File created: 9-Mar-88 15:54:25 **{IVY}<HOGG>LISP>MEDLEY>PRESSFROMNS.;13**

changes to: (VARS PRESSFROMNSCOMS)
(FNS \CREATECHARSET.PRESS \CREATECHARSETZERO.PRESS \CREATEPRESSFONT \COERCEFONT)
(RECORDS PRESSDATA)

previous date: 4-Mar-88 12:52:46 {IVY}<HOGG>LISP>MEDLEY>PRESSFROMNS.;9

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

;;


```
(RPAQQ PRESSFROMNSCOMS
       [(* This file uses CONSTANTS defined in PRESS, so it is necessary to LOADFROM PRESS before changing this
          file.)
        (FNS \SMASHPRESSFONTS)
        (FNS GETCHARPRESSTRANSLATION PRESS.NSARRAY PUTCHARPRESSTRANSLATION)
        (FNS \DSPFONT.PRESS \DSPSPACEFACTOR.PRESS \ENTITYSTART.PRESS \SETSPACE.PRESS \STARTPAGE.PRESS
             \PRESS.COERCEFONT \DSPFONT.PRESSFONT SETUPFONTS.PRESS)
        (FNS \CREATEPRESSFONT \CREATECHARSET.PRESS \CREATECHARSETZERO.PRESS)
        (FNS \PRESSCURVE2)
        (COMS (* Generic utility for coercing fonts, could be used by other devices)
              (FNS \COERCEFONT))
        (ALISTS (FONTCOERCIONS PRESS)
               (MISSINGFONTCOERCIONS PRESS))
        (GLOBALVARS FONTCOERCIONS MISSINGFONTCOERCIONS)
        (FNS \STRINGWIDTH.PRESS \CHARWIDTH.PRESS \OUTCHARFN.PRESS)
        (* * new declaration for PRESSDATA)
        (DECLARE%: DONTCOPY (RECORDS PRESSDATA))
        (INITRECORDS PRESSDATA)
        (* * NSTOASCIITRANSLATIONS is a list with elements of the form (charset translationArrayName)
           %, where translationArrayName is bound to a translation array for charset which contains (fontFamily
                                                                                                     charcode)
           lists)
        (FNS \NSTOASCIIARRAY \NSTOASCIITRANSLATION)
        (GLOBALVARS NSTOASCIITRANSLATIONS PRESSFONTFAMILIES)
        [INITVARS (PRESSFONTFAMILIES '((GACHA)
                                       (TIMESROMAN)
                                       (HELVETICA)
                                       (SYMBOL)
                                       (MATH)
                                       (HIPPO)
                                       (CYRILLIC)
                                       (NEWVEC)
                                       (SNEWVEC)
                                       (HNEWVEC)
                                       (VNEWVEC]
        (INITVARS (NSTOASCIITRANSLATIONS))
        (ADDVARS (NSTOASCIITRANSLATIONS (0 ASCIIFROM0ARRAY)
                       (38 ASCIIFROM38ARRAY)
                       (39 ASCIIFROM39ARRAY)
                       (239 ASCIIFROM239ARRAY)))
        (UGLYVARS ASCIIFROM0ARRAY ASCIIFROM38ARRAY ASCIIFROM39ARRAY ASCIIFROM239ARRAY)
        (P (\SMASHPRESSFONTS))
        (DECLARE%: DONTCOPY (CONSTANTS (unknownCharTranslation '(MATH 59])
```

(* * This file uses CONSTANTS defined in PRESS, so it is necessary to LOADFROM PRESS before changing this file.)

```
(DEFINEQ

(\SMASHPRESSFONTS
  [LAMBDA NIL                                                    ; Edited 29-Feb-88 10:21 by thh:
    ;; Executed after all patchfns have been loaded, coerces existing Koto press fonts into NS-type press fonts
    (for F in (FONTSAVAILABLE '* '* '* '* 'PRESS) do (\CREATECHARSET 0 (FONTCREATE F])

)

(DEFINEQ

(GETCHARPRESSTRANSLATION
  [LAMBDA (CHARCODE FONT)                                        (* thh%: "28-Feb-86 12:03")
                                                                 (* returns the Press translation for a character in a font)
    (COND
       ((OR (CHARCODEP CHARCODE)
            (EQ CHARCODE 256))                                   (* bitmap for char 256 is what gets printed if char not found)
        )
       ((OR (STRINGP CHARCODE)
            (LITATOM CHARCODE))
        (SETQ CHARCODE (CHCON1 CHARCODE)))
```

```
       (T (\ILLEGAL.ARG CHARCODE)))
    (LET [TR CSINFO (FONTDESC (\GETFONTDESC FONT 'PRESS]          (* fetch the csinfo for the character set of this character.)
         (SETQ CSINFO (\GETCHARSETINFO (\CHARSET CHARCODE)
                                FONTDESC))
         (SETQ TR (\GETBASEPTR (fetch (CHARSETINFO CHARSETBITMAP) of CSINFO)
                        (UNFOLD (\CHAR8CODE CHARCODE)
                                2)))                              (* Return a copy)
         (LIST (CAR TR)
               (CDR TR])
```

(PRESS.NSARRAY

```
  [LAMBDA (CHARSET FAMILY ASCIIARRAY)                             (* thh%: "28-Feb-86 12:08")
                                                                  (* using info in ASCIIARRAY or ASCIITONSTRANSLATIONS,
                                                                  creates an array of (pressFont charcode) lists)
    (LET* ((min (TIMES 256 CHARSET))
           (max (PLUS min 255))
           (array (ARRAY 256 NIL NIL 0)))
          [for item in (COND
                          [ASCIIARRAY `((%, FAMILY ASCIIARRAY]
                          (T ASCIITONSTRANSLATIONS))
             bind asciiArray do

          (* * item is of the form (PressFont TranslationArray NSFont))

                                (SETQ asciiArray (EVAL (CADR item)))
                                (COND
                                   (asciiArray (for i from 0 to 255 do (SETA array (REMAINDER (ELT asciiArray i)
                                                                                              256)
                                                                                      (LIST (CAR item)
                                                                                            i))
                                               when (AND (LEQ min (ELT asciiArray i))
                                                         (LEQ (ELT asciiArray i)
                                                              max]
          array])
```

(PUTCHARPRESSTRANSLATION

```
  [LAMBDA (CHARCODE FONT NEWTRANSLATION)                          ; Edited 29-Feb-88 10:28 by thh:
                                                                  ; Changes the Press translation for a character in a font
    (COND
       ((CHARCODEP CHARCODE))
       ((OR (STRINGP CHARCODE)
            (LITATOM CHARCODE))
        (SETQ CHARCODE (CHCON1 CHARCODE)))
       (T (\ILLEGAL.ARG CHARCODE)))
    (PROG* ((FONTDESC (\GETFONTDESC FONT 'PRESS))
            (CSINFO (\GETCHARSETINFO (\CHARSET CHARCODE)
                           FONTDESC))
            (CHAR8CODE (\CHAR8CODE CHARCODE))
            (TR (\NSTOASCIITRANSLATION NEWTRANSLATION NIL FONTDESC)))
           (UNINTERRUPTABLY
               (\RPLPTR (ffetch (CHARSETINFO CHARSETBITMAP) of CSINFO)
                      (UNFOLD CHAR8CODE 2)
                      TR)
               (\PUTBASE (ffetch (CHARSETINFO WIDTHS) of CSINFO)
                      CHAR8CODE
                      (\FGETCHARWIDTH (CAR TR)
                             (CDR TR)))
               [change (ffetch CHARSETASCENT of CSINFO)
                      (MAX DATUM (ffetch \SFAscent of (CAR TR]
               [change (ffetch CHARSETDESCENT of CSINFO)
                      (MAX DATUM (ffetch \SFDescent of (CAR TR]
               [freplace \SFHeight of FONTDESC with (PLUS (change (ffetch \SFAscent of FONTDESC)
                                                                  (MAX DATUM (ffetch CHARSETASCENT of CSINFO)))
                                                          (change (ffetch \SFDescent of FONTDESC)
                                                                  (MAX DATUM (ffetch CHARSETDESCENT of CSINFO])
           (RETURN NEWTRANSLATION])

)

(DEFINEQ
```

(\DSPFONT.PRESS

```
  [LAMBDA (PRSTREAM FONT)                                         (* rmk%: "25-Feb-86 11:05")

          (* * The DSPFONT method for PRESS-type image streams -- change the stream's current logical font to FONT;
          the device font changes only when we print a character)

    (PROG (OLDFONT FDENTRY (PRDATA (ffetch IMAGEDATA of PRSTREAM)))
          (SETQ OLDFONT (ffetch PRLOGICALFONT of PRDATA))
          (COND
             ([OR (NULL FONT)
                  (EQ OLDFONT (SETQ FONT (OR (\GETFONTDESC FONT 'PRESS T)
                                             (FONTCOPY OLDFONT FONT]
              (RETURN OLDFONT)))
          (freplace PRLOGICALFONT of PRDATA with FONT)
```

```
        (freplace PRLOGICALCHARSET of PRDATA with NIL)
        [\SETSPACE.PRESS PRSTREAM (FIXR (TIMES (ffetch PRSPACEFACTOR of PRDATA)
                                               (\FGETCHARWIDTH FONT (CHARCODE SPACE]
        [freplace PRLINEFEED of PRDATA with (IDIFFERENCE (CONSTANT (IMINUS MicasPerPoint))
                                                         (FONTPROP FONT 'HEIGHT]
        (\FIXLINELENGTH.PRESS PRSTREAM)
        (RETURN OLDFONT])
```

(\DSPSPACEFACTOR.PRESS

```
[LAMBDA (STREAM FACTOR)                                          (* rmk%: "24-Feb-86 09:49")
  (LET ((PRDATA (ffetch IMAGEDATA of STREAM)))
       (PROG1 (ffetch PRSPACEFACTOR of PRDATA)
           [COND
              (FACTOR (SHOW.PRESS STREAM)
                      (freplace PRSPACEFACTOR of PRDATA with FACTOR)
                      (\SETSPACE.PRESS STREAM (FIXR (TIMES FACTOR (\FGETCHARWIDTH (ffetch PRLOGICALFONT
                                                                                    of PRDATA)
                                                                  (CHARCODE SPACE])])
```

(\ENTITYSTART.PRESS

```
[LAMBDA (PRSTREAM)                                               (* thh%: "10-Dec-86 08:33")
  (PROG ((PRDATA (fetch IMAGEDATA of PRSTREAM)))
        (freplace PRSPACEWIDTH of PRDATA with NIL)

        (* This really should be the spacewidth of the current font. But then, if we switch fonts to one whose space*spacefactor
        comes out the same, we won't know to put out a setspace command.
        So when we actually set up the first font in this entity, we will end up putting out an explicit setspace
        (even if the space factor is 1))

        (freplace PRFONT of PRDATA with NIL)
        (freplace PRLOGICALFONT of PRDATA with NIL)

        (* We set the font to NIL, knowing that the current font can be recoverd from the PRCURRFDE.
        This font will be set in the press file before the first show, if no explicit dspfont intervenes.
        Note, however, that up until the first dspfont, the widthscache still corresponds to what was the PRLOGICALFONT)

        (freplace DLSTARTBYTE of PRDATA with (\GETFILEPTR PRSTREAM))
        (freplace ELSTARTBYTE of PRDATA with (\GETFILEPTR (fetch ELSTREAM of PRDATA)))
        (freplace STARTCHARBYTE of PRDATA with (\GETFILEPTR PRSTREAM))
                                                                 (* Entity starts with position at 0,0 so must re-establish current
                                                                 position (?))
        (SETXY.PRESS PRSTREAM (fetch PRXPOS of PRDATA)
               (fetch PRYPOS of PRDATA])
```

(\SETSPACE.PRESS

```
[LAMBDA (PRSTREAM S)                                             (* rmk%: "31-Mar-86 16:08")
  (PROG (ELSTREAM (PRDATA (fetch IMAGEDATA of PRSTREAM)))
        (AND (EQ S (ffetch PRSPACEWIDTH of PRDATA))
             (RETURN))
        (SHOW.PRESS PRSTREAM)
        (SETQ ELSTREAM (fetch ELSTREAM of (fetch IMAGEDATA of PRSTREAM)))
        (if (ILEQ S 2047)
            then (\WOUT ELSTREAM (IPLUS (LLSH SetSpaceXShortCode 8)
                                        S))
          else (\BOUT ELSTREAM SetSpaceXCode)
               (\WOUT ELSTREAM S))
        (freplace PRSPACEWIDTH of PRDATA with S])
```

(\STARTPAGE.PRESS

```
[LAMBDA (PRSTREAM)                                               (* rmk%: "25-Feb-86 11:36")
                                                                 (* Should be called only when no previous page is open)
  (PROG (CFONT HFONT SPACEFACTOR (PRDATA (ffetch IMAGEDATA of PRSTREAM)))
        (SETQ CFONT (ffetch PRLOGICALFONT of PRDATA))

        (* Save current font so that \ENTITYSTART.PRESS can make PRLOGICALFONT be NIL, indicating that there is no actual
        font at the beginning of a page)

        (\ENTITYSTART.PRESS PRSTREAM)
        [COND
           ((ffetch PRHEADING of PRDATA)
            (SETQ SPACEFACTOR (ffetch PRSPACEFACTOR of PRDATA))
            (freplace PRSPACEFACTOR of PRDATA with 1)
            (SETQ HFONT (ffetch PRHEADINGFONT of PRDATA))
            (\DSPFONT.PRESS PRSTREAM HFONT)                      (* Set up heading font)
            [SETXY.PRESS PRSTREAM (ffetch PRLEFT of PRDATA)
                   (IDIFFERENCE (ffetch PRTOP of PRDATA)
                          (FONTPROP HFONT 'ASCENT]
            (PRIN3 (ffetch PRHEADING of PRDATA)
                   PRSTREAM)                                     (* Skip an inch before page number)
            (SHOW.PRESS PRSTREAM)
            (SETX.PRESS PRSTREAM (IPLUS MICASPERINCH (ffetch PRXPOS of PRDATA)))
            (PRIN3 "Page " PRSTREAM)
            (PRIN3 (add (ffetch PRPAGENUM of PRDATA)
```

```
                        1)
                    PRSTREAM)
            (NEWLINE.PRESS PRSTREAM)                              (* Skip 2 lines)
            (NEWLINE.PRESS PRSTREAM)
            (freplace PRSPACEFACTOR of PRDATA with SPACEFACTOR))
           (T (SETXY.PRESS PRSTREAM (ffetch PRLEFT of PRDATA)
                    (IDIFFERENCE (ffetch PRTOP of PRDATA)
                          (FONTPROP CFONT 'ASCENT]          (* Now we set the font to our (previous) current font)
        (\DSPFONT.PRESS PRSTREAM CFONT])
```

(\PRESS.COERCEFONT

```
  [LAMBDA (FONT FAMILY)                                           (* rmk%: "25-Mar-86 15:44")
                                                                  (* coerces FONT to be new FAMILY FAMILY, and caches result
    on \PRESS.COERCEDFONTS)
    (DECLARE (GLOBALVARS \PRESS.COERCEDFONTS))
    (COND
       [[OR (NOT FAMILY)
            (EQ FAMILY (FONTPROP FONT 'FAMILY]

          (* Don't call FONTCOPY if it's the same font. This avoids circularity thru AVGCHARWIDTH and CHARWIDTH before the
          font has been stored in \FONTSINCORE.)

        (COND
           ((EQ 'PRESS (FONTPROP FONT 'DEVICE))                   (* How could it not be PRESS? Ask Tad.)
            FONT)
           (T (FONTCOPY FONT 'DEVICE 'PRESS]
       ((OR (FONTP FAMILY)
            (LISTP FAMILY))                                       (* FAMILY is a font specification)
        (FONTCOPY FAMILY 'DEVICE 'PRESS))
       [(FONTP (CADR (ASSOC FONT (CDR (ASSOC FAMILY \PRESS.COERCEDFONTS]
       (T (LET [(pressFont (OR (FONTCOPY FONT 'FAMILY FAMILY 'DEVICE 'PRESS 'NOERROR T)
                               (FONTCOPY FONT 'FAMILY FAMILY 'FACE 'STANDARD 'DEVICE 'PRESS]
                (push [CDR (OR (ASSOC FAMILY \PRESS.COERCEDFONTS)
                               (CAR (push \PRESS.COERCEDFONTS (CONS FAMILY]
                      (LIST FONT pressFont))
               pressFont])
```

(\DSPFONT.PRESSFONT

```
  [LAMBDA (PRSTREAM PRFONT)                                       (* thh%: "16-Jun-86 10:50")
                                                                  (* Changes the Pressfiles device font)
    (PROG (FDENTRY LFONT OLDFONT (PRDATA (ffetch IMAGEDATA of PRSTREAM)))
          (SETQ OLDFONT (ffetch PRFONT of PRDATA))
          (SHOW.PRESS PRSTREAM)
          (SETQ FDENTRY (\DEFINEFONT.PRESS PRSTREAM PRFONT))
          (COND
             ((NEQ (ffetch FONTSET# of FDENTRY)
                   (ffetch FONTSET# of (ffetch PRCURRFDE of PRDATA)))
                                                                  (* Swtich font sets)
                                                                  (* must save and restore current logical font since
                                                                  \ENTITYSTART.PRESS makes it NIL)
              (SETQ LFONT (ffetch PRLOGICALFONT of PRDATA))
              (\ENTITYEND.PRESS PRSTREAM)
              (\ENTITYSTART.PRESS PRSTREAM)
              (\DSPFONT.PRESS PRSTREAM LFONT)))
          (freplace PRCURRFDE of PRDATA with FDENTRY)
          (freplace PRFONT of PRDATA with PRFONT)
          (\BOUT (ffetch ELSTREAM of PRDATA)
                 (LOGOR FontCode (ffetch FONT# of FDENTRY)))
          (RETURN OLDFONT])
```

(\SETUPFONTS.PRESS

```
  [LAMBDA (PRSTREAM FONTS)                                        (* thh%: "10-Dec-86 08:43")

          (* creates fonts in the initial fontset. and sets heading font. Leaves PRFONT as NIL.
          This means that \DSPFONT.PRESS of the heading font will establish that as the current font when the first page opens.)

          (* since FONTS are logical, not device, fonts, they are not added to the fontset here)

    (for F FLG inside (OR FONTS DEFAULTFONT) do (SETQ F (FONTCREATE F NIL NIL NIL 'PRESS))
                                                (COND
                                                   (FLG NIL)
                                                   (T (\DSPFONT.PRESS PRSTREAM F)
                                                                  (* Install first font as current logical font and heading font.)
                                                      (\ENTITYEND.PRESS PRSTREAM)
                                                      (replace PRHEADINGFONT of (fetch IMAGEDATA of PRSTREAM)
                                                         with F)
                                                      (SETQ FLG T])

)

(DEFINEQ

(\CREATEPRESSFONT
  [LAMBDA (FAMILY PSIZE FACE ROTATION DEVICE)                     ; Edited  9-Mar-88 15:54 by thh:
```

```
;; Widths array is fully allocated, with zeroes for characters with no information.  An array is not allocated for fixed WidthsY.  DEVICE is PRESS or
;; INTERPRESS
(DECLARE (GLOBALVARS PRESSFONTWIDTHSFILES))
(RESETLST                                                  ; RESETLST to make sure the fontfiles get closed
    (PROG ((FD (create FONTDESCRIPTOR
                       FONTDEVICE _ DEVICE
                       FONTFAMILY _ FAMILY
                       FONTSIZE _ PSIZE
                       FONTFACE _ FACE
                       \SFFACECODE _ (\FACECODE FACE)
                       ROTATION _ ROTATION
                       FONTSCALE _ (CONSTANT (FQUOTIENT 2540 72))
                       \SFHeight _ 0
                       \SFAscent _ 0
                       \SFDescent _ 0)))
          (OR (\GETCHARSETINFO 0 FD T)
              (RETURN NIL))
          (RETURN FD)))])
```

(**CREATECHARSET.PRESS**

```
  [LAMBDA (FAMILY SIZE FACE ROTATION DEVICE CHARSET FONTDESC)     ; Edited  9-Mar-88 15:19 by thh:

;;; determines widths and translations to print the charset with Press fonts.  Note that we get widths from widths of font translated to, which should be
;;; original press values because translations are always to press fonts.

;;; NOTE: This code makes fonts that translate to themselves circular, and also gives fonts high reference counts. The translations should not be circular.

    (DECLARE (GLOBALVARS PRESSFONTFAMILIES))
    (PROG ((CSETTRANSLATIONARRAY (\NSTOASCIIARRAY CHARSET))
           CSINFO widths (translationArray (ARRAY 256 NIL NIL 0))
           (ascent 0)
           (descent 0)
           CSETZEROTRANSLATIONS)
     ;; Determine translations for this charset
          [COND
             [(ZEROP CHARSET)
              ;; set up charsetinfo -- includes any coercions to known press fonts
              (SETQ CSINFO (\CREATECHARSETZERO.PRESS FAMILY SIZE FACE ROTATION DEVICE FONTDESC))
              (OR CSINFO (RETURN NIL))                     ; unable to coerce to a press font
              ;; get translations for charset-0
              (COND
                 [(SETQ CSETZEROTRANSLATIONS (ASSOC (FONTPROP FONTDESC 'FAMILY)
                                                    PRESSFONTFAMILIES))
                                                           ; use identity transformation
                  (for i from 0 to 255 do (SETA translationArray i (CONS FONTDESC i)))
                                                           ; except for font-specific non-identities
                  (for X in (CDR CSETZEROTRANSLATIONS) do (SETA translationArray (CAR X)
                                                                (\NSTOASCIITRANSLATION (CADR X)
                                                                      FAMILY FONTDESC]
                 (T ;; Not a press font: assume NS font which will be translated into a press font
                    (for i from 0 to 255 do (SETA translationArray i
                                                  (\NSTOASCIITRANSLATION
                                                   (COND
                                                      ((AND CSETTRANSLATIONARRAY (ELT CSETTRANSLATIONARRAY i)))
                                                      (T (LIST (OR FAMILY (FONTPROP FONTDESC 'FAMILY))
                                                               i)))
                                                   FAMILY FONTDESC]
             (T ;; CHARSET not zero, assume NS codes
                (for i from 0 to 255 do (SETA translationArray i (\NSTOASCIITRANSLATION (AND CSETTRANSLATIONARRAY
                                                                                             (ELT
                                                                                             CSETTRANSLATIONARRAY
                                                                                                  i))
                                                                                        FAMILY FONTDESC]
     ;; Set the widths array and install the translations in the CHARSETINFO
          (OR CSINFO (SETQ CSINFO (create CHARSETINFO)))
          (SETQ widths (fetch (CHARSETINFO WIDTHS) of CSINFO))
          (for i from 0 to 255 bind translation pressFont newAscent newDescent
             do (SETQ translation (ELT translationArray i))
                (SETQ pressFont (CAR translation))
                [COND
                   ((AND (ZEROP CHARSET)
                         (EQ pressFont FONTDESC))          ; this is charset-0 font translating to itself, use widths already
                                                          ; defined
                    (\FSETWIDTH widths i (\FGETWIDTH widths (CDR translation)))
                    (SETQ newAscent (fetch (CHARSETINFO CHARSETASCENT) of CSINFO))
                    (SETQ newDescent (fetch (CHARSETINFO CHARSETDESCENT) of CSINFO)))
                   (T (\FSETWIDTH widths i (\FGETCHARWIDTH pressFont (CDR translation)))
                      (SETQ newAscent (ffetch (FONTDESCRIPTOR \SFAscent) of pressFont))
                      (SETQ newDescent (ffetch (FONTDESCRIPTOR \SFDescent) of pressFont]
```

```
            (SETQ ascent (MAX ascent newAscent))
            (SETQ descent (MAX descent newDescent)))
         (replace (CHARSETINFO CHARSETBITMAP) of CSINFO with (ffetch (ARRAYP BASE) of translationArray))
         (replace (CHARSETINFO CHARSETASCENT) of CSINFO with ascent)
         (replace (CHARSETINFO CHARSETDESCENT) of CSINFO with descent)
         (RETURN CSINFO])
```

(**CREATECHARSETZERO.PRESS**

```
  [LAMBDA (FAMILY SIZE FACE ROTATION DEVICE FD)                    ; Edited 9-Mar-88 15:27 by thh:
```

;;; creates CSINFO for charset 0 of press fonts from info in widths file (without translations).

```
    (DECLARE (GLOBALVARS PRESSFONTWIDTHSFILES FONTCOERCIONS MISSINGFONTCOERCIONS))
    (RESETLST                                                      ; RESETLST to make sure the fontfiles get closed
        (PROG* (WSTRM STRMCACHE FIXEDFLAGS RELFLAG FIRSTCHAR LASTCHAR TEM WIDTHSY WIDTHS
                (PRESSMICASIZE (IQUOTIENT (ITIMES SIZE 2540)
                                          72))
                (NSMICASIZE (FIXR (FQUOTIENT (ITIMES SIZE 2540)
                                             72)))
                (FACECODE (\FACECODE FACE))
                (CSINFO (create CHARSETINFO))
                CHARSETHEIGHT FOO FBBOX)
```

;;; Go look for the fonts.widths file that has this font's info in it.

```
               (OR [bind XLATEDNAME NEWFAMILY NEWNSMICASIZE NEWFACECODE for F inside PRESSFONTWIDTHSFILES
                     when (INFILEP F) first (SETQ XLATEDNAME (\COERCEFONT FAMILY SIZE FACE ROTATION
                                                                          'PRESS FONTCOERCIONS))
                                            [COND
                                               (XLATEDNAME (SETQ NEWFAMILY (CAR XLATEDNAME))
                                                           (SETQ NEWNSMICASIZE (FIXR (FQUOTIENT (ITIMES (CADR
                                                                                                          XLATEDNAME
                                                                                                          )
                                                                                                2540)
                                                                                                72)))
                                                           (SETQ NEWFACECODE (\FACECODE (CADDR XLATEDNAME]
                     do                                            ; Look thru the candidate PRESSFONTWIDTHSFILES for a file
                                                                   ; that has a description for this font.
                        [COND
                           [(SETQ WSTRM (\GETSTREAM F 'INPUT T))
                            (COND
                               ((RANDACCESSP WSTRM)
                                (RESETSAVE NIL (LIST 'SETFILEPTR WSTRM (GETFILEPTR WSTRM)))
                                (SETFILEPTR WSTRM 0]
                           (T (RESETSAVE (SETQ WSTRM (OPENSTREAM F 'INPUT 'OLD 8))
                                         '(PROGN (CLOSEF? OLDVALUE]
                        [OR (RANDACCESSP WSTRM)
                            (COPYBYTES WSTRM (SETQ WSTRM (OPENSTREAM '{NODIRCORE} 'BOTH 'NEW]
                        (push STRMCACHE WSTRM)                     ; Save for coercions below
                        (COND
                           ((SETQ RELFLAG (\POSITIONFONTFILE WSTRM (OR NEWNSMICASIZE NSMICASIZE)
                                                             FIRSTCHAR LASTCHAR (OR NEWFAMILY FAMILY)
                                                             (OR NEWFACECODE FACECODE)))
                                                                   ; OK, we found this font described in this file.
                            (COND
                               (XLATEDNAME (replace FONTDEVICESPEC of FD with XLATEDNAME)
                                           (SETQ NSMICASIZE NEWNSMICASIZE)))
                            (RETURN T]
                     [bind XLATEDNAME NEWFAMILY NEWNSMICASIZE NEWFACECODE XLATEDNAMES first (SETQ STRMCACHE
                                                                                                (DREVERSE STRMCACHE))
                        while (SETQ XLATEDNAME (\COERCEFONT FAMILY SIZE FACE ROTATION 'PRESS MISSINGFONTCOERCIONS
                                                            XLATEDNAMES))
                        thereis (push XLATEDNAMES XLATEDNAME)
                               (for old WSTRM in STRMCACHE first (SETQ NEWFAMILY (CAR XLATEDNAME))
                                                                 (SETQ NEWNSMICASIZE (FIXR (FQUOTIENT
                                                                                             (ITIMES (CADR XLATEDNAME)
                                                                                                     2540)
                                                                                             72)))
                                                                 (SETQ NEWFACECODE (\FACECODE (CADDR XLATEDNAME)))
                                  do                               ; Now try coercing the family name
                                     ;; We know the file was left open and is randaccessp from the previous loop, which must have run off the
                                     ;; end of the file list
                                     (SETFILEPTR WSTRM 0)
                                     (COND
                                        ((SETQ RELFLAG (\POSITIONFONTFILE WSTRM NEWNSMICASIZE FIRSTCHAR LASTCHAR
                                                                          NEWFAMILY NEWFACECODE))
                                         (replace FONTDEVICESPEC of FD with XLATEDNAME)
                                         (SETQ NSMICASIZE NEWNSMICASIZE)
                                         (RETURN T]
                        (RETURN NIL))
```

;;; Having found the font-widths file, now read the width info from it.

```
               (SETQ RELFLAG (ZEROP RELFLAG))                      ; Actually, \POSITIONFONTFILE returns zero if the font metrics
                                                                   ; are size-relative and must be scaled.
```

```
(SETQ WIDTHS (fetch (CHARSETINFO WIDTHS) of CSINFO))
(SETFILEPTR WSTRM (UNFOLD (\FIXPIN WSTRM)
                          BYTESPERWORD))
```

;; Read the location of the WD segment for this font (we're in the directory part of the file now), and go there.

```
(SETQ FBBOX (SIGNED (\WIN WSTRM)
                    BITSPERWORD))                ; replace (FONTDESCRIPTOR FBBOX) of FD with (SIGNED
                                                 ; (\WIN WSTRM) BITSPERWORD)
                                                 ; Get the max bounding width for the font
(replace (CHARSETINFO CHARSETDESCENT) of CSINFO with (IMINUS (SIGNED (\WIN WSTRM)
                                                                     BITSPERWORD)))
                                                 ; Descent is -FBBOY
(SETQ FOO (\WIN WSTRM))                          ; replace (FONTDESCRIPTOR FBBDX) of FD with (SIGNED
                                                 ; (\WIN WSTRM) BITSPERWORD)
                                                 ; And the standard kern value (?)
(SETQ CHARSETHEIGHT (SIGNED (\WIN WSTRM)
                            BITSPERWORD))        ; replace \SFHeight of FD with (SIGNED (\WIN WSTRM)
                                                 ; BITSPERWORD)
                                                 ; Height is FBBDY
[COND
   (RELFLAG                                      ; Dimensions are relative, must be scaled
          ;; replace (FONTDESCRIPTOR FBBOX) of FD with (IQUOTIENT (ITIMES (fetch (FONTDESCRIPTOR FBBOX) of
          ;; FD) NSMICASIZE) 1000)
          (replace (CHARSETINFO CHARSETDESCENT) of CSINFO with (IQUOTIENT
                                                                 (ITIMES (fetch (CHARSETINFO
                                                                                  CHARSETDESCENT
                                                                                  )
                                                                                of CSINFO)
                                                                         NSMICASIZE)
                                                                 1000))
          ;; replace (FONTDESCRIPTOR FBBDX) of FD with (IQUOTIENT (ITIMES (fetch (FONTDESCRIPTOR FBBDX) of FD)
          ;; NSMICASIZE) 1000)
          (SETQ CHARSETHEIGHT (IQUOTIENT (ITIMES CHARSETHEIGHT NSMICASIZE)
                                         1000]
(replace (CHARSETINFO CHARSETASCENT) of CSINFO with (IDIFFERENCE CHARSETHEIGHT
                                                                 (fetch CHARSETDESCENT of CSINFO)))
(SETQ FIXEDFLAGS (LRSH (\BIN WSTRM)
                       6))                       ; The fixed flags
(\BIN WSTRM)                                     ; Skip the spares
[COND
   ((EQ 2 (LOGAND FIXEDFLAGS 2))                 ; This font is fixed width.
    (SETQ TEM (\WIN WSTRM))                      ; Read the fixed width for this font
    [COND
       ((AND RELFLAG (NOT (ZEROP TEM)))          ; If it's size relative, scale it.
        (SETQ TEM (IQUOTIENT (ITIMES TEM NSMICASIZE)
                             1000]
    (for I from FIRSTCHAR to LASTCHAR do         ; Fill in the char widths table with the width.
                                        (\FSETWIDTH WIDTHS I TEM)))
   (T                                            ; Variable width font, so we have to read widths.
                                                 ; AIN WIDTHS FIRSTCHAR (ADD1 (IDIFFERENCE LASTCHAR
                                                 ; FIRSTCHAR)) WSTRM
      (for I from FIRSTCHAR to LASTCHAR do (\FSETWIDTH WIDTHS I noInfoCode))
      (\BINS (\GETOFD WSTRM 'INPUT)
             WIDTHS
             (UNFOLD FIRSTCHAR BYTESPERWORD)
             (UNFOLD (ADD1 (IDIFFERENCE LASTCHAR FIRSTCHAR))
                     BYTESPERWORD))              ; Read the X widths.
      (for I from FIRSTCHAR to LASTCHAR when (EQ noInfoCode (\FGETWIDTH WIDTHS I))
         do                                      ; For chars that have no width info, let width be zero.
            (\FSETWIDTH WIDTHS I 0))
      (COND
         (RELFLAG                                ; If the widths are size-relative, scale them.
                (for I from FIRSTCHAR to LASTCHAR
                   do (\FSETWIDTH WIDTHS I (IQUOTIENT (ITIMES (\FGETWIDTH WIDTHS I)
                                                              NSMICASIZE)
                                                      1000]
[COND
   [(EQ 1 (LOGAND FIXEDFLAGS 1))
    (COND
       ((ILESSP (GETFILEPTR WSTRM)
                (GETEOFPTR WSTRM))
        (SETQ WIDTHSY (\WIN WSTRM)))
       (T                                        ; STAR FONT FILES LIKE TO LEAVE OFF THE Y WIDTH.
          (SETQ WIDTHSY 0)))                     ; The fixed width-Y for this font;  the width-Y field is a single
                                                 ; integer in the FD
    (replace (CHARSETINFO YWIDTHS) of CSINFO with (COND
                                                    ((AND RELFLAG (NOT (ZEROP WIDTHSY)))
                                                     (IQUOTIENT (ITIMES WIDTHSY NSMICASIZE)
                                                                1000))
                                                    (T WIDTHSY]
   (T                                            ; Variable Y-width font.  Fill it in as above
      (SETQ WIDTHSY (replace (CHARSETINFO YWIDTHS) of CSINFO with (\CREATECSINFOELEMENT)))
      (for I from FIRSTCHAR to LASTCHAR do (\FSETWIDTH WIDTHSY I noInfoCode))
      (\BINS (\GETOFD WSTRM 'INPUT)
             WIDTHSY
```

```
                        (UNFOLD FIRSTCHAR BYTESPERWORD)
                        (UNFOLD (ADD1 (IDIFFERENCE LASTCHAR FIRSTCHAR))
                               BYTESPERWORD))                ; Read the Y widths
                  (for I from FIRSTCHAR to LASTCHAR when (EQ noInfoCode (\FGETWIDTH WIDTHSY I))
                     do                                      ; Let any characters with no width info be zero height
                        (\FSETWIDTH WIDTHSY I 0))
                  (COND
                     (RELFLAG                                ; If the widths are size-relative, scale them.
                            (for I from FIRSTCHAR to LASTCHAR
                               do (\FSETWIDTH WIDTHSY I (IQUOTIENT (ITIMES (\FGETWIDTH WIDTHSY I)
                                                                           NSMICASIZE)
                                                                   1000]
            (RETURN CSINFO)))])

)

(DEFINEQ
```

(**PRESSCURVE2**

```
  [LAMBDA (PRSTREAM SPLINE DASHING BRUSHFONT)                  (* thh%: "16-Jun-86 10:53")
                                                               (* Given a spline curve and a font, draw the lines to PRSTREAM)
    (RESETLST
        (RESETSAVE NIL (LIST '\DSPFONT.PRESSFONT PRSTREAM (\DSPFONT.PRESSFONT PRSTREAM BRUSHFONT)))
        [PROG ((PRDATA (fetch IMAGEDATA of PRSTREAM)))
              (COND
                 ((IGREATERP (IDIFFERENCE (GETFILEPTR (fetch ELSTREAM of PRDATA))
                                          (fetch ELSTARTBYTE of PRDATA))
                             25000)
                  (\ENTITYEND.PRESS PRSTREAM)              (* Hack to prevent mysterious overflow in length of entities)
                  (\ENTITYSTART.PRESS PRSTREAM]
        (\BOUT (fetch ELSTREAM of (fetch IMAGEDATA of PRSTREAM))
               ResetSpaceCode)

          (* because the space code shouldn't be interpreted specially when we are drawing in the vector font)

        (PROG ((XPOLY (create POLYNOMIAL))
               (X'POLY (create POLYNOMIAL))
               (YPOLY (create POLYNOMIAL))
               (Y'POLY (create POLYNOMIAL))
               (X (fetch (SPLINE SPLINEX) of SPLINE))
               (Y (fetch (SPLINE SPLINEY) of SPLINE))
               (X' (fetch (SPLINE SPLINEDX) of SPLINE))
               (Y' (fetch (SPLINE SPLINEDY) of SPLINE))
               (X'' (fetch (SPLINE SPLINEDDX) of SPLINE))
               (Y'' (fetch (SPLINE SPLINEDDY) of SPLINE))
               (X''' (fetch (SPLINE SPLINEDDDX) of SPLINE))
               (Y''' (fetch (SPLINE SPLINEDDDY) of SPLINE))
               (%#KNOTS (fetch %#KNOTS of SPLINE))
               (X0 (ELT (fetch (SPLINE SPLINEX) of SPLINE)
                        1))
               (Y0 (ELT (fetch (SPLINE SPLINEY) of SPLINE)
                        1))
               IX IY DX DY XT YT X'T Y'T NEWXT NEWYT XDIFF YDIFF XWALLDT YWALLDT DUPLICATEKNOT EXTRANEOUS TT
               NEWT DELTA DASHON DASHLST DASHCNT HALFVECWIDTH PUTDX EXTRADX PUTDY EXTRADY)
              (SETQ HALFVECWIDTH (FONTPROP BRUSHFONT 'SIZE))

          (* Half the width of the brush, in dots. Used to help decide when the line we're drawing goes off-paper.)

              (SETQ DASHON T)

          (* These are initialized outside the prog-bindings cause the compiler can't hack so many initialized variables)

              (SETQ DASHLST DASHING)
              (SETQ DASHCNT (CAR DASHING))
              (SETXY.PRESS PRSTREAM (FIXR (FTIMES X0 MicasPerScan))
                     (FIXR (FTIMES Y0 MicasPerScan)))      (* Move to the first knot on the curve)
              (replace VECMOVINGRIGHT of (fetch IMAGEDATA of PRSTREAM) with T)
                                                           (* Start by assuming we're moving in increasing X
                                                           (since the vector fonts only have strokes that work in that
                                                           direction))
              (replace VECWASDISPLAYING of (fetch IMAGEDATA of PRSTREAM) with (AND (GEQ X0 0)
                                                                                   (GEQ Y0 0)))
              (replace VECSEGCHARS of (fetch IMAGEDATA of PRSTREAM) with NIL)
              (replace VECCURX of (fetch IMAGEDATA of PRSTREAM) with X0)
                                                           (* And set the current X and Y positions, denominated in dover
                                                           spots)
              (replace VECCURY of (fetch IMAGEDATA of PRSTREAM) with Y0)
                                                           (* Set up initial values in vec variables, perform SetX/SetY.)
              (SETQ TT 0.0)
              (SETQ DELTA 16)
              (SETQ IX (FIXR X0))
              (SETQ IY (FIXR Y0))
              [for KNOT# from 1 to (SUB1 %#KNOTS)
                 do (LOADPOLY XPOLY X'POLY (ELT X''' KNOT#)
                              (ELT X'' KNOT#)
                              (ELT X' KNOT#)
                              (ELT X KNOT#))               (* Set up the polynomials that describe X and X' over this
                                                           segment)
```

```
(LOADPOLY YPOLY Y'POLY (ELT Y''' KNOT#)
       (ELT Y'' KNOT#)
       (ELT Y' KNOT#)
       (ELT Y KNOT#))                          (* Set up the polynomials that describe Y and Y' over this segment)
(SETQ XT (POLYEVAL TT XPOLY 3))                (* XT _ X (t) --Evaluate the next point)
(SETQ YT (POLYEVAL TT YPOLY 3))                (* YT _ Y (t))
(COND
   [(NOT (IEQP KNOT# (SUB1 %#KNOTS)))
```

(* This isn't the last knot. Check to see if the next knot in line is a duplicated knot.)

```
    (SETQ DUPLICATEKNOT (AND (EQP (ELT X (ADD1 KNOT#))
                                  (ELT X (IPLUS KNOT# 2)))
                             (EQP (ELT Y (ADD1 KNOT#))
                                  (ELT Y (IPLUS KNOT# 2]
   (T (SETQ DUPLICATEKNOT NIL)))
[until (GEQ TT 1.0) do
```

(* Run the parameter, TT, from 0.0 up to 1.0. That moves the X and Y locations smoothly from this knot to the next one.)

```
                                (SETQ X'T (POLYEVAL TT X'POLY 2))
                                                   (* X'T _ X' (t))
                                (SETQ Y'T (POLYEVAL TT Y'POLY 2))
                                                   (* Y'T _ Y' (t))
                                (COND
                                   ((EQP X'T 0.0)  (* Never let X' really get to 0.0 -- things become ill-conditioned there.)
                                    (SETQ X'T 5.0E-4)))
                                (COND
                                   ((EQP Y'T 0.0)  (* Likewise Y'.)
                                    (SETQ Y'T 5.0E-4)))
                                [COND
                                   ((FGTP X'T 0.0) (* If X' is positive, we'll try moving in the +X direction)
                                    (SETQ DX DELTA))
                                   (T              (* If not, we'll try the -X direction.)
                                      (SETQ DX (IMINUS DELTA]
                                [COND
                                   ((FGTP Y'T 0.0) (* Likewise, if Y' is positive, try moving by DELTA in the +Y direction)
                                    (SETQ DY DELTA))
                                   (T (SETQ DY (IMINUS DELTA]
                                (SETQ XWALLDT (FQUOTIENT (FDIFFERENCE (IPLUS IX DX)
                                                                      XT)
                                                         X'T))
                                                   (* Compute a dT, based on moving by DELTA in X.)
                                (SETQ YWALLDT (FQUOTIENT (FDIFFERENCE (IPLUS IY DY)
                                                                      YT)
                                                         Y'T))
                                                   (* And a dT based on moving by DELTA in Y.)
                                [COND
                                   ((FLESSP XWALLDT YWALLDT)
```

(* Use the smaller of the two dT's. In this case, dT for X was smaller, so compute a new DY as depending on DX.)

```
                                    (SETQ NEWT (FPLUS TT XWALLDT))
                                    (SETQ DY (IDIFFERENCE (FIXR (FPLUS YT (FTIMES XWALLDT Y'T)))
                                                          IY)))
                                   (T
```

(* Changing Y gave the smaller dT. Compute a new DX, as though it depended on DY.)

```
                                      (SETQ NEWT (FPLUS TT YWALLDT))
                                      (SETQ DX (IDIFFERENCE (FIXR (FPLUS XT (FTIMES YWALLDT X'T)))
                                                            IX]
                                (SETQ PUTDX DX)
                                (SETQ EXTRADX 0)
                                (SETQ PUTDY DY)
                                (SETQ EXTRADY 0)
                                [COND
                                   ((IGREATERP DX 16)
                                    (SETQ PUTDX 16)
                                    (SETQ EXTRADX (IDIFFERENCE DX 16]
                                [COND
                                   ((IGREATERP -16 DX)
                                    (SETQ PUTDX -16)
                                    (SETQ EXTRADX (IPLUS DX 16]
                                [COND
                                   ((IGREATERP DY 16)
                                    (SETQ PUTDY 16)
                                    (SETQ EXTRADY (IDIFFERENCE DY 16]
                                [COND
                                   ((IGREATERP -16 DY)
                                    (SETQ PUTDY -16)
                                    (SETQ EXTRADY (IPLUS DY 16]
                                (COND
                                   ([AND (FGTP NEWT 1.0)
```

```
                                                    (OR DUPLICATEKNOT (EQ KNOT# (SUB1 %#KNOTS]
                                                (SETQ NEWT 1.0)))
                                            (SETQ NEWXT (POLYEVAL NEWT XPOLY 3))
                                                                (* New XT _ X (new t))
                                            (SETQ NEWYT (POLYEVAL NEWT YPOLY 3))
                                                                (* New YT _ Y (new t))
                                            (SETQ XDIFF (ABS (FDIFFERENCE (IPLUS IX DX)
                                                                          NEWXT)))
                                            (SETQ YDIFF (ABS (FDIFFERENCE (IPLUS IY DY)
                                                                          NEWYT)))
                                            (COND
                                               ((AND (IGREATERP DELTA 1)
                                                     (OR (FGTP XDIFF 1.0)
                                                         (FGTP YDIFF 1.0)))
```

(* If we're more than a dover spot off where we'd expect to be because of the size of DELTA--and if there's room to make DELTA smaller--then try DELTA_DELTA/2)

```
                                                (SETQ DELTA (LRSH DELTA 1)))
                                               (T
```

(* No, this estimate is close enough. Put out a vector segment based on it, and move to the new TT.)

```
                                                   (\VECPUT PRSTREAM PUTDX PUTDY HALFVECWIDTH)
                                                                (* Print out a stroke using the vector font.)
                                                   (COND
                                                      ((OR (NEQ EXTRADX 0)
                                                           (NEQ EXTRADY 0))
                                                                (* If, actually, it was too big for one stroke, use another.)
                                                       (\VECPUT PRSTREAM EXTRADX EXTRADY HALFVECWIDTH)))
                                                   (SETQ IX (IPLUS IX DX))
                                                                (* Our new current location, in Dover spots)
                                                   (SETQ IY (IPLUS IY DY))
                                                   (SETQ TT NEWT)   (* Set TT to its new value)
                                                   (SETQ XT NEWXT)  (* And set the new floating-point values for X
                                                                    (t) and Y (t)%.)
                                                   (SETQ YT NEWYT)
                                                   (COND
                                                      ((AND (ILESSP DELTA 16)
                                                            (OR (FLESSP XDIFF 0.5)
                                                                (FLESSP YDIFF 0.5)))
                                                                (* If we were especially close, try making DELTA larger for the
                                                                next go round.)
                                                       (SETQ DELTA (LLSH DELTA 1]
                                            (SETQ TT (FDIFFERENCE TT 1.0))
```

(* Having moved past a knot, back the value of the parameter TT back down.
However, don't set it to 0.0--let's try to keep the line going from where it got to in passing the last knot.)

```
                                            (COND
                                               (DUPLICATEKNOT
```

(* This next knot is a duplicate. Skip over it, and start from the following knot.
This will avoid odd problems trying to go nowhere while obeying the constraints of X' and Y' at that knot--since it's a duplicate, X' and Y' are discontinuous there.)

```
                                                    (add KNOT# 1]
                                (\ENDVECRUN PRSTREAM HALFVECWIDTH)))])

)


(* * Generic utility for coercing fonts, could be used by other devices)

(DEFINEQ

(\COERCEFONT
  [LAMBDA (FAMILY SIZE FACE ROTATION DEVICE COERCELIST BUTNOT CREATEFLG)
                                                                ; Edited  9-Mar-88 12:58 by thh:

    ;; Returns a font name that the requested font specification coerces to according to COERCELIST.  If CREATEFLG is T, only returns name-lists for
    ;; which a font descriptor has been created.  BUTNOT can be a list of font-specs which are not an acceptable coercion--e.g.  a previous one that
    ;; failed, so we want to keep looking beyond that one.

;;; NOSLUG?  means don't create an empty (slug) csinfo if the charset is not found, just return NIL (probably only useful for display fonts)

    ;; COERCELIST is an alist of font coercions indexed by device, with the value for each device being a list of the form ((user-font real-font)
    ;; (user-font real-font) ...) --- Each user-font is either simply a family name, or a list of FAMILY, and optionally SIZE, and FACE, in standard
    ;; font-name order.  Any of these can be NIL, meaning that any requested value matches.  In addition, the SIZE can be either a specific number, or
    ;; a constraint of the form (< n) or (> n), which matches requested sizes that are less than or greater than the constraint size n.  --- The real-font is a
    ;; similar family-name or list, except that a NIL field here means that the requested parameter is simply carried over.  Also, no size constraints, only
    ;; explicit sizes, are allowed.  (e.g., (GACHA) or (GACHA (< 10)) or (GACHA 10))

    (for TRANSL in (CDR (ASSOC DEVICE COERCELIST)) bind NEWCSINFO USERSPEC REALSPEC FAMCONSTRAINT SIZECONSTRAINT
                                                       FACECONSTRAINT NEWFONTNAME
       when (AND (SETQ USERSPEC (CAR TRANSL))
                 (OR [NULL (SETQ FAMCONSTRAINT (COND
                                                  ((LISTP USERSPEC)
                                                   (pop USERSPEC))
```

```
                                                (T (PROG1 USERSPEC (SETQ USERSPEC NIL]
                    (EQ FAMILY FAMCONSTRAINT))
                (OR (NOT (SETQ SIZECONSTRAINT (pop USERSPEC)))
                    (EQ SIZE SIZECONSTRAINT)
                    (AND (LISTP SIZECONSTRAINT)
                         (SELECTQ (CAR SIZECONSTRAINT)
                              (< (LESSP SIZE (CADR SIZECONSTRAINT)))
                              (> (GREATERP SIZE (CADR SIZECONSTRAINT)))
                             NIL)))
                (OR (NOT (SETQ FACECONSTRAINT (pop USERSPEC)))
                    (EQUAL FACE FACECONSTRAINT))
                (SETQ REALSPEC (CADR TRANSL))
                (SETQ NEWFONTNAME (LIST (OR [COND
                                            ((LISTP REALSPEC)
                                             (pop REALSPEC))
                                            (T (PROG1 REALSPEC (SETQ REALSPEC NIL]
                                        FAMILY)
                                    (OR (pop REALSPEC)
                                        SIZE)
                                    (\FONTFACE (OR (pop REALSPEC)
                                                   FACE))
                                   ROTATION DEVICE))
                (NOT (for EXCLUDE in BUTNOT thereis (EQUAL EXCLUDE NEWFONTNAME)))
                (OR (NULL CREATEFLG)
                    (FONTCREATE NEWFONTNAME NIL NIL NIL NIL T)))
       do (RETURN NEWFONTNAME])

)

(ADDTOVAR FONTCOERCIONS (PRESS ((SYMBOL (< 10))
                                (SYMBOL 10))
                               ((SYMBOL (> 12))
                                (SYMBOL 12))))

(ADDTOVAR MISSINGFONTCOERCIONS (PRESS (MODERN HELVETICA)
                                      (CLASSIC TIMESROMAN)
                                      (LOGOTYPE LOGO)
                                      (TERMINAL GACHA)
                                      (MODERN FRUTIGER)
                                      (CLASSIC CENTURY)))

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS FONTCOERCIONS MISSINGFONTCOERCIONS)
)

(DEFINEQ
```

(\STRINGWIDTH.PRESS

```
  [LAMBDA (STREAM STRING RDTBL)                                    (* rmk%: "24-Feb-86 09:49")
                                                                   (* Returns the width of STRING in the press STREAM,
                                                                   observing spacefactor)
    (\STRINGWIDTH.GENERIC STRING (ffetch PRLOGICALFONT of (ffetch IMAGEDATA of STREAM))
           RDTBL
           (ffetch PRSPACEWIDTH of (ffetch IMAGEDATA of STREAM])
```

(\CHARWIDTH.PRESS

```
  [LAMBDA (STREAM CHARCODE)                                        (* rmk%: "24-Feb-86 09:49")
                                                                   (* Gets the width of CHARCODE in a Press STREAM,
                                                                   observing spacefactor)
    (COND
       ((EQ CHARCODE (CHARCODE SPACE))
        (ffetch PRSPACEWIDTH of (ffetch IMAGEDATA of STREAM)))
       (T (\FGETCHARWIDTH (ffetch PRLOGICALFONT of (ffetch IMAGEDATA of STREAM))
                 CHARCODE])
```

(\OUTCHARFN.PRESS

```
  [LAMBDA (PRSTREAM CHARCODE)                                      (* rmk%: "24-Feb-86 12:18")
                                                                   (* Handle all the special-purpose characters going to a PRESS
                                                                   file)
    (SELCHARQ CHARCODE
         (EOL                                                      (* New Line)
              (NEWLINE.PRESS PRSTREAM)
              (replace (STREAM CHARPOSITION) of PRSTREAM with 0))
         (LF                                                       (* Line feed--move down, but not over)
             (\DSPXPOSITION.PRESS PRSTREAM (PROG1 (DSPXPOSITION NIL PRSTREAM)
                                                  (NEWLINE.PRESS PRSTREAM))))
         (^L                                                       (* Form Feed)
             (replace (STREAM CHARPOSITION) of PRSTREAM with 0)
             (NEWPAGE.PRESS PRSTREAM))
         (PROG (XPOS NEWXPOS CLIPPINGREGION PRCHARCODE TRANSLATION (CHARSET (\CHARSET CHARCODE))
                     (PRDATA (fetch IMAGEDATA of PRSTREAM)))
               [if (NEQ CHARSET (ffetch PRLOGICALCHARSET of PRDATA))
                   then (LET [(CSINFO (\GETCHARSETINFO CHARSET (ffetch PRLOGICALFONT of PRDATA]
                             (UNINTERRUPTABLY
```

```
                                (freplace PRWIDTHSCACHE of PRDATA with (fetch (CHARSETINFO WIDTHS) of CSINFO))
                                (freplace PRTRANSLATIONCACHE of PRDATA with (fetch (CHARSETINFO CHARSETBITMAP)
                                                                                   of CSINFO))
                                (freplace PRLOGICALCHARSET of PRDATA with CHARSET))]
                        (SETQ TRANSLATION (\GETBASEPTR (ffetch PRTRANSLATIONCACHE of PRDATA)
                                                (UNFOLD (\CHAR8CODE CHARCODE)
                                                        2)))
                        (if (NEQ (CAR TRANSLATION)
                                 (fetch PRFONT of PRDATA))
                            then (\DSPFONT.PRESSFONT PRSTREAM (CAR TRANSLATION)))
                        (SETQ PRCHARCODE (CDR TRANSLATION))
                        (SETQ XPOS (fetch PRXPOS of PRDATA))
                        [SETQ NEWXPOS (IPLUS XPOS (COND
                                                      ((EQ CHARCODE (CHARCODE SPACE))
                                                       (ffetch PRSPACEWIDTH of PRDATA))
                                                      (T (\FGETWIDTH (ffetch (PRESSDATA PRWIDTHSCACHE) of PRDATA)
                                                                     (\CHAR8CODE CHARCODE]
                        (COND
                           ((AND [IGEQ XPOS (fetch LEFT of (SETQ CLIPPINGREGION (fetch PRClippingRegion of PRDATA]
                                 (ILEQ NEWXPOS (fetch RIGHT of CLIPPINGREGION))
                                 (IGEQ (fetch PRYPOS of PRDATA)
                                       (fetch BOTTOM of CLIPPINGREGION)))  (* Bottom test should really subtract off the descent, and also
                                                                           should do a top-test)
                                                                           (* The Y-tests can probably be done inside SETXY, SETY, and
                            DSPFONT.)
                            [COND
                               ((NOT (ffetch CHARWASDISPLAYING of PRDATA))  (* Was being clipped, now not)
                                (freplace CHARWASDISPLAYING of PRDATA with T)
                                (SHOW.PRESS PRSTREAM)                       (* SHOW shouldn't be necessary, but |...|)
                                (SETXY.PRESS PRSTREAM XPOS (fetch PRYPOS of PRDATA]
                            (\BOUT PRSTREAM PRCHARCODE))
                           (T (SHOW.PRESS PRSTREAM)                         (* Don't put out any characters if out of the clipping region)
                              (freplace CHARWASDISPLAYING of PRDATA with NIL)))
                        (replace PRXPOS of PRDATA with NEWXPOS])

)


          (* * new declaration for PRESSDATA)

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

[DATATYPE PRESSDATA (PRHEADING                                  ; The string to be printed atop each page.
                           PRHEADINGFONT                        ; Font to print the heading in
                           PRXPOS                               ; Current X position
                           PRYPOS                               ; Current Y position
                           PRFONT                               ; Current font
                           PRCURRFDE PRESSFONTDIR (PRWIDTHSCACHE POINTER
                                                                ; Widths table for the current logical character set
                                                 )
                           PRCOLOR PRLINEFEED PRPAGESTATE PDSTREAM ELSTREAM XPRPAGEREGION PRDOCNAME (PRLEFT
                                                                                                     WORD)
                                                                ; Page left margin
                           (PRBOTTOM WORD)                      ; Page bottom margin
                           (PRRIGHT WORD)                       ; Page right margin
                           (PRTOP WORD)                         ; Page top margin
                           (PRPAGENUM WORD)                     ; Current Page number
                           (PRNEXTFONT# BYTE)
                           (PRMAXFONTSET BYTE)
                           (PRPARTSTART INTEGER)
                           (DLSTARTBYTE INTEGER)
                           (ELSTARTBYTE INTEGER)
                           (STARTCHARBYTE INTEGER)
                           (VECMOVINGRIGHT FLAG)                ; If we're drawing a curve with vector fonts, are we moving to the
                                                                ; right?
                           (VECWASDISPLAYING FLAG)
                           ;; Used during curve/line clipping to remember whether we were on-screen or not, so we know when to force a
                           ;; SETXY.
                           VECSEGCHARS                          ; Cache for vector characters while we're moving to the left.
                           VECCURX                              ; Current X position within vector code, in Dover spots
                           VECCURY                              ; Current Y position with vector code, in Dover spots
                           PRSPACEFACTOR PRSPACEWIDTH (CHARWASDISPLAYING FLAG)
                                                                ; Says whether we have been printing characters inside the
                                                                ; clipping region
                           PRClippingRegion
                           ;; The edges of the paper, as far as PRESS is concerned.  Used to protect SPRUCE users who get killed when the
                           ;; image goes off-paper
                           PRLOGICALFONT                        ; Current logical font
                           PRLOGICALCHARSET                     ; Current logical character set, whose info is cached.  NIL if
                                                                ; cache is invalid
                           (PRTRANSLATIONCACHE POINTER          ; Translation table for the current logical character set
                                               ))
      PRSPACEFACTOR _ 1 PRXPOS _ 0 PRYPOS _ 0                   ; We assume that the origin is translated to the bottom-left of the
                                                                ; page region
```

```
       PRClippingRegion _ (create REGION
                                 LEFT _ SPRUCEPAPERLEFTMICAS
                                 BOTTOM _ SPRUCEPAPERBOTTOMMICAS
                                 WIDTH _ (DIFFERENCE SPRUCEPAPERRIGHTMICAS SPRUCEPAPERLEFTMICAS)
                                 HEIGHT _ 29210)
       (ACCESSFNS ((PRWIDTH (IDIFFERENCE (fetch (PRESSDATA PRRIGHT) of DATUM)
                                         (fetch (PRESSDATA PRLEFT) of DATUM)))
                   (PRHEIGHT (IDIFFERENCE (fetch (PRESSDATA PRTOP) of DATUM)
                                          (fetch (PRESSDATA PRBOTTOM) of DATUM)))
                   (PRPAGEREGION (fetch (PRESSDATA XPRPAGEREGION) of DATUM)
                          (PROGN (replace (PRESSDATA XPRPAGEREGION) of DATUM with NEWVALUE)
                                 (replace (PRESSDATA PRLEFT) of DATUM with (fetch (REGION LEFT) of NEWVALUE))
                                 (replace (PRESSDATA PRBOTTOM) of DATUM with (fetch (REGION BOTTOM) of NEWVALUE))
                                 (replace (PRESSDATA PRRIGHT) of DATUM with (IPLUS (fetch (REGION LEFT)
                                                                                      of NEWVALUE)
                                                                                   (fetch (REGION WIDTH)
                                                                                      of NEWVALUE)))
                                 (replace (PRESSDATA PRTOP) of DATUM with (IPLUS (fetch (REGION BOTTOM)
                                                                                    of NEWVALUE)
                                                                                 (fetch (REGION HEIGHT)
                                                                                    of NEWVALUE]
)

(/DECLAREDATATYPE 'PRESSDATA
       '(POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER
              POINTER POINTER WORD WORD WORD WORD BYTE BYTE FIXP FIXP FIXP FIXP FLAG FLAG POINTER POINTER
              POINTER POINTER POINTER FLAG POINTER POINTER POINTER POINTER)
       ;; ---field descriptor list elided by lister---
       '62)
)

(/DECLAREDATATYPE 'PRESSDATA
       '(POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER
              POINTER POINTER WORD WORD WORD WORD BYTE BYTE FIXP FIXP FIXP FIXP FLAG FLAG POINTER POINTER
              POINTER POINTER POINTER FLAG POINTER POINTER POINTER POINTER)
       ;; ---field descriptor list elided by lister---
       '62)


          (* * NSTOASCIITRANSLATIONS is a list with elements of the form
          (charset translationArrayName)%, where translationArrayName is bound to a translation array for charset which contains
          (fontFamily charcode) lists)

(DEFINEQ

(\NSTOASCIIARRAY
  [LAMBDA (CHARSET)                                  (* thh%: "17-Feb-86 09:05")
                                                     (* gets the translation array to use for this charset)
    (EVAL (CADR (ASSOC CHARSET NSTOASCIITRANSLATIONS])

(\NSTOASCIITRANSLATION
  [LAMBDA (TRANSLATION FAMILY FONTDESC)              (* thh%: " 5-Mar-86 10:23")
                                                     (* returns (fontdesc . charcode) to use in place of the specified
                                                     8-bit charcode)

          (* FAMILY, if specified, is font family to use when not specified by the translation array)

          (* * determine the (family charcode) translation)

    (OR TRANSLATION (SETQ TRANSLATION unknownCharTranslation))
    [COND
       ((FIXP TRANSLATION)
        (SETQ TRANSLATION (LIST (OR FAMILY FONTDESC)
                                TRANSLATION]

          (* * coerce to a full font descriptor)

    (CONS (\PRESS.COERCEFONT FONTDESC (CAR TRANSLATION))
          (CADR TRANSLATION])

)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS NSTOASCIITRANSLATIONS PRESSFONTFAMILIES)
)

(RPAQ? PRESSFONTFAMILIES '((GACHA)
                           (TIMESROMAN)
                           (HELVETICA)
                           (SYMBOL)
                           (MATH)
                           (HIPPO)
                           (CYRILLIC)
```

```
                            (NEWVEC)
                            (SNEWVEC)
                            (HNEWVEC)
                            (VNEWVEC)))

(RPAQ? NSTOASCIITRANSLATIONS )

(ADDTOVAR NSTOASCIITRANSLATIONS (0 ASCIIFROM0ARRAY)
                                (38 ASCIIFROM38ARRAY)
                                (39 ASCIIFROM39ARRAY)
                                (239 ASCIIFROM239ARRAY))

(READVARS-FROM-STRINGS '(ASCIIFROM0ARRAY ASCIIFROM38ARRAY ASCIIFROM39ARRAY ASCIIFROM239ARRAY)
       "({Y256 POINTER 0 {R163 NIL}  (SYMBOL 126)  (SYMBOL 127) NIL NIL  (SYMBOL 120) NIL 96 NIL NIL  (SYMBOL
       55)  (SYMBOL 34)  (SYMBOL 33)  (SYMBOL 35) NIL  (SYMBOL 6) NIL NIL  (SYMBOL 2) NIL  (SYMBOL 123) NIL
       (SYMBOL 13) 39 {R25 NIL}  (SYMBOL 125) {R44 NIL} }  {Y256 POINTER 0  (HIPPO 118) {R64 NIL}  (HIPPO 65)
         (HIPPO 66) NIL  (HIPPO 71)  (HIPPO 68)  (HIPPO 69) NIL NIL  (HIPPO 90)  (HIPPO 72)  (HIPPO 81)  (
       HIPPO 73)  (HIPPO 75)  (HIPPO 76)  (HIPPO 77)  (HIPPO 78)  (HIPPO 67)  (HIPPO 79)  (HIPPO 80) NIL  (
       HIPPO 82)  (HIPPO 83) NIL  (HIPPO 84)  (HIPPO 85)  (HIPPO 70)  (HIPPO 88)  (HIPPO 89)  (HIPPO 87) NIL
       NIL NIL  (HIPPO 97)  (HIPPO 98) NIL  (HIPPO 103)  (HIPPO 100)  (HIPPO 101) NIL NIL  (HIPPO 122)  (
       HIPPO 104)  (HIPPO 113)  (HIPPO 105)  (HIPPO 107)  (HIPPO 108)  (HIPPO 109)  (HIPPO 110)  (HIPPO 99)
       (HIPPO 111)  (HIPPO 112) NIL  (HIPPO 114)  (HIPPO 115)  (HIPPO 106)  (HIPPO 116)  (HIPPO 117)  (HIPPO
       102)  (HIPPO 120)  (HIPPO 121)  (HIPPO 119) {R130 NIL} }  {Y256 POINTER 0  (CYRILLIC 127) {R32 NIL}  (
       CYRILLIC 65)  (CYRILLIC 66)  (CYRILLIC 86)  (CYRILLIC 71)  (CYRILLIC 68)  (CYRILLIC 69)  (CYRILLIC 36)
         (CYRILLIC 87)  (CYRILLIC 90)  (CYRILLIC 73)  (CYRILLIC 74)  (CYRILLIC 75)  (CYRILLIC 76)  (CYRILLIC
       77)  (CYRILLIC 78)  (CYRILLIC 79)  (CYRILLIC 80)  (CYRILLIC 82)  (CYRILLIC 83)  (CYRILLIC 84)  (
       CYRILLIC 85)  (CYRILLIC 70)  (CYRILLIC 81)  (CYRILLIC 126)  (CYRILLIC 42)  (CYRILLIC 123)  (CYRILLIC
       125)  (CYRILLIC 94)  (CYRILLIC 88)  (CYRILLIC 67)  (CYRILLIC 64)  (CYRILLIC 89)  (CYRILLIC 72) {R15
       NIL}  (CYRILLIC 97)  (CYRILLIC 98)  (CYRILLIC 118)  (CYRILLIC 103)  (CYRILLIC 100)  (CYRILLIC 101)  (
       CYRILLIC 52)  (CYRILLIC 119)  (CYRILLIC 122)  (CYRILLIC 105)  (CYRILLIC 106)  (CYRILLIC 107)  (
       CYRILLIC 108)  (CYRILLIC 109)  (CYRILLIC 110)  (CYRILLIC 111)  (CYRILLIC 112)  (CYRILLIC 114)  (
       CYRILLIC 115)  (CYRILLIC 116)  (CYRILLIC 117)  (CYRILLIC 102)  (CYRILLIC 113)  (CYRILLIC 54)  (
       CYRILLIC 56)  (CYRILLIC 91)  (CYRILLIC 93)  (CYRILLIC 95)  (CYRILLIC 120)  (CYRILLIC 143)  (CYRILLIC
       50)  (CYRILLIC 121)  (CYRILLIC 104) {R12 NIL}  (CYRILLIC 99) {R129 NIL} }  {Y256 POINTER 0 {R36 NIL}
       (TIMESROMAN 155)  (TIMESROMAN 156) {R6 NIL}  (TIMESROMAN 152)  (TIMESROMAN 153) NIL  (TIMESROMAN 159)
        (MATH 33)  (MATH 70)  (SYMBOL 104)  (SYMBOL 105) NIL NIL  (SYMBOL 96)  (SYMBOL 97)  (MATH 113) NIL  (
       SYMBOL 109)  (SYMBOL 108)  (MATH 116)  (MATH 118)  (MATH 115)  (MATH 117)  (MATH 64) NIL  (SYMBOL 37)
        (SYMBOL 38) {R4 NIL}  (MATH 109) NIL  (MATH 66)  (MATH 78)  (MATH 44)  (SYMBOL 40)  (SYMBOL 44)  (
       SYMBOL 41)  (MATH 126)  (MATH 81)  (SYMBOL 36)  (MATH 98) NIL NIL  (SYMBOL 92)  (SYMBOL 91)  (SYMBOL
       19)  (SYMBOL 18)  (SYMBOL 27)  (SYMBOL 26) NIL NIL  (MATH 75)  (MATH 72) NIL  (MATH 79)  (SYMBOL 8)  (
       SYMBOL 9)  (MATH 54)  (SYMBOL 11)  (TIMESROMAN 183)  (SYMBOL 5)  (MATH 104) NIL  (SYMBOL 58) NIL  (
       SYMBOL 54) NIL NIL  (MATH 22)  (SYMBOL 16)  (MATH 80)  (SYMBOL 17)  (SYMBOL 29) NIL  (SYMBOL 115)  (
       MATH 7)  (SYMBOL 39) NIL  (SYMBOL 25)  (MATH 19)  (MATH 1)  (SYMBOL 112)  (SYMBOL 7) {R41 NIL}  (
       SYMBOL 59) {R6 NIL}  (MATH 82) NIL  (SYMBOL 100)  (SYMBOL 101)  (SYMBOL 98)  (SYMBOL 99)  (SYMBOL 57)
        (SYMBOL 56)  (SYMBOL 94)  (SYMBOL 95)  (MATH 90)  (MATH 68)  (MATH 100) {R69 NIL} })
       ")

(\SMASHPRESSFONTS)

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

(RPAQQ unknownCharTranslation (MATH 59))

[CONSTANTS (unknownCharTranslation '(MATH 59]
)
)

(PUTPROPS PRESSFROMNS COPYRIGHT ("Xerox Corporation" 1986 1988))
```

FUNCTION INDEX

VARIABLE INDEX

CONSTANT INDEX

RECORD INDEX