ())
 (SETQ LINE-EDGE-ALIST ())
 (SEND LINES ':SET-OBJECTS-LIST ())
 (SETQ DASHED-LINE-EDGE ALIST ())
 (SEND DASHED-LINES ':SET-OBJECTS-LIST ())
 (SEND S-I-WINDOW ':EXPOSE)
 (G-CLEAR WINDOW)
 (USING-RESOURCE (GRAPHIC-BOX-MENU GRAPHIC-BOX-MENU)
  (SEND GRAPHIC-BOX-MENU ':SET-EDGES 750. 131. 1088. 749.)
```

```
    (UNWIND-PROTECT
      (*CATCH 'EXIT-BOX-EDITOR
        (LET-GLOBALLY ((TV:WHO-LINE-PROCESS CURRENT-PROCESS))
          (DO ((COMMAND)
               (NEW-ALIST 'FIRST))
              (NIL)
            (TV:MOUSE-WARP MOUSE-X MOUSE-Y)
            (IF (SETQ COMMAND (FUNCALL GRAPHIC-BOX-MENU ':CHOOSE))
              (TV:DELAYING-SCREEN-MANAGEMENT
                (SETQ MOUSE-X TV:MOUSE-X MOUSE-Y TV:MOUSE-Y)
                (SETQ NEW-ALIST (FUNCALL COMMAND WINDOW ))))
            ))))))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; BOX ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;; G-BOX

(DEFFLAVOR G-BOX
       (LEFT TOP RIGHT BOTTOM (WINDOW TERMINAL-IO) (FILL-FLAG ()))
       ()
  (:INITABLE-INSTANCE-VARIABLES)
  (:GETTABLE-INSTANCE-VARIABLES)
  (:SETTABLE-INSTANCE-VARIABLES))

(DEFMETHOD (G-BOX :SHOW) ()
  (OR (NOT FILL-FLAG)
     (SEND WINDOW ':DRAW-RECTANGLE (- RIGHT LEFT) (- BOTTOM TOP) LEFT TOP 2))
  (SHOW-RECTANGLE LEFT TOP RIGHT BOTTOM WINDOW 5))

(DEFMETHOD (G-BOX :MOVE) ()
  (LET ((WIDTH (1+ (- RIGHT LEFT))) (HEIGHT (1+ (- BOTTOM TOP))))
    (MULTIPLE-VALUE-BIND (XOFF YOFF)
       (TV:SHEET-CALCULATE-OFFSETS WINDOW TV:MOUSE-SHEET)
     (TV:WITH-MOUSE-GRABBED
       (TV:MOUSE-SET-BLINKER-DEFINITION ':BOX-STAY-INSIDE-BLINKER 0 0 NIL
                             ':SET-SIZE WIDTH HEIGHT)
       (TV:MOUSE-WARP (+ 3 LEFT XOFF) (+ 3 TOP YOFF))
       (TV:BLINKER-SET-VISIBILITY TV:MOUSE-BLINKER T)
       (PROCESS-WAIT "RELEASE BUTTON" #'(LAMBDA () (ZEROP TV:MOUSE-LAST-BUTTONS)))
       (PROCESS-WAIT "BUTTON" #'(LAMBDA () (NOT (ZEROP TV:MOUSE-LAST-BUTTONS))))
       (MULTIPLE-VALUE-BIND (X Y) (FUNCALL TV:MOUSE-BLINKER ':READ-CURSORPOS)
         (SHOW-RECTANGLE LEFT TOP RIGHT BOTTOM WINDOW 0)
         (SETQ LEFT (- X 3 XOFF) TOP (- Y 3 YOFF))
         (SETQ RIGHT (1- (+ LEFT WIDTH)) BOTTOM (1- (+ TOP HEIGHT)))
         (SEND SELF ':SHOW)
         (TV:BLINKER-SET-VISIBILITY TV:MOUSE-BLINKER NIL)))))
  )

(DEFUN SHOW-RECTANGLE (X1 Y1 X2 Y2 &OPTIONAL (WINDOW TERMINAL-IO) (ALU 6))
  (SEND WINDOW ':DRAW-LINE X1 Y1 X1 Y2 ALU)            ;LEFT SIDE
  (SEND WINDOW ':DRAW-LINE (1+ X1) Y2 (1- X2) Y2 ALU)   ;LOWER SIDE
  (SEND WINDOW ':DRAW-LINE X2 Y2 X2 Y1 ALU)            ;RIGHT SIDE
  (SEND WINDOW ':DRAW-LINE (1- X2) Y1 (1+ X1) Y1 ALU))  ;UPPER SIDE

(DEFMETHOD (G-BOX :EDGES) ()
  (PROG () (RETURN LEFT TOP RIGHT BOTTOM)))

;;; G-BOX-WITH-MOUSE

(DEFFLAVOR G-BOX-WITH-MOUSE () (G-BOX)
  (:INITABLE-INSTANCE-VARIABLES)
  (:GETTABLE-INSTANCE-VARIABLES)
  (:SETTABLE-INSTANCE-VARIABLES))

(DEFMETHOD (G-BOX-WITH-MOUSE :AFTER :INIT) (&REST IGNORE)
   (MULTIPLE-VALUE-BIND (XOFF YOFF)
      (TV:SHEET-CALCULATE-OFFSETS WINDOW TV:MOUSE-SHEET)
    (TV:MOUSE-WARP 250. 250.)
    (MULTIPLE-VALUE-BIND (LEFT1 TOP1 RIGHT1 BOTTOM1)
      (TV:MOUSE-SPECIFY-RECTANGLE)
     (SETQ LEFT (- LEFT1 3 XOFF) TOP (- TOP1 3 YOFF)
         RIGHT (- RIGHT1 5 XOFF) BOTTOM (- BOTTOM1 5 YOFF))
    (SHOW-RECTANGLE LEFT TOP RIGHT BOTTOM WINDOW 5))))

;;; OBJECTS

(DEFFLAVOR OBJECTS ((OBJECTS-LIST ())) ()
  (:GETTABLE-INSTANCE-VARIABLES)
  (:INITABLE-INSTANCE-VARIABLES)
```

```
 (:SETTABLE-INSTANCE-VARIABLES))

(DEFMETHOD (OBJECTS :ADD-OBJECT) (OBJ)
 (PUSH OBJ OBJECTS-LIST))

(DEFMETHOD (OBJECTS :DELETE-OBJECT) (OBJ)
 (SETQ OBJECTS-LIST (DELETE OBJ OBJECTS-LIST)))

(DEFMETHOD (OBJECTS :SHOW-OBJECTS) ()
 (IF (EQ OBJECTS-LIST ()) ()
    (DOTIMES (I (LENGTH OBJECTS-LIST))
     (LET ((OBJ (NTH I (REVERSE OBJECTS-LIST))))
      (SEND OBJ ':SHOW)))))

(DEFMETHOD (OBJECTS :SET-WINDOW) (WINDOW)
 (IF (EQ OBJECTS-LIST ()) ()
    (DOTIMES (I (LENGTH OBJECTS-LIST))
     (LET ((OBJ (NTH I OBJECTS-LIST)))
      (SEND OBJ ':SET-WINDOW WINDOW)))))

(SETQ BOXES (MAKE-INSTANCE 'OBJECTS))

(DEFUN CREATE-BOX (W)
 (LET ((BOX (MAKE-BOX)))
  (PUSH (LIST BOX
          (SEND BOX ':LEFT)
          (SEND BOX ':TOP)
          (SEND BOX ':RIGHT)
          (SEND BOX ':BOTTOM)) BOX-EDGE-ALIST)))

(DEFUN M-BOX (W &REST IGNORE)
 (LET ((BOX (BOX-EDITOR-FIND-BOX T)))
  (IF (EQ BOX NIL)
     (BEEP)
     (SEND BOX ':MOVE)
     (PUSH (LIST BOX
             (SEND BOX ':LEFT)
             (SEND BOX ':TOP)
             (SEND BOX ':RIGHT)
             (SEND BOX ':BOTTOM)) BOX-EDGE-ALIST)
     (REDISPLAY W))))

(DEFUN MAKE-BOX ()
 (PROG ()
     (LET ((BOX (MAKE-INSTANCE 'G-BOX-WITH-MOUSE)))
      (SEND BOXES ':ADD-OBJECT BOX)
      (RETURN BOX))))

(DEFUN SET-FILL-FLAG (w &REST IGNORE)
 (LET ((BOX (BOX-EDITOR-FIND-BOX-with-check w t)))
  (IF (EQ BOX NIL)
     (BEEP)
   (SEND BOX ':SET-FILL-FLAG T)
   (SEND BOXES ':DELETE-OBJECT BOX)
   (SEND BOXES ':ADD-OBJECT BOX)
   (PUSH (LIST BOX
           (SEND BOX ':LEFT)
           (SEND BOX ':TOP)
           (SEND BOX ':RIGHT)
           (SEND BOX ':BOTTOM)) BOX-EDGE-ALIST))))

(DEFUN GB-COPY (W &REST IGNORE)
 (LET ((BOX (BOX-EDITOR-FIND-BOX ())))
  (IF (EQ BOX NIL)
     (BEEP)
     (LET ((NEW (MAKE-INSTANCE 'G-BOX))
         (LEFT (SEND BOX ':LEFT))
         (TOP (SEND BOX ':TOP))
         (RIGHT (SEND BOX ':RIGHT))
         (BOTTOM (SEND BOX ':BOTTOM)))
      (SEND NEW ':SET-LEFT LEFT)
      (SEND NEW ':SET-TOP TOP)
      (SEND NEW ':SET-RIGHT RIGHT)
      (SEND NEW ':SET-BOTTOM BOTTOM)
      (SEND NEW ':MOVE)
      (SEND BOX ':SHOW)
      (SEND BOXES ':ADD-OBJECT NEW)
      (PUSH (LIST NEW
              (SEND NEW ':LEFT)
              (SEND NEW ':TOP)
              (SEND NEW ':RIGHT)
```

```
            (SEND NEW ':BOTTOM)) BOX-EDGE-ALIST)))))

 (DEFUN D-BOX (W)
  (OR (SEND BOXES ':DELETE-OBJECT (BOX-EDITOR-FIND-BOX T))
      (BEEP))
  (REDISPLAY W))

 (DEFUN BOX-EDITOR-FIND-BOX (&OPTIONAL (DEL-F ())(CHAR 24))
  (TV:WITH-MOUSE-GRABBED
   (SETQ BOX NIL)
   (TV:MOUSE-WARP 300. 300.)
   (PROCESS-WAIT "BUTTON UP" #'(LAMBDA () (ZEROP TV:MOUSE-LAST-BUTTONS)))
   (TV:MOUSE-SET-BLINKER-DEFINITION ':CHARACTER 0 0 ':ON
                                    ':SET-CHARACTER CHAR)
   (PROCESS-WAIT "BUTTON DOWN" #'(LAMBDA () (NOT (ZEROP TV:MOUSE-LAST-BUTTONS))))
   (SETQ X TV:MOUSE-X Y TV:MOUSE-Y)
   (AND (BIT-TEST 1 TV:MOUSE-LAST-BUTTONS)
       (DOLIST (B BOX EDGE-ALIST)
        (AND (≥ X (SECOND B)) (≥ Y (THIRD B))
            (< X (FOURTH B)) (< Y (FIFTH B))
            (OR (EQ DEL-F ())(SETQ BOX EDGE-ALIST (DELETE B BOX-EDGE-ALIST)) T)
            (RETURN (SETQ BOX (FIRST B))))))))
  BOX)

 (DEFUN BOX-EDITOR-FIND-BOX-WITH-CHECK (WINDOW &OPTIONAL (DEL-F ())(CHAR 24))
  (TV:WITH-MOUSE GRABBED
   (SETQ BOX NIL)
   (TV:MOUSE-WARP 300. 300.)
   (PROCESS-WAIT "BUTTON UP" #'(LAMBDA () (ZEROP TV:MOUSE-LAST-BUTTONS)))
   (TV:MOUSE-SET-BLINKER-DEFINITION ':CHARACTER 0 0 ':ON
                                    ':SET-CHARACTER CHAR)
   (PROCESS-WAIT "BUTTON DOWN" #'(LAMBDA () (NOT (ZEROP TV:MOUSE-LAST-BUTTONS))))
   (SETQ X TV:MOUSE-X Y TV:MOUSE-Y)
   (AND (BIT-TEST 1 TV:MOUSE-LAST-BUTTONS)
       (DOLIST (B BOX EDGE-ALIST)
        (AND (≥ X (SECOND B)) (≥ Y (THIRD B))
            (< X (FOURTH B)) (< Y (FIFTH B))
            (check-box (first b) window)
            (OR (EQ DEL-F ())(SETQ BOX-EDGE-ALIST (DELETE B BOX-EDGE-ALIST)) T)
            (RETURN (SETQ BOX (FIRST B))))))))
  BOX)

 (DEFUN CHECK-BOX (BOX WINDOW)
  (LET ((LEFT (SEND BOX ':LEFT))
       (TOP (SEND BOX ':TOP))
       (RIGHT (SEND BOX ':RIGHT))
       (BOTTOM (SEND BOX ':BOTTOM)))
   (MULTIPLE-VALUE-BIND (XOFF YOFF)
      (TV:SHEET-CALCULATE-OFFSETS WINDOW TV:MOUSE-SHEET)
     (TV:WITH-MOUSE-GRABBED
      (TV:MOUSE-SET-BLINKER-DEFINITION
        ':BOX-STAY-INSIDE-BLINKER 0 0 NIL
        ':SET-SIZE (1+ (- RIGHT LEFT)) (1+ (- BOTTOM TOP)))
      (TV:MOUSE-WARP (+ 3 LEFT XOFF) (+ 3 TOP YOFF))
      (TV:BLINKER-SET-VISIBILITY TV:MOUSE-BLINKER T)
      (PROCESS-WAIT "RELEASE BUTTON" #'(LAMBDA () (ZEROP TV:MOUSE-LAST-BUTTONS)))
      (PROCESS WAIT "BUTTON" #'(LAMBDA () (NOT (ZEROP TV:MOUSE-LAST-BUTTONS))))
      (IF (EQ TV:MOUSE-LAST-BUTTONS 2)
          ()
          T)))))
;;;;;;;;;;;;;;;;;;;;;;;;;;   CIRCULAR BOX   ;;;;;;;;;;;;;;;;;;;;;;;;;;


(DEFUN MAKE-C-BOX (WINDOW &REST IGNORE)
 (PROG ()
     (MULTIPLE-VALUE-BIND (XOFF YOFF)
        (TV:SHEET-CALCULATE-OFFSETS WINDOW TV:MOUSE-SHEET)
      (TV:MOUSE-WARP 250. 250.)
      (MULTIPLE-VALUE BIND (LEFT1 TOP1 RIGHT1 BOTTOM1)
         (TV:MOUSE-SPECIFY-RECTANGLE)
       (LET ((LEFT (- LEFT1 3 XOFF)) (TOP (- TOP1 3 YOFF))
             (RIGHT (- RIGHT1 5 XOFF)) (BOTTOM (- BOTTOM1 5 YOFF)))
        (LET ((C-BOX (MAKE-INSTANCE 'C-BOX)))
         (SEND C-BOX ':SET-LEFT LEFT)
         (SEND C-BOX ':SET-TOP TOP)
         (SEND C-BOX ':SET-RIGHT RIGHT)
         (SEND C-BOX ':SET-BOTTOM BOTTOM)
         (SEND C-BOXES ':ADD-OBJECT C-BOX)
         (SEND C-BOX ':SHOW)
         (RETURN C-BOX)))))))
```

```
(DEFFLAVOR C-BOX (LEFT TOP RIGHT BOTTOM (WINDOW TERMINAL-IO)) ()
  (:INITABLE-INSTANCE-VARIABLES)
  (:GETTABLE-INSTANCE-VARIABLES)
  (:SETTABLE-INSTANCE-VARIABLES))

(DEFMETHOD (C-BOX :SHOW) ()
  (IF (≥ (- RIGHT LEFT) (- BOTTOM TOP))
      (SHOW-C-BOX-HORIZONTAL)
      (SHOW-C-BOX-VERTICAL)))

(DEFUN-METHOD SHOW-C-BOX-HORIZONTAL C-BOX (&REST IGNORE)
  (LET ((RADIUS (// (- BOTTOM TOP) 2)))
    (LET ((CENTER-X1 (+ LEFT RADIUS))
          (CENTER-Y1 (+ TOP RADIUS))
          (CENTER-X2 (- RIGHT RADIUS))
          (CENTER-Y2 (+ TOP RADIUS)))
      (SEND WINDOW ':DRAW-CIRCULAR-ARC
            CENTER-X1 CENTER-Y1 RADIUS (// PAI 2) (// (* PAI 3) 2))
      (SEND WINDOW ':DRAW-CIRCULAR-ARC
            CENTER-X2 CENTER-Y2 RADIUS (// (* PAI 3) 2) (// PAI 2))
      (SEND WINDOW ':DRAW-LINE CENTER-X1 TOP CENTER-X2 TOP)
      (SEND WINDOW ':DRAW-LINE CENTER-X1 BOTTOM CENTER-X2 BOTTOM))))

(DEFUN-METHOD SHOW-C-BOX-VERTICAL C-BOX (&REST IGNORE)
  (LET ((RADIUS (// (- RIGHT LEFT) 2)))
    (LET ((CENTER-X1 (+ LEFT RADIUS))
          (CENTER-Y1 (+ TOP RADIUS))
          (CENTER-X2 (+ LEFT  RADIUS))
          (CENTER-Y2 (- BOTTOM RADIUS)))
      (SEND WINDOW ':DRAW-CIRCULAR-ARC CENTER-X1 CENTER-Y1 RADIUS 0 PAI)
      (SEND WINDOW ':DRAW-CIRCULAR-ARC CENTER-X2 CENTER-Y2 RADIUS PAI 0)
      (SEND WINDOW ':DRAW-LINE LEFT CENTER-Y1 LEFT CENTER-Y2)
      (SEND WINDOW ':DRAW-LINE RIGHT CENTER-Y1 RIGHT CENTER-Y2))))

(SETQ C-BOXES (MAKE-INSTANCE 'OBJECTS))

(DEFMETHOD (C-BOX :MOVE) (&REST IGNORE)
  (LET ((WIDTH (1+ (- RIGHT LEFT))) (HEIGHT (1+ (- BOTTOM TOP))))
    (MULTIPLE-VALUE-BIND (XOFF YOFF)
        (TV:SHEET-CALCULATE-OFFSETS WINDOW TV:MOUSE-SHEET)
      (TV:WITH-MOUSE-GRABBED
        (TV:MOUSE-SET-BLINKER-DEFINITION ':BOX-STAY-INSIDE-BLINKER 0 0 NIL
                                         ':SET-SIZE WIDTH HEIGHT)
        (TV:MOUSE-WARP (+ 3 LEFT XOFF) (+ 3 TOP YOFF))
        (TV:BLINKER-SET-VISIBILITY TV:MOUSE-BLINKER T)
        (PROCESS-WAIT "RELEASE BUTTON" #'(LAMBDA () (ZEROP TV:MOUSE-LAST-BUTTONS)))
        (PROCESS-WAIT "BUTTON" #'(LAMBDA () (NOT (ZEROP TV:MOUSE-LAST-BUTTONS))))
        (MULTIPLE-VALUE-BIND (X Y) (FUNCALL TV:MOUSE-BLINKER ':READ-CURSORPOS)
          (SETQ LEFT (- X 3 XOFF) TOP (- Y 3 YOFF))
          (SETQ RIGHT (1- (+ LEFT WIDTH)) BOTTOM (1- (+ TOP HEIGHT)))
          (SEND SELF ':SHOW)
          (TV:BLINKER-SET-VISIBILITY TV:MOUSE-BLINKER NIL))))))

(DEFUN CREATE-C-BOX (W)
  (LET ((C-BOX (MAKE-C-BOX W)))
    (PUSH (LIST C-BOX
                (SEND C-BOX ':LEFT)
                (SEND C-BOX ':TOP)
                (SEND C-BOX ':RIGHT)
                (SEND C-BOX ':BOTTOM)) C-BOX-EDGE-ALIST)))

(DEFUN MOVE-C-BOX (W &REST IGNORE)
  (LET ((C-BOX (BOX-EDITOR-FIND-C-BOX T)))
    (IF (EQ C-BOX NIL)
        (BEEP)
        (SEND C-BOX ':MOVE)
        (PUSH (LIST C-BOX
                    (SEND C-BOX ':LEFT)
                    (SEND C-BOX ':TOP)
                    (SEND C-BOX ':RIGHT)
                    (SEND C-BOX ':BOTTOM)) C-BOX-EDGE-ALIST)
        (REDISPLAY W))))

(DEFUN BOX-EDITOR-FIND-C-BOX (&OPTIONAL (DEL-F ())(CHAR 24))
  (TV:WITH-MOUSE-GRABBED
    (SETQ C-BOX NIL)
    (TV:MOUSE-WARP 300. 300.)
    (PROCESS-WAIT "BUTTON UP" #'(LAMBDA () (ZEROP TV:MOUSE-LAST-BUTTONS)))
    (TV:MOUSE-SET-BLINKER-DEFINITION ':CHARACTER 0 0 ':ON
                                     ':SET-CHARACTER CHAR)
    (PROCESS-WAIT "BUTTON DOWN" #'(LAMBDA () (NOT (ZEROP TV:MOUSE-LAST-BUTTONS))))
```

```
    (SETQ X TV:MOUSE-X Y TV:MOUSE-Y)
    (AND (BIT-TEST 1 TV:MOUSE-LAST-BUTTONS)
        (DOLIST (C-B C-BOX-EDGE-ALIST)
          (AND (≥ X (SECOND C-B)) (≥ Y (THIRD C-B))
              (< X (FOURTH C-B)) (< Y (FIFTH C-B))
              (OR (EQ DEL-F ())(SETQ C-BOX-EDGE-ALIST (DELETE C-B C-BOX-EDGE-ALIST)) t)
              (RETURN (SETQ C-BOX (FIRST C-B)))))))
  C-BOX)

 (DEFMETHOD (C-BOX :EDGES) ()
  (PROG () (RETURN LEFT TOP RIGHT BOTTOM)))

 (DEFUN COPY-C-BOX (W &REST IGNORE)
  (LET ((C-BOX (BOX-EDITOR-FIND-C-BOX ())))
   (IF (EQ C-BOX NIL)
      (BEEP)
      (LET ((NEW (MAKE-INSTANCE 'C-BOX))
           (LEFT (SEND C-BOX ':LEFT))
           (TOP (SEND C-BOX ':TOP))
           (RIGHT (SEND C-BOX ':RIGHT))
           (BOTTOM (SEND C-BOX ':BOTTOM)))
        (SEND NEW ':SET-LEFT LEFT)
        (SEND NEW ':SET-TOP TOP)
        (SEND NEW ':SET-RIGHT RIGHT)
        (SEND NEW ':SET-BOTTOM BOTTOM)
        (SEND NEW ':MOVE)
        (SEND C-BOX ':SHOW)
        (SEND C-BOXES ':ADD-OBJECT NEW)
        (PUSH (LIST NEW
                 (SEND NEW ':LEFT)
                 (SEND NEW ':TOP)
                 (SEND NEW ':RIGHT)
                 (SEND NEW ':BOTTOM)) C-BOX-EDGE-ALIST)))))

(DEFUN DELETE-C-BOX (W)
 (OR (SEND C-BOXES ':DELETE-OBJECT (BOX-EDITOR-FIND-C-BOX T))
    (BEEP))
 (REDISPLAY W))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;    STRING    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;; STRING

(DEFFLAVOR G-STRING (LEFT TOP RIGHT BOTTOM (STRING ()) (WINDOW TERMINAL-IO)
                  (FONT FONTS:TR18B)) ()
 (:INITABLE-INSTANCE-VARIABLES)
 (:GETTABLE-INSTANCE-VARIABLES)
 (:SETTABLE INSTANCE-VARIABLES))

(DEFMETHOD (G-STRING :SHOW) ()
 (SEND WINDOW ':DRAW-STRING-f STRING LEFT
     (- BOTTOM (- (FED:FONT-BLINKER-HEIGHT FONT)
             (FED:FONT-BASELINE FONT)))
     FONT))


(DEFUN C-STRING (W)
 (prog () loop
      (cond ((eq (errset
 (LET ((STRING (MAKE-STRING W)))
  (SEND STRING ':MOVE)
  (PUSH (LIST STRING (SEND STRING ':LEFT)
                (SEND STRING ':TOP)
                (SEND STRING ':RIGHT)
                (SEND STRING ':BOTTOM))
      STRING-EDGE-ALIST)) ()) nil) (beep) (go loop)))))

(DEFUN MAKE-STRING (W)
      (SETQ CHR NIL)
      (SEND S-i-WINDOW ':SELECT)
      (SEND S-i-WINDOW ':REFRESH)
      (SEND S-i-WINDOW ':SET-CURSORPOS 0 0)
      (setq chrs (readline s-i-window))
      (send s-window ':refresh)
      (send s-window ':set-cursorpos 0 0)
      (send s-window ':set-font-map (list default-font))
      (send s-window ':string-out chrs)
      (LET ((X (SEND S-WINDOW ':CURSOR-X))
          (Y (FED:FONT-BLINKER-HEIGHT DEFAULT-FONT)))
            (SETQ OBJ (MAKE-INSTANCE 'G-STRING
                          ':LEFT 0
```

```
                                ':TOP 0
                                ':RIGHT (1- X)
                                ':BOTTOM (1- Y)
                                ':STRING CHRS
                                ':FONT DEFAULT-FONT)))
      (SEND W ':SELECT)
      (SEND S-WINDOW ':BURY)
      (SEND STRINGS ':ADD-OBJECT OBJ)
      obj)

(DEFUN RUBOUT ()
 (POP CHRS)
 (SEND S-I-WINDOW ':REFRESH)
 (SEND S-I-WINDOW ':SET-CURSORPOS 0 0)
 (DOLIST (C (REVERSE CHRS))
  (SEND S-I-WINDOW ':TYO C)))

(DEFUN COPY-STRING (W &REST IGNORE)
 (LET ((STRING (BOX-EDITOR-FIND-STRING ())))
  (OR (EQ STRING ())
      (LET* ((LEFT (SEND STRING ':LEFT))
             (TOP (SEND STRING ':TOP))
             (RIGHT (SEND STRING ':RIGHT))
             (BOTTOM (SEND STRING ':BOTTOM))
             (S (SEND STRING ':STRING))
             (FONT (SEND STRING ':FONT))
             (OBJ (MAKE-INSTANCE 'G-STRING
                                 ':LEFT LEFT
                                 ':TOP TOP
                                 ':RIGHT RIGHT
                                 ':BOTTOM BOTTOM
                                 ':STRING S
                                 ':FONT FONT)))
        (SEND OBJ ':MOVE)
        (SEND STRINGS ':ADD-OBJECT OBJ)
        (PUSH (LIST OBJ (SEND OBJ ':LEFT)
                        (SEND OBJ ':TOP)
                        (SEND OBJ ':RIGHT)
                        (SEND OBJ ':BOTTOM))
              STRING-EDGE-ALIST)))))


(DEFMETHOD (G-STRING :MOVE) (&REST IGNORE)
 (LET ((WIDTH (1+ (- RIGHT LEFT))) (HEIGHT (1+ (- BOTTOM TOP))))
  (MULTIPLE-VALUE-BIND (XOFF YOFF)
      (TV:SHEET-CALCULATE-OFFSETS WINDOW TV:MOUSE-SHEET)
    (TV:WITH-MOUSE-GRABBED
     (TV:MOUSE-SET-BLINKER-DEFINITION ':BOX-STAY-INSIDE-BLINKER 0 0 NIL
                                      ':SET-SIZE WIDTH HEIGHT)
     (TV:MOUSE-WARP (+ 3 LEFT XOFF) (+ 3 TOP YOFF))
     (TV:BLINKER-SET-VISIBILITY TV:MOUSE-BLINKER T)
     (PROCESS-WAIT "RELEASE BUTTON" #'(LAMBDA () (ZEROP TV:MOUSE-LAST-BUTTONS)))
     (PROCESS-WAIT "BUTTON" #'(LAMBDA () (NOT (ZEROP TV:MOUSE-LAST-BUTTONS))))
     (MULTIPLE-VALUE-BIND (X Y) (FUNCALL TV:MOUSE-BLINKER ':READ-CURSORPOS)
      (SETQ LEFT (- X 3 XOFF) TOP (- Y 3 YOFF))
      (SETQ RIGHT (1- (+ LEFT WIDTH)) BOTTOM (1- (+ TOP HEIGHT)))
      (SEND SELF ':SHOW)
      (TV:BLINKER-SET-VISIBILITY TV:MOUSE-BLINKER NIL))))))

(DEFUN M-STRING (W &REST IGNORE)
 (LET ((STRING (BOX-EDITOR-FIND-STRING T)))
  (IF (EQ STRING NIL)
      (BEEP)
      (SEND STRING ':MOVE)
      (PUSH (LIST STRING
                  (SEND STRING ':LEFT)
                  (SEND STRING ':TOP)
                  (SEND STRING ':RIGHT)
                  (SEND STRING ':BOTTOM)) STRING-EDGE-ALIST)
      (REDISPLAY W))))

(DEFUN D-STRING (W)
 (OR (SEND STRINGS ':DELETE-OBJECT (BOX-EDITOR-FIND-STRING T))
     (BEEP))
 (REDISPLAY W))

(DEFUN BOX-EDITOR-FIND-STRING (&OPTIONAL (DEL-F ())(CHAR 24))
 (SETQ STRING NIL)
 (TV:WITH-MOUSE-GRABBED
  (TV:MOUSE-WARP 300. 300.)
  (PROCESS-WAIT "BUTTON UP" #'(LAMBDA () (ZEROP TV:MOUSE-LAST-BUTTONS)))
```

```
    (TV:MOUSE-SET-BLINKER-DEFINITION ':CHARACTER 0 0 ':ON
                                     ':SET-CHARACTER CHAR)
    (PROCESS-WAIT "BUTTON DOWN" #'(LAMBDA () (NOT (ZEROP TV:MOUSE-LAST-BUTTONS))))
    (SETQ X TV:MOUSE-X Y TV:MOUSE-Y)
    (AND (BIT-TEST 1 TV:MOUSE-LAST-BUTTONS)
         (DOLIST (B STRING-EDGE-ALIST)
           (AND (≥ X (SECOND B)) (≥ Y (THIRD B))
                (< X (FOURTH B)) (< Y (FIFTH B))
                (OR (EQ DEL-F ())(SETQ STRING EDGE-ALIST (DELETE B STRING-EDGE-ALIST)) T)
                (RETURN (SETQ STRING (FIRST B)))))))
  STRING)

(DEFUN LIST-FONTS LOADED ()
  (SEND FONT-SELECT-WINDOW ':SELECT)
  (FORMAT FONT-SELECT-WINDOW "LOADED FONTS - MOUSE ONE TO SELECT~%")
  (FUNCALL FONT-SELECT-WINDOW ':ITEM-LIST 'FONT
           (LOCAL-DECLARE ((SPECIAL LIST))
             (LET ((LIST NIL))
               (MAPATOMS ALL #'(LAMBDA (X) (AND (BOUNDP X)(TYPEP (SYMEVAL X) 'FONT)
                                                (PUSH X LIST)))
                         "FONTS")
               (SORT LIST #'STRING-LESSP)))))


(DEFUN LIST-FONTS-FILE-COMPUTER ()
  (SEND FONT-SELECT-WINDOW ':SELECT)
  (FORMAT FONT-SELECT-WINDOW "
FONTS ON DISK- MOUSE ONE TO SELECT~%")
  (FUNCALL FONT-SELECT-WINDOW ':ITEM-LIST 'FONT
           (SORT (MAPCAR #'(LAMBDA (X) (INTERN (STRING (FUNCALL (CAR X) ':NAME)) "FONTS"))
                 (CDR (FS:DIRECTORY-LIST
                        (SEND (FS:PARSE-PATHNAME "SYS:FONTS;")
                              ':NEW-PATHNAME ':NAME ':WILD
                              ':CANONICAL-TYPE SI:*DEFAULT-BINARY-FILE-TYPE*
                              ':VERSION ':NEWEST)
                        ':FAST)))
            #'ALPHALESSP)))


(DEFUN LOAD-FONT-FROM-SYM (SYM &OPTIONAL (QUERY-P NIL))
  (LET ((FONT-PATH (SEND (FED:FONT-PATHNAME-TEMPLATE) ':NEW-PATHNAME ':NAME (STRING SYM)
                         ':CANONICAL-TYPE SI:*DEFAULT-BINARY-FILE-TYPE*)))
    (AND (PROBEF FONT-PATH)
         (OR (NOT QUERY-P) (FED:PROMPT-LINE-Y-OR-N-P "LOAD ~A? " FONT-PATH))
         (LOAD FONT-PATH "FONTS"))))
(DEFUN LIST-FONTS-ON-TSUJI ()
  (SEND FONT-SELECT-WINDOW ':SELECT)
  (FORMAT FONT-SELECT-WINDOW "
FONTS IN >TSUJI>FONTS>- MOUSE ONE TO SELECT~%")
  (FUNCALL FONT-SELECT-WINDOW ':ITEM-LIST 'FONT
           (SORT (MAPCAR #'(LAMBDA (X) (INTERN (STRING (FUNCALL (CAR X) ':NAME)) "FONTS"))
                 (CDR (FS:DIRECTORY-LIST
                        (SEND (FS:PARSE-PATHNAME ">TSUJI>FONTS>")
                              ':NEW-PATHNAME ':NAME ':WILD
                              ':CANONICAL-TYPE SI:*DEFAULT-BINARY-FILE-TYPE*
                              ':VERSION ':NEWEST)
                        ':FAST)))
            #'ALPHALESSP)))

(DEFUN FONT-SELECT ()
  (SEND FONT-SELECT-WINDOW ':SELECT)
  (SEND FONT-SELECT-WINDOW ':SET-CURSORPOS 0 0)
  (SEND FONT-SELECT-WINDOW ':REFRESH)
  (LIST-FONTS-LOADED)
  (LIST-FONTS-FILE-COMPUTER)
  (LIST-FONTS-ON-TSUJI)
  (TV:MOUSE-WARP 300. 50.)
  (TV:WITH-MOUSE-GRABBED
    (PROCESS-WAIT "BUTTON UP" #'(LAMBDA () (ZEROP TV:MOUSE-LAST-BUTTONS)))
    (TV:MOUSE-SET-BLINKER-DEFINITION ':CHARACTER 0 0 ':ON
                                     ':SET-CHARACTER 24)
    (PROCESS-WAIT "BUTTON DOWN" #'(LAMBDA () (NOT (ZEROP TV:MOUSE-LAST-BUTTONS))))
    (SETQ FONT-NAME
          (SECOND
            (SEND FONT-SELECT-WINDOW ':MOUSE-SENSITIVE-ITEM
                  (+ 3 TV:MOUSE-X) (+ 5 TV:MOUSE-Y)))))
    (IF (BOUNDP FONT-NAME)
        ()
        (OR (AND (> 500. TV:MOUSE-Y)(LOAD-FONT-FROM-SYM FONT-NAME))
            (LOAD (IMPLODE (APPEND
```

```
                (EXPLODE '>TSUJI>FONTS>)
                (DELETE '|| (EXPLODE (STRING FONT-NAME))))) "FONTS")))))


(DEFUN SET-FONT (W &REST IGNORE)
 (LET ((STRING (BOX-EDITOR-FIND-STRING T)))
  (IF (EQ STRING NIL)
     (BEEP)
     (SEND FONT-SELECT-WINDOW ':SELECT)
     (FONT-SELECT)
     (LET ((FONT (EVAL font-name)))
       (SEND STRING ':SET-FONT FONT)
       (SEND S-WINDOW ':SET-FONT-MAP '(FONT))
       (SEND S-WINDOW ':SET-CURSORPOS 0 0)
       (SEND S-WINDOW ':STRING OUT (SEND STRING ':STRING))
       (MULTIPLE-VALUE-BIND (X Y)
          (SEND S-WINDOW ':READ-CURSORPOS)
        (SEND S-WINDOW ':TYO 141.)
        (MULTIPLE-VALUE-BIND (XX YY)
           (SEND S-WINDOW ':READ-CURSORPOS)
          (SEND STRING ':SET-RIGHT (+ (SEND STRING ':LEFT) (1- X)))
          (SEND STRING ':SET-BOTTOM (+ (SEND STRING ':TOP) (1- YY)))
          (PUSH (LIST STRING
                  (SEND STRING ':LEFT)
                  (SEND STRING ':TOP)
                  (SEND STRING ':RIGHT)
                  (SEND STRING ':BOTTOM)) STRING-EDGE-ALIST)
          (SEND S-WINDOW ':SET-FONT-MAP '(DEFAULT-FONT))
          (SEND W ':SELECT)
          (REDISPLAY W)))))))

(DEFUN SET-DEFAULT-FONT (W &REST IGNORE)
      (SEND FONT-SELECT-WINDOW ':SELECT)
      (FONT-SELECT)
      (SETQ DEFAULT-FONT (eval FONT-name))
      (SEND W ':SELECT))


(SETQ STRINGS (MAKE-INSTANCE 'OBJECTS))


;;;;;;;;;;;;;;;;    LINE    ;;;;;;;;;;;;;;;;

(DEFFLAVOR LINE
     (START-X START-Y END-X END-Y (WINDOW TERMINAL-IO))
     ()
 (:INITABLE-INSTANCE-VARIABLES)
 (:GETTABLE-INSTANCE-VARIABLES)
 (:SETTABLE-INSTANCE-VARIABLES))

(DEFMETHOD (LINE :SHOW) ()
 (SEND WINDOW ':DRAW-LINE START-X START-Y END-X END-Y))

(DEFUN C-LINE (W)
 (LET ((LINE (MAKE-LINE W)))
  (PUSH (LIST LINE (SEND LINE ':START-X)
               (SEND LINE ':START-Y)
               (SEND LINE ':END-X)
               (SEND LINE ':END-Y))
       LINE-EDGE-ALIST)))

(DEFUN MAKE-LINE (W &OPTIONAL (CHAR 7))
 (PROG ()
 (TV:WITH-MOUSE-GRABBED
   (tv:mouse-warp 250. 250.)
   (PROCESS-WAIT "BUTTON UP" #'(LAMBDA () (ZEROP TV:MOUSE-LAST-BUTTONS)))
   (TV:MOUSE-SET-BLINKER-DEFINITION ':CHARACTER 0 0 ':ON
                          ':SET-CHARACTER CHAR)
   (PROCESS-WAIT "BUTTON DOWN" #'(LAMBDA () (NOT (ZEROP TV:MOUSE-LAST-BUTTONS))))
   (SETQ X TV:MOUSE-X Y TV:MOUSE-Y)
   (BEEP)
   (PROCESS-WAIT "BUTTON UP" #'(LAMBDA () (ZEROP TV:MOUSE-LAST-BUTTONS)))
   (TV:MOUSE-SET-BLINKER-DEFINITION ':CHARACTER 0 0 ':ON
                          ':SET-CHARACTER CHAR)
   (PROCESS-WAIT "BUTTON DOWN" #'(LAMBDA () (NOT (ZEROP TV:MOUSE-LAST-BUTTONS))))
   (SETQ XX TV:MOUSE-X YY TV:MOUSE-Y)
   (LET ((ALINE (MAKE-INSTANCE 'LINE ':START-X (1+ X)
                 ':START-Y (+ 2 Y)
                 ':END-X (1+ XX)
```

```
                    ':END-Y (+ 2 YY)
                    ':WINDOW W)))
      (SEND ALINE ':SHOW)
      (SEND LINES ':ADD-OBJECT ALINE)
      (RETURN ALINE)))))

(SETQ LINES (MAKE-INSTANCE 'OBJECTS))

;;;;;;;;;;;;;;;; VERTICAL/HORIZONTAL LINE ;;;;;;;;;;;;;;;;;;;

(DEFUN V-LINE (W)
  (LET ((LINE (MAKE-V-LINE W)))
    (PUSH (LIST LINE (SEND LINE ':START-X)
                     (SEND LINE ':START-Y)
                     (SEND LINE ':END-X)
                     (SEND LINE ':END-Y))
          LINE-EDGE-ALIST)))

(DEFUN MAKE-V-LINE (W &OPTIONAL (CHAR 7))
  (PROG ()
  (TV:WITH-MOUSE-GRABBED
    (TV:MOUSE-WARP 250. 250.)
    (PROCESS-WAIT "BUTTON UP" #'(LAMBDA () (ZEROP TV:MOUSE-LAST-BUTTONS)))
    (TV:MOUSE-SET-BLINKER-DEFINITION ':CHARACTER 0 0 ':ON
                                     ':SET-CHARACTER CHAR)
    (PROCESS-WAIT "BUTTON DOWN" #'(LAMBDA () (NOT (ZEROP TV:MOUSE-LAST-BUTTONS))))
    (SETQ X TV:MOUSE-X Y TV:MOUSE-Y)
    (BEEP)
    (PROCESS-WAIT "BUTTON UP" #'(LAMBDA () (ZEROP TV:MOUSE-LAST-BUTTONS)))
    (TV:MOUSE-SET-BLINKER-DEFINITION ':CHARACTER 0 0 ':ON
                                     ':SET CHARACTER CHAR)
    (PROCESS-WAIT "BUTTON DOWN" #'(LAMBDA () (NOT (ZEROP TV:MOUSE-LAST-BUTTONS))))
    (SETQ YY TV:MOUSE-Y)
    (LET ((ALINE (MAKE-INSTANCE 'LINE ':START-X (1+ X)
                      ':START-Y (+ 2 Y)
                      ':END-X (1+ X)
                      ':END-Y (+ 2 YY)
                      ':WINDOW W)))
      (SEND ALINE ':SHOW)
      (SEND LINES ':ADD-OBJECT ALINE)
      (RETURN ALINE)))))

(DEFUN H-LINE (W)
  (LET ((LINE (MAKE-H-LINE W)))
    (PUSH (LIST LINE (SEND LINE ':START-X)
                     (SEND LINE ':START-Y)
                     (SEND LINE ':END-X)
                     (SEND LINE ':END-Y))
          LINE-EDGE-ALIST)))

(DEFUN MAKE-H-LINE (W &OPTIONAL (CHAR 7))
  (PROG ()
  (TV:WITH-MOUSE-GRABBED
    (tv:mouse-warp 250. 250.)
    (PROCESS-WAIT "BUTTON UP" #'(LAMBDA () (ZEROP TV:MOUSE-LAST-BUTTONS)))
    (TV:MOUSE-SET-BLINKER-DEFINITION ':CHARACTER 0 0 ':ON
                                     ':SET-CHARACTER CHAR)
    (PROCESS-WAIT "BUTTON DOWN" #'(LAMBDA () (NOT (ZEROP TV:MOUSE-LAST-BUTTONS))))
    (SETQ X TV:MOUSE-X Y TV:MOUSE-Y)
    (BEEP)
    (PROCESS-WAIT "BUTTON UP" #'(LAMBDA () (ZEROP TV:MOUSE-LAST-BUTTONS)))
    (TV:MOUSE-SET-BLINKER-DEFINITION ':CHARACTER 0 0 ':ON
                                     ':SET-CHARACTER CHAR)
    (PROCESS-WAIT "BUTTON DOWN" #'(LAMBDA () (NOT (ZEROP TV:MOUSE-LAST-BUTTONS))))
    (SETQ XX TV:MOUSE-X)
    (LET ((ALINE (MAKE-INSTANCE 'LINE ':START-X (1+ X)
                      ':START-Y (+ 2 Y)
                      ':END-X (1+ XX)
                      ':END-Y (+ 2 Y)
                      ':WINDOW W)))
      (SEND ALINE ':SHOW)
      (SEND LINES ':ADD-OBJECT ALINE)
      (RETURN ALINE)))))

(DEFUN BOX-EDITOR-FIND-LINE (&OPTIONAL (DEL-F ())(CHAR 24))
  (SETQ LINE NIL)
  (TV:WITH-MOUSE-GRABBED
    (tv:mouse-warp 300. 300.)
    (PROCESS-WAIT "BUTTON UP" #'(LAMBDA () (ZEROP TV:MOUSE-LAST-BUTTONS)))
    (TV:MOUSE-SET-BLINKER-DEFINITION ':CHARACTER 0 0 ':ON
                                     ':SET-CHARACTER CHAR)
```

```
  (PROCESS-WAIT "BUTTON DOWN" #'(LAMBDA () (NOT (ZEROP TV:MOUSE-LAST-BUTTONS))))
  (SETQ X TV:MOUSE-X Y TV:MOUSE-Y)
  (AND (BIT-TEST 1 TV:MOUSE-LAST-BUTTONS)
       (DOLIST (L LINE-EDGE-ALIST)
        (LET ((X1 (SECOND L))
              (Y1 (THIRD L))
              (X2 (FOURTH L))
              (Y2 (FIFTH L)))
          (AND (> 0.1 (ABS (- (ATAN (- Y1 Y2) (- X1 X2))
                              (ATAN (- Y Y2) (- X X2)))))
               (> 0.1 (ABS (- (ATAN (- Y2 Y1) (- X2 X1))
                              (ATAN (- Y Y1) (- X X1)))))
               (OR (EQ DEL-F ())(SETQ LINE-EDGE-ALIST (DELETE L LINE-EDGE-ALIST)) T)
               (RETURN (SETQ LINE (FIRST L))))))))
 LINE)

(DEFUN D LINE (W)
 (OR (SEND LINES :DELETE-OBJECT (BOX-EDITOR-FIND-LINE T))
     (BEEP))
 (REDISPLAY W))
```

;;;;;;;;;;;;;;;; HARDCOPY ;;;;;;;;;;;;;;;;

```
TV:
(DEFUN HARDCOPY (SHEET)
 (IF SI:*SCREEN-HARDCOPY-DEVICE*
     (MULTIPLE-VALUE BIND (FROM-ARRAY WIDTH HEIGHT)
       (SNAPSHOT-SCREEN-DECODE-ARRAY SHEET)
      (USING-RESOURCE (TO-ARRAY HARDCOPY-BIT-ARRAY (LOGAND -40 (+ WIDTH 37)) HEIGHT)
       (IF (EQ *SCREEN-HARDCOPY-ANNOUNCEMENT* :FLASH)
          (COMPLEMENT-BOW-MODE))
       (SNAPSHOT-SCREEN FROM-ARRAY TO-ARRAY WIDTH HEIGHT)
       (IF (EQ *SCREEN-HARDCOPY-ANNOUNCEMENT* :BEEP)
          (BEEP))
       (IF (EQ *SCREEN-HARDCOPY ANNOUNCEMENT* :FLASH)
          (COMPLEMENT-BOW-MODE))
       (WITH-OPEN-STREAM (STREAM (SI:MAKE-HARDCOPY-STREAM
                                  SI:*SCREEN HARDCOPY-DEVICE*
                                  :BITMAP-ONLY-P T
                                  :BANNER-FILE-NAME "SCREEN HARDCOPY"))
         (SEND STREAM :SHOW-BITMAP TO-ARRAY WIDTH HEIGHT))))
    (NOTIFY NIL "I Don't Know How To Hardcopy The Screen At Your Site")))


tv:
(DEFMETHOD (GRAPHICS-MIXIN :DRAW-STRING-f)
        (STRING FROM-X FROM-Y &OPTIONAL (FONT (SHEET-CURRENT-FONT SELF))
                                        (ALU CHAR-ALUF)
                                        (TOWARD-X (1+ FROM-X))
                                        (TOWARD-Y FROM-Y)
                                        (STRETCH-P NIL))
 (send self :draw-string
             STRING FROM-X FROM-Y TOWARD-X TOWARD-Y STRETCH-P FONT ALU))


USER:

(DEFUN SET-WINDOW (W)
   (SEND W :EXPOSE)
   (SEND W :REFRESH)
   (SEND BOXES :SET-WINDOW W)
   (SEND C BOXES :SET-WINDOW W)
   (SEND LINES :SET-WINDOW W)
   (SEND DASHED-LINES :SET-WINDOW W)
   (SEND STRINGS :SET-WINDOW W)
   (SEND BOXES :SHOW-OBJECTS)
   (SEND C-BOXES :SHOW-OBJECTS)
   (SEND LINES :SHOW-OBJECTS)
   (SEND DASHED-LINES :SHOW OBJECTS)
   (SEND STRINGS :SHOW-OBJECTS))

(DEFUN HC (W)
   (SET-WINDOW HC-WINDOW)
   (SET-WINDOW W)
   (PROCESS-RUN-TEMPORARY-FUNCTION "OHP-OUT" 'TV:HARDCOPY HC-WINDOW))
```

;;;;;;;;;;;;;;;;;;;; SAVE OBJECT ;;;;;;;;;;;;;;;;;;;;

```
(DEFUN SAVE-BOX ()
 (SETQ BOX-SAVE-LIST ())
```

```
  (IF (EQ BOX-EDGE-ALIST ()) ()
      (DOTIMES (I (LENGTH BOX-EDGE-ALIST))
       (LET ((BOX (NTH I (REVERSE BOX-EDGE-ALIST))))
        (POP BOX)
        (PUSH BOX BOX-SAVE-LIST)))))

(DEFUN SAVE-C-BOX ()
 (SETQ C-BOX-SAVE-LIST ())
 (IF (EQ C-BOX-EDGE-ALIST ()) ()
     (DOTIMES (I (LENGTH C-BOX-EDGE-ALIST))
      (LET ((C-BOX (NTH I (REVERSE C-BOX-EDGE-ALIST))))
       (POP C-BOX)
       (PUSH C-BOX C-BOX-SAVE-LIST)))))

(DEFUN SAVE-LINE ()
 (SETQ LINE-SAVE-LIST ())
 (IF (EQ LINE-EDGE-ALIST ()) ()
     (DOTIMES (I (LENGTH LINE-EDGE-ALIST))
      (LET ((LINE (NTH I (REVERSE LINE-EDGE-ALIST))))
       (POP LINE)
       (PUSH LINE LINE-SAVE-LIST)))))

(DEFUN SAVE-DASHED-LINE ()
 (SETQ DASHED-LINE-SAVE-LIST ())
 (IF (EQ DASHED-LINE-EDGE-ALIST ()) ()
     (DOTIMES (I (LENGTH DASHED-LINE-EDGE-ALIST))
      (LET ((DASHED-LINE (NTH I (REVERSE DASHED-LINE-EDGE-ALIST))))
       (POP DASHED-LINE)
       (PUSH DASHED-LINE DASHED-LINE-SAVE-LIST)))))

(DEFUN SAVE-STRING ()
 (SETQ STRING-SAVE-LIST ())
 (IF (EQ STRING-EDGE-ALIST ()) ()
     (DOTIMES (I (LENGTH STRING-EDGE-ALIST))
      (LET ((STRING (NTH I (REVERSE STRING-EDGE-ALIST))))
       (LET ((OBJ (POP STRING)))
        (LET ((SS (SEND OBJ ':STRING)))
         (LET ((FONT (TV:FONT-NAME (SEND OBJ ':FONT))))
       (PUSH FONT STRING)
       (PUSH SS STRING)
       (PUSH STRING STRING-SAVE-LIST)))))))

(DEFUN SAVE-OHP (&REST IGNORE)
 (PROG () LOOP-FILE
     (COND ((EQ (ERRSET
              (LET ((FILE-NAME (READ-FILE-NAME)))
               (SAVE-BOX)
               (SAVE-C-BOX)
               (SAVE-LINE)
               (SAVE-STRING)
               (SAVE-DASHED-LINE)
               (WITH-OPEN-FILE (SAVE FILE-NAME ':OUT)
                (PRINT (LIST BOX-SAVE-LIST
                             C-BOX-SAVE-LIST
                             LINE SAVE-LIST
                             STRING-SAVE-LIST
                             DASHED-LINE-SAVE-LIST) SAVE))) ()) NIL)
         (BEEP)
         (GO LOOP-FILE)))))

(DEFUN LOAD-OHP (W &REST IGNORE)
 (PROG () LOOP-FILE
     (COND ((EQ (ERRSET
              (LET ((FILE-NAME (READ-FILE-NAME)))
               (WITH-OPEN-FILE (LOAD FILE-NAME ':IN)
                (SETQ LOAD-LIST (READ LOAD)))
               (LOAD-BOX (FIRST LOAD-LIST))
               (LOAD-C-BOX (SECOND LOAD-LIST))
               (LOAD-LINE (THIRD LOAD-LIST))
               (LOAD-STRING (FOURTH LOAD-LIST))
               (IF (> (LENGTH LOAD-LIST) 4)
                   (LOAD-DASHED-LINE (fifth load-list))
                   (SETQ DASHED-LINE-EDGE-ALIST ())
                   (SEND DASHED-LINES ':SET-OBJECTS-LIST ())) ()) NIL)
         (BEEP)
         (GO LOOP-FILE)))
    (SEND W ':SELECT)
    (REDISPLAY W)))

(DEFUN LOAD-BOX (BOX-LOAD-LIST)
 (SETQ BOX-EDGE-ALIST ())
```

```
 (SEND BOXES ':SET-OBJECTS-LIST ())
 (DOLIST (B (REVERSE BOX-LOAD-LIST))
  (LET ((BOX (MAKE-INSTANCE 'G-BOX)))
    (SEND BOX ':SET-LEFT (FIRST B))
    (SEND BOX ':SET-TOP (SECOND B))
    (SEND BOX ':SET-RIGHT (THIRD B))
    (SEND BOX ':SET-BOTTOM (FOURTH B))
    (SEND BOXES ':ADD-OBJECT BOX)
    (PUSH (PUSH BOX B) BOX-EDGE-ALIST))))

(DEFUN LOAD-C-BOX (C-BOX-LOAD-LIST)
 (SETQ C-BOX-EDGE-ALIST ())
 (SEND C-BOXES ':SET-OBJECTS-LIST ())
 (DOLIST (C-B (REVERSE C-BOX-LOAD-LIST))
  (LET ((C-BOX (MAKE-INSTANCE 'C-BOX)))
    (SEND C-BOX ':SET-LEFT (FIRST C-B))
    (SEND C-BOX ':SET-TOP (SECOND C-B))
    (SEND C-BOX ':SET-RIGHT (THIRD C-B))
    (SEND C-BOX ':SET-BOTTOM (FOURTH C-B))
    (SEND C-BOXES ':ADD-OBJECT C-BOX)
    (PUSH (PUSH C-BOX C-B) C-BOX-EDGE-ALIST))))

(DEFUN LOAD-LINE (LINE-LOAD-LIST)
 (SETQ LINE-EDGE-ALIST ())
 (SEND LINES ':SET-OBJECTS-LIST ())
 (DOLIST (L (REVERSE LINE-LOAD-LIST))
  (LET ((LINE (MAKE-INSTANCE 'LINE)))
    (SEND LINE ':SET-START-X (FIRST L))
    (SEND LINE ':SET-START-Y (SECOND L))
    (SEND LINE ':SET-END-X (THIRD L))
    (SEND LINE ':SET-END-Y (FOURTH L))
    (SEND LINES ':ADD-OBJECT LINE)
    (PUSH (PUSH LINE L) LINE-EDGE-ALIST))))

(DEFUN LOAD-DASHED-LINE (DASHED-LINE-LOAD-LIST)
 (SETQ DASHED-LINE-EDGE-ALIST ())
 (SEND DASHED-LINES ':SET-OBJECTS-LIST ())
 (DOLIST (L (REVERSE DASHED-LINE-LOAD-LIST))
  (LET ((DASHED-LINE (MAKE-INSTANCE 'DASHED-LINE)))
    (SEND DASHED-LINE ':SET-START-X (FIRST L))
    (SEND DASHED-LINE ':SET-START-Y (SECOND L))
    (SEND DASHED-LINE ':SET-END-X (THIRD L))
    (SEND DASHED-LINE ':SET-END-Y (FOURTH L))
    (SEND DASHED-LINES ':ADD-OBJECT DASHED-LINE)
    (PUSH (PUSH DASHED-LINE L) DASHED-LINE-EDGE-ALIST))))

(DEFUN LOAD-STRING (STRING-LOAD-LIST)
 (SETQ STRING-EDGE-ALIST ())
 (SEND STRINGS ':SET-OBJECTS-LIST ())
 (DOLIST (S (REVERSE STRING-LOAD-LIST))
  (LET ((STRING (MAKE-INSTANCE 'G-STRING)))
    (SEND STRING ':SET-STRING (POP S))
    (SEND STRING ':SET-FONT (EVAL (POP S)))
    (SEND STRING ':SET-LEFT (FIRST S))
    (SEND STRING ':SET-TOP (SECOND S))
    (SEND STRING ':SET-RIGHT (THIRD S))
    (SEND STRING ':SET-BOTTOM (FOURTH S))
    (SEND STRINGS ':ADD-OBJECT STRING)
    (PUSH (PUSH STRING S) STRING-EDGE-ALIST))))

(DEFUN READ-FILE-NAME ()
 (PROG ()
     (SELECT-FILE-WINDOW)
     (LET ((FILE-NAME (FS:MERGE-PATHNAME-DEFAULTS (READ FILE-WINDOW))))
       (SEND FILE-WINDOW ':BURY)
       (RETURN FILE-NAME))))


;;;;;;;;;;;;;;;;;;;;  SAVE/LOAD MULTIPLE  ;;;;;;;;;;;;;;;;;;;;

(DEFUN NEXT (W &REST IGNORE)
 (SAVE-BOX)
 (SAVE-C-BOX)
 (SAVE-LINE)
 (SAVE-STRING)
 (SAVE-DASHED-LINE)
 (OR (AND (EQ BOX-SAVE-LIST ())
          (EQ C-BOX-SAVE-LIST ())
          (EQ LINE-SAVE-LIST ())
          (EQ STRING-SAVE-LIST ())
          (EQ DASHED-LINE-SAVE-LIST ()))
```

```
      (PUSH (LIST BOX SAVE-LIST C-BOX-SAVE-LIST LINE-SAVE-LIST
                  STRING-SAVE-LIST DASHED-LINE-SAVE-LIST) SHEETS2))
    (IF (EQ SHEETS1 ())
        (NEW)
        (LET ((LOAD-LIST (POP SHEETS1)))
          (LOAD-BOX (FIRST LOAD-LIST))
          (LOAD-C-BOX (SECOND LOAD-LIST))
          (LOAD-LINE (THIRD LOAD-LIST))
          (LOAD-STRING (FOURTH LOAD-LIST))
          (LOAD-DASHED-LINE (FIFTH LOAD-LIST))))
    (REDISPLAY W))

  (DEFUN NEW ()
    (SETQ BOX-EDGE-ALIST ()
          C-BOX-EDGE-ALIST ()
          LINE-EDGE-ALIST ()
          STRING-EDGE-ALIST ()
          DASHED-LINE-EDGE-ALIST ())
    (SEND BOXES ':SET-OBJECTS-LIST ())
    (SEND C-BOXES ':SET-OBJECTS-LIST ())
    (SEND LINES ':SET-OBJECTS-LIST ())
    (SEND STRINGS ':SET-OBJECTS-LIST ())
    (SEND DASHED-LINES ':SET OBJECTS-LIST ()))

  (DEFUN PREVIOUS (W &REST IGNORE)
    (SAVE-BOX)
    (SAVE-C-BOX)
    (SAVE-LINE)
    (SAVE-STRING)
    (SAVE-DASHED-LINE)
    (IF (EQ SHEETS2 ())
        (BEEP)
        (LET ((LOAD-LIST (POP SHEETS2)))
          (OR (AND (EQ BOX-SAVE-LIST ())
                   (EQ C-BOX-SAVE-LIST ())
                   (EQ LINE-SAVE-LIST ())
                   (EQ STRING-SAVE LIST ())
                   (EQ DASHED-LINE-SAVE-LIST ()))
              (PUSH (LIST BOX-SAVE-LIST
                          C-BOX SAVE-LIST
                          LINE-SAVE-LIST
                          STRING-SAVE-LIST
                          DASHED-LINE-SAVE-LIST) SHEETS1))
          (LOAD BOX (FIRST LOAD-LIST))
          (LOAD-C-BOX (SECOND LOAD-LIST))
          (LOAD-LINE (THIRD LOAD-LIST))
          (LOAD STRING (FOURTH LOAD-LIST))
          (LOAD-DASHED-LINE (FIFTH LOAD-LIST))))
    (REDISPLAY W))

  (DEFUN LOAD-OHP-SHEETS (W &REST IGNORE)
    (PROG () LOOP-FILE
          (COND ((EQ (ERRSET
                       (LET ((FILE NAME (READ-FILE-NAME)))
                         (WITH-OPEN-FILE (LOAD FILE-NAME ':IN)
                           (SETQ SHEETS1 (READ LOAD)))) ()) NIL)
                (BEEP)
                (GO LOOP-FILE))))
    (CHECK-DASHED LINE)
    (SETQ SHEETS2 ())
    (LET ((LOAD-LIST (POP SHEETS1)))
      (LOAD-BOX (FIRST LOAD-LIST))
      (LOAD-C-BOX (SECOND LOAD-LIST))
      (LOAD-LINE (THIRD LOAD-LIST))
      (LOAD-STRING (FOURTH LOAD-LIST))
      (LOAD-DASHED-LINE (FIFTH LOAD-LIST)))
    (SEND W ':SELECT)
    (REDISPLAY W))

(DEFVAR DUMMY-LIST)

(DEFUN CHECK-DASHED-LINE ()
  (IF (> (LENGTH (FIRST SHEETS1)) 4)
      ()
      (SETQ DUMMY-LIST ())
      (DOLIST (LOAD-LIST SHEETS1)
        (LET ((LIST (REVERSE LOAD-LIST)))
          (PUSH (REVERSE (PUSH () LIST)) DUMMY-LIST)))
      (SETQ SHEETS1 (REVERSE DUMMY-LIST))))

(DEFUN SAVE-OHP-SHEETS (W &REST IGNORE)
```

```
(PROG () LOOP-FILE
    (COND ((EQ (ERRSET
            (LET ((FILE-NAME (READ-FILE-NAME)))
             (SAVE-BOX)
             (SAVE-C-BOX)
             (SAVE-LINE)
             (SAVE-STRING)
             (SAVE-DASHED-LINE)
             (WITH-OPEN-FILE (SAVE FILE-NAME ':OUT)
              (PRINT (APPEND (REVERSE SHEETS2)
                        (LIST (LIST BOX-SAVE-LIST
                                    C-BOX-SAVE-LIST
                                    LINE-SAVE-LIST
                                    STRING-SAVE-LIST
                                    DASHED-LINE-SAVE-LIST))
                        SHEETS1) SAVE))) ()) NIL)
          (BEEP)
          (GO LOOP-FILE)))))

;;;;;;;;;;;;;;;;;;;;;;;;; DASHED LINE ;;;;;;;;;;;;;;;;;;;;;;;;;

(DEFFLAVOR DASHED-LINE
     (START-X START-Y END-X END-Y (WINDOW TERMINAL-IO))
     ()
 (:INITABLE-INSTANCE-VARIABLES)
 (:GETTABLE-INSTANCE-VARIABLES)
 (:SETTABLE-INSTANCE-VARIABLES))

(DEFMETHOD (DASHED-LINE :SHOW) ()
 (SEND WINDOW ':DRAW-DASHED-LINE START-X START-Y END-X END-Y tv:alu-xor 10. t 0 8.))

(DEFUN C-DASHED-LINE (W)
 (LET ((DASHED-LINE (MAKE-DASHED-LINE W)))
  (PUSH (LIST DASHED-LINE (SEND DASHED-LINE ':START-X)
              (SEND DASHED-LINE ':START-Y)
              (SEND DASHED-LINE ':END-X)
              (SEND DASHED-LINE ':END-Y))
       DASHED-LINE-EDGE-ALIST)))

(DEFUN MAKE-DASHED-LINE (W &OPTIONAL (CHAR 7))
 (PROG ()
 (TV:WITH-MOUSE-GRABBED
  (tv:mouse-warp 250. 250.)
  (PROCESS-WAIT "BUTTON UP" #'(LAMBDA () (ZEROP TV:MOUSE-LAST-BUTTONS)))
  (TV:MOUSE-SET-BLINKER-DEFINITION ':CHARACTER 0 0 ':ON
                        ':SET-CHARACTER CHAR)
  (PROCESS-WAIT "BUTTON DOWN" #'(LAMBDA () (NOT (ZEROP TV:MOUSE-LAST-BUTTONS))))
  (SETQ X TV:MOUSE-X Y TV:MOUSE-Y)
  (BEEP)
  (PROCESS-WAIT "BUTTON UP" #'(LAMBDA () (ZEROP TV:MOUSE-LAST-BUTTONS)))
  (TV:MOUSE-SET-BLINKER-DEFINITION ':CHARACTER 0 0 ':ON
                        ':SET-CHARACTER CHAR)
  (PROCESS-WAIT "BUTTON DOWN" #'(LAMBDA () (NOT (ZEROP TV:MOUSE-LAST-BUTTONS))))
  (SETQ XX TV:MOUSE-X YY TV:MOUSE-Y)
  (LET ((ADASHED-LINE (MAKE-INSTANCE 'DASHED-LINE ':START-X (1+ X)
                ':START-Y (+ 2 Y)
                ':END-X (1+ XX)
                ':END-Y (+ 2 YY)
                ':WINDOW W)))
   (SEND ADASHED-LINE ':SHOW)
   (SEND DASHED-LINES ':ADD-OBJECT ADASHED-LINE)
   (RETURN ADASHED-LINE)))))

(SETQ DASHED-LINES (MAKE-INSTANCE 'OBJECTS))

;;;;;;;;;;;;;;;;;;;;;;;;; VERTICAL/HORIZONTAL DASHED-LINE ;;;;;;;;;;;;;;;;;;;;;;;;;

(DEFUN V-DASHED-LINE (W)
 (LET ((DASHED-LINE (MAKE-V-DASHED-LINE W)))
  (PUSH (LIST DASHED-LINE (SEND DASHED-LINE ':START-X)
              (SEND DASHED-LINE ':START-Y)
              (SEND DASHED-LINE ':END-X)
              (SEND DASHED-LINE ':END-Y))
       DASHED-LINE-EDGE-ALIST)))

(DEFUN MAKE-V-DASHED-LINE (W &OPTIONAL (CHAR 7))
 (PROG ()
 (TV:WITH-MOUSE-GRABBED
  (TV:MOUSE-WARP 250. 250.)
```

```
    (PROCESS-WAIT "BUTTON UP" #'(LAMBDA () (ZEROP TV:MOUSE-LAST-BUTTONS)))
    (TV:MOUSE-SET-BLINKER-DEFINITION ':CHARACTER 0 0 ':ON
                                     ':SET-CHARACTER CHAR)
    (PROCESS-WAIT "BUTTON DOWN" #'(LAMBDA () (NOT (ZEROP TV:MOUSE-LAST-BUTTONS))))
    (SETQ X TV:MOUSE-X Y TV:MOUSE-Y)
    (BEEP)
    (PROCESS-WAIT "BUTTON UP" #'(LAMBDA () (ZEROP TV:MOUSE-LAST-BUTTONS)))
    (TV:MOUSE-SET-BLINKER-DEFINITION ':CHARACTER 0 0 ':ON
                                     ':SET-CHARACTER CHAR)
    (PROCESS-WAIT "BUTTON DOWN" #'(LAMBDA () (NOT (ZEROP TV:MOUSE-LAST-BUTTONS))))
    (SETQ YY TV:MOUSE-Y)
    (LET ((ADASHED-LINE (MAKE-INSTANCE 'DASHED-LINE ':START-X (1+ X)
                       ':START-Y (+ 2 Y)
                       ':END-X (1+ X)
                       ':END-Y (+ 2 YY)
                       ':WINDOW W)))
     (SEND ADASHED-LINE ':SHOW)
     (SEND DASHED-LINES ':ADD-OBJECT ADASHED-LINE)
     (RETURN ADASHED-LINE)))))

(DEFUN H-DASHED-LINE (W)
  (LET ((DASHED-LINE (MAKE-H-DASHED-LINE W)))
    (PUSH (LIST DASHED LINE (SEND DASHED-LINE ':START-X)
                (SEND DASHED-LINE ':START-Y)
                (SEND DASHED-LINE ':END-X)
                (SEND DASHED-LINE ':END-Y))
          DASHED-LINE-EDGE-ALIST)))

(DEFUN MAKE-H-DASHED-LINE (W &OPTIONAL (CHAR 7))
  (PROG ()
  (TV:WITH MOUSE-GRABBED
    (tv:mouse-warp 250. 250.)
    (PROCESS-WAIT "BUTTON UP" #'(LAMBDA () (ZEROP TV:MOUSE-LAST-BUTTONS)))
    (TV:MOUSE-SET-BLINKER-DEFINITION ':CHARACTER 0 0 ':ON
                                     ':SET-CHARACTER CHAR)
    (PROCESS-WAIT "BUTTON DOWN" #'(LAMBDA () (NOT (ZEROP TV:MOUSE-LAST-BUTTONS))))
    (SETQ X TV:MOUSE-X Y TV:MOUSE-Y)
    (BEEP)
    (PROCESS-WAIT "BUTTON UP" #'(LAMBDA () (ZEROP TV:MOUSE-LAST-BUTTONS)))
    (TV:MOUSE-SET-BLINKER-DEFINITION ':CHARACTER 0 0 ':ON
                                     ':SET-CHARACTER CHAR)
    (PROCESS-WAIT "BUTTON DOWN" #'(LAMBDA () (NOT (ZEROP TV:MOUSE-LAST-BUTTONS))))
    (SETQ XX TV:MOUSE-X)
    (LET ((ADASHED-LINE (MAKE-INSTANCE 'DASHED-LINE ':START-X (1+ X)
                       ':START-Y (+ 2 Y)
                       ':END-X (1+ XX)
                       ':END-Y (+ 2 Y)
                       ':WINDOW W)))
     (SEND ADASHED-LINE ':SHOW)
     (SEND DASHED-LINES ':ADD-OBJECT ADASHED-LINE)
     (RETURN ADASHED-LINE)))))

(DEFUN BOX-EDITOR-FIND-DASHED-LINE (&OPTIONAL (DEL-F ())(CHAR 24))
  (SETQ DASHED-LINE NIL)
  (TV:WITH-MOUSE-GRABBED
    (tv:mouse-warp 300. 300.)
    (PROCESS-WAIT "BUTTON UP" #'(LAMBDA () (ZEROP TV:MOUSE-LAST-BUTTONS)))
    (TV:MOUSE-SET-BLINKER-DEFINITION ':CHARACTER 0 0 ':ON
                                     ':SET-CHARACTER CHAR)
    (PROCESS-WAIT "BUTTON DOWN" #'(LAMBDA () (NOT (ZEROP TV:MOUSE-LAST-BUTTONS))))
    (SETQ X TV:MOUSE-X Y TV:MOUSE-Y)
    (AND (BIT-TEST 1 TV:MOUSE-LAST-BUTTONS)
       (DOLIST (L DASHED LINE-EDGE-ALIST)
         (LET ((X1 (SECOND L))
               (Y1 (THIRD L))
               (X2 (FOURTH L))
               (Y2 (FIFTH L)))
           (and (> 0.1 (abs (- (atan (- y1 y2) (- x1 x2))
                              (atan (- y y2) (- x x2)))))
                (> 0.1 (abs (- (atan (- y2 y1) (- x2 x1))
                              (atan (- y y1) (- x x1)))))
                (OR (EQ DEL-F ())
                   (setq DASHED-line-edge-alist
                                 (DELETE L DASHED-LINE-EDGE-ALIST)) t)
                (RETURN (SETQ DASHED-LINE (FIRST L)))))))))
  DASHED LINE)

(DEFUN D-DASHED-LINE (W)
  (OR (SEND DASHED-LINES ':DELETE-OBJECT (BOX-EDITOR-FIND-DASHED-LINE T))
      (BEEP))
  (redisplay w))
```

```
tv:
(DEFMETHOD (GRAPHICS-MIXIN :DRAW-DASHED-LINE)
        (FROM-X FROM-Y TO-X TO-Y
                &OPTIONAL (ALU CHAR-ALUF) (DASH-SPACING 20.)
                RELAXED-P (OFFSET 0) (DASH-LENGTH (// DASH-SPACING 2)))
  (CHECK-ARG-TYPE ALU :FIXNUM)
  (SETQ FROM-X (+ FROM-X (SHEET-INSIDE-LEFT))
        FROM-Y (+ FROM-Y (SHEET-INSIDE-TOP))
        TO-X (+ TO-X (SHEET-INSIDE-LEFT))
        TO-Y (+ TO-Y (SHEET-INSIDE-TOP)))
  (LET* ((LWIDTH (- TO-X FROM-X))
         (LHEIGHT (- TO-Y FROM-Y))
         (LENGTH (ISQRT (+ (^ LWIDTH 2) (^ LHEIGHT 2)))))
    (if (eq length 0) ()
    (COND ((NOT RELAXED-P)
           ;; We want to draw line segments such that the last one ends up
           ;; finishing at the end-point.  Therefore, separate the entire
           ;; line length into an odd number of regions, each half of the
           ;; dash-spacing, and fill in every other one with a line segment.
           ;; Just use dash-spacing to decide how many segments, so the
           ;; spacing may not be quite right.
           (LET* ((N-HALF-SEGS (LOGIOR 1 (// LENGTH (// DASH-SPACING 2))))
                  (X-LEN (// (FLOAT LWIDTH) N-HALF-SEGS))
                  (Y-LEN (// (FLOAT LHEIGHT) N-HALF-SEGS)))
             (DRAW-DASHED-LINE-INTERNAL
               FROM-X FROM-Y
               (* 2 X-LEN) (* 2 Y-LEN)
               X-LEN Y-LEN
               (ASH N-HALF-SEGS -1) ALU)))
          (T
           ;; Don't worry about where it ends.  Compute the sine and cosine and
           ;; just draw dashes of the exact specified size.
          (let* ((sin (// (float lheight) (float length)))
                 (cos (// (float lwidth) (float length))))
            (DRAW-DASHED-LINE-INTERNAL-1
              (+ FROM-X (* OFFSET cos))
              (+ FROM-Y (* OFFSET sin))
              (+ to-x (* offset cos))
              (+ to-y (* offset sin))
              (* DASH-SPACING cos)
              (* DASH-SPACING sin)
              (* DASH-LENGTH cos)
              (* DASH-LENGTH sin)
              (// (- LENGTH (FIXr OFFSET)) (FIXr DASH-SPACING)) ALU)
            ))))))
tv:
(DEFUN-METHOD DRAW-DASHED-LINE-INTERNAL-1 GRAPHICS-MIXIN
        (START-X START-Y end-x end-y X-SPACING Y-SPACING X-LENGTH Y-LENGTH N-SEGS ALU)
  (PREPARE-SHEET (SELF)
    (LOOP FOR X FROM (FLOAT START-X) BY X-SPACING
          FOR Y FROM (FLOAT START-Y) BY Y-SPACING
          FOR I FROM 0 TO N-SEGS DO
          (if (eq i n-segs)
              (%draw-line (fix x) (fix y) (fix end-x) (fix end-y) alu nil self)
          (%DRAW-LINE (FIX X) (FIX Y) (FIX (+ X X-LENGTH)) (FIX (+ Y Y-LENGTH))
                      ALU NIL SELF)))))
```