File created: 12-Jul-2022 15:09:31 **{DSK}<Users>kaplan>Local>medley3.5>working-medley>sources>FONT.;11**

changes to: (FNS \INSTALLCHARSETINFO \CREATECHARSET WRITESTRIKEFONTFILE \READSTRIKEFONTFILE)
(VARS FONTCOMS)

previous date: 11-Jul-2022 23:05:20 {DSK}<Users>kaplan>Local>medley3.5>working-medley>sources>FONT.;3

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

```
;;
;; Copyright (c) 1981-1994, 1999, 2021 by Venue & Xerox Corporation.

(RPAQQ FONTCOMS
       [;; font functions
        (FNS CHARWIDTH CHARWIDTHY STRINGWIDTH \CHARWIDTH.DISPLAY \STRINGWIDTH.DISPLAY \STRINGWIDTH.GENERIC)
        (FNS DEFAULTFONT FONTCLASS FONTCLASSUNPARSE FONTCLASSCOMPONENT SETFONTCLASSCOMPONENT)
        [COMS                                                 ; Until we pin down the exact interface
              (P (MOVD 'FONTCLASSCOMPONENT 'FONTCOMPONENT)
                 (MOVD 'SETFONTCLASSCOMPONENT 'SETFONTCOMPONENT]
        [COMS                                                 ; MAPPING FOR DOS FILENAMES
              (INITVARS (*DISPLAY-FONT-NAME-MAP* '((TIMESROMAN . TR)
                                                   (HELVETICA . HV)
                                                   (TIMESROMAND . TD)
                                                   (HELVETICAD . HD)
                                                   (MODERN . MD)
                                                   (CLASSIC . CL)
                                                   (GACHA . GC)
                                                   (TITAN . TI)
                                                   (LETTERGOTHIC . LG)
                                                   (BOLDPS . BP)
                                                   (TERMINAL . TM)
                                                   (CLASSICTHIN . CT)
                                                   (HIPPO . HP)
                                                   (LOGO . LG)
                                                   (MATH . MA)
                                                   (OLDENGLISH . OE)
                                                   (SYMBOL . SY)]
        (COMS ;; Creation:
              (FNS FONTCREATE \FONT.SYMBOLMEMB \FONT.SYMBOLASSOC \FONT.COMPARESYMBOL))
        (COMS ;; Property extraction:
              (FNS FONTASCENT FONTDESCENT FONTHEIGHT FONTPROP \AVGCHARWIDTH))
        (COMS ;; Bitmap editing/manipulation:
              (FNS GETCHARBITMAP PUTCHARBITMAP MOVECHARBITMAP))
        (FNS FONTCOPY FONTSAVAILABLE FONTFILEFORMAT FONTP FONTUNPARSE SETFONTDESCRIPTOR CHARCODEP EDITCHAR
             \STREAMCHARWIDTH \UNITWIDTHSVECTOR \CREATEDISPLAYFONT \CREATECHARSET.DISPLAY
             \CREATE-REAL-CHARSET.DISPLAY \BUILDSLUGCSINFO \SEARCHDISPLAYFONTFILES \SEARCHFONTFILES
             \FINDFONTFILE \FONTSYMBOL \DEVICESYMBOL \FONTFACE \FONTFACE.COLOR \FONTFILENAME \FONTFILENAME.OLD
             \FONTFILENAME.NEW \FONTINFOFROMFILENAME \FONTINFOFROMFILENAME.OLD \GETFONTDESC \COERCEFONTDESC
             \LOOKUPFONT \LOOKUPFONTSINCORE \READDISPLAYFONTFILE)
        (COMS ;; \FINDFONTFILE \FONTFILENAME \SEARCHFONTFILES \FONTINFOFROMFILENAME are redefined to deal with character-set
              ;; directories.  That behavior is conditioned on the setting of the global variable *USEOLDFONTDIRECTORIES*, T at PARC, maybe
              ;; NIL most other places.
              (ADDVARS (*OLD-FONT-EXTENSIONS* STRIKE))
              (INITVARS (*USEOLDFONTDIRECTORIES* NIL))
              (GLOBALVARS *OLD-FONT-EXTENSIONS* *USEOLDFONTDIRECTORIES*)
              ;; Establishes DISPLAYFONTFILECACHE to avoid rereading charsets when size coercions are done (e.g. for nsdisplaysizes or
              ;; smallscreen)
              )
        (COMS ;; Establishes DISPLAYFONTFILECACHE to avoid rereading charsets when size coercions are done (e.g. for nsdisplaysizes or
              ;; smallscreen)
              (INITVARS (CACHEDISPLAYFONTS))
              (GLOBALVARS CACHEDISPLAYFONTS)
                                                              ; STRIKE format file support
              (FNS \READSTRIKEFONTFILE \SFMAKEBOLD \SFMAKEITALIC \SFMAKEROTATEDFONT \SFROTATECSINFO
                   \SFROTATEFONTCHARACTERS \SFFIXOFFSETSAFTERROTATION \SFROTATECSINFOOFFSETS \SFMAKECOLOR)
              (FNS WRITESTRIKEFONTFILE STRIKECSINFO))
        (INITRECORDS FONTCLASS FONTDESCRIPTOR CHARSETINFO)
        (SYSRECORDS FONTCLASS FONTDESCRIPTOR CHARSETINFO)
        (INITVARS (\FONTSINCORE)
                  (\DEFAULTDEVICEFONTS)
                  (\UNITWIDTHSVECTOR))
        (GLOBALVARS DISPLAYFONTDIRECTORIES \DEFAULTDEVICEFONTS \UNITWIDTHSVECTOR)
        (DECLARE%: DONTEVAL@LOAD DOCOPY (P (\UNITWIDTHSVECTOR)))
        (CONSTANTS (NORUNCODE 255))
        (EXPORT (OPTIMIZERS FONTPROP))
        [DECLARE%: DONTCOPY (EXPORT (RECORDS FONTCLASS FONTDESCRIPTOR FONTFACE CHARSETINFO)
```

```
                                (MACROS FONTASCENT FONTDESCENT FONTHEIGHT \FGETOFFSET \FSETOFFSET \FGETWIDTH
                                       \FSETWIDTH \FGETCHARWIDTH \FSETCHARWIDTH \FGETIMAGEWIDTH
                                       \FSETIMAGEWIDTH \GETCHARSETINFO \CREATECSINFOELEMENT
                                       \CREATEFONTCHARSETVECTOR)
                                (FUNCTIONS \CREATEKERNELEMENT \FSETLEFTKERN \FGETLEFTKERN)
                                (CONSTANTS (\MAXNSCHAR 65535]
        (COMS                                                    ; NS Character specific code
              (FNS \CREATECHARSET \INSTALLCHARSETINFO)
              (GLOBALVARS DISPLAYFONTCOERCIONS MISSINGDISPLAYFONTCOERCIONS MISSINGCHARSETDISPLAYFONTCOERCIONS
                     CHARSETERRORFLG)
              (INITVARS (DISPLAYFONTCOERCIONS NIL)
                     [MISSINGCHARSETDISPLAYFONTCOERCIONS '(((GACHA)
                                                           (TERMINAL))
                                                          ((MODERN)
                                                           (CLASSIC))
                                                          ((TIMESROMAN)
                                                           (CLASSIC))
                                                          ((HELVETICA)
                                                           (MODERN))
                                                          ((TERMINAL 6)
                                                           (MODERN 6))
                                                          ((TERMINAL 8)
                                                           (MODERN 8))
                                                          ((TERMINAL 10)
                                                           (MODERN 10))
                                                          ((TERMINAL 12)
                                                           (MODERN 12]
                     [MISSINGDISPLAYFONTCOERCIONS '(((GACHA)
                                                     (TERMINAL))
                                                    ((MODERN)
                                                     (CLASSIC))
                                                    ((TIMESROMAN)
                                                     (CLASSIC))
                                                    ((HELVETICA)
                                                     (MODERN))
                                                    ((TERMINAL)
                                                     (MODERN]
                     (CHARSETERRORFLG NIL)
                     (\DEFAULTCHARSET 0)))
        (FNS \FONTRESETCHARWIDTHS)
        [DECLARE%: DONTEVAL@LOAD (INITVARS (DISPLAYFONTEXTENSIONS 'DISPLAYFONT)
                                       (DISPLAYFONTDIRECTORIES '(
                                                                 {DSK}/USR/LOCAL/LDE/FONTS/DISPLAY/PRESENTATION/
                                                                 {dsk}/usr/local/lde/fonts/display/publishing/]
        (MACROS \FGETCHARIMAGEWIDTH \GETFONTDESC \SETCHARSETINFO)
        (LOCALVARS . T)
        (PROP FILETYPE FONT)
        (DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS (ADDVARS (NLAMA)
                                                                              (NLAML)
                                                                              (LAMA FONTCOPY])
```

;; font functions

```
(DEFINEQ
```

(**CHARWIDTH**

```
  [LAMBDA (CHARCODE FONT)                                    (* rmk%: "12-Apr-85 09:46")
                                                             ; gets the width of a character code in a font/stream
    (OR (\CHARCODEP CHARCODE)
        (\ILLEGAL.ARG CHARCODE))
    (LET (TEMP)
         (COND
            ((type? FONTDESCRIPTOR FONT)
             (\FGETCHARWIDTH FONT CHARCODE))
            ((SETQ TEMP (\OUTSTREAMARG FONT T))              ; NIL font goes thru here--primary output file
             (IMAGEOP 'IMCHARWIDTH TEMP TEMP CHARCODE))
            (T (\FGETCHARWIDTH (FONTCREATE FONT)
                      CHARCODE])
```

(**CHARWIDTHY**

```
  [LAMBDA (CHARCODE FONT)                                    (* edited%: "18-Mar-86 19:30")
                                                             ; Gets the Y-component of the width of a character code in a
                                                             ; font.
    (OR (\CHARCODEP CHARCODE)
        (\ILLEGAL.ARG CHARCODE))
    (LET (TEMP WY)
         (COND
            ((type? FONTDESCRIPTOR FONT)
             (SETQ WY (ffetch (CHARSETINFO YWIDTHS) of (\GETCHARSETINFO (\CHARSET CHARCODE)
                                                                FONT)))
             (COND
                ((FIXP WY))
                (WY (\FGETWIDTH WY (\CHAR8CODE CHARCODE)))
                (T 0)))
            ((type? STREAM (SETQ TEMP (\OUTSTREAMARG FONT T)))    ; NIL font goes thru here--primary output file
             (IMAGEOP 'IMCHARWIDTHY TEMP TEMP CHARCODE))
```

```
            (T [SETQ WY (ffetch (CHARSETINFO YWIDTHS) of (\GETCHARSETINFO (\CHARSET CHARCODE)
                                                                (FONTCREATE FONT]
               (COND
                  ((FIXP WY))
                  (WY (\FGETWIDTH WY (\CHAR8CODE CHARCODE)))
                  (T 0])
```

(STRINGWIDTH

```
[LAMBDA (STR FONT FLG RDTBL)                                ; Edited 8-Jan-88 14:41 by Snow
```

;; Returns the width of STR according to FONT

```
(LET (TEMP)
```

;; Used in \MAPCHARS

```
     (COND
        [(type? FONTDESCRIPTOR FONT)
         (\STRINGWIDTH.GENERIC STR FONT (AND FLG (\GTREADTABLE RDTBL))
                (\FGETCHARWIDTH FONT (CHARCODE SPACE]
        [(AND FONT (SETQ TEMP (\OUTSTREAMARG FONT T)))      ; if you gave something for FONT, coerce it to a stream, and call
                                                            ; the stringwidth function of it.
         (IMAGEOP 'IMSTRINGWIDTH TEMP TEMP STR (AND FLG (\GTREADTABLE RDTBL]
        (T (SETQ TEMP (FONTCREATE (OR FONT DEFAULTFONT)))   ; NIL font will pass thru here. ie, defaultfont is used to do the
                                                            ; stringwidth instead of the font of *standard-output*
           (\STRINGWIDTH.GENERIC STR TEMP (AND FLG (\GTREADTABLE RDTBL))
                  (\FGETCHARWIDTH TEMP (CHARCODE SPACE])
```

(\CHARWIDTH.DISPLAY

```
[LAMBDA (STREAM CHARCODE)                                   (* rmk%: "12-Apr-85 09:42")
                                                            ; gets the width of a character code in a display stream. Need to
                                                            ; fix up for spacefactor.
   (\FGETCHARWIDTH (ffetch (\DISPLAYDATA DDFONT) of (ffetch IMAGEDATA of STREAM))
          CHARCODE])
```

(\STRINGWIDTH.DISPLAY

```
[LAMBDA (STREAM STR RDTBL)                                  ; Edited 3-Apr-87 12:07 by jop
```

;; Returns the width of for the current font/spacefactor in STREAM.

```
(LET ((DD (ffetch IMAGEDATA of STREAM)))
     (\STRINGWIDTH.GENERIC STR (ffetch (\DISPLAYDATA DDFONT) of DD)
            RDTBL
            (ffetch DDSPACEWIDTH of DD])
```

(\STRINGWIDTH.GENERIC

```
[LAMBDA (STR FONT RDTBL SPACEWIDTH)                         ; Edited 3-Apr-87 13:47 by jop
```

;; Returns the width of STR with SPACEWIDTH for the width of spaces. RDTBL has already been coerced, so no FLG is needed

;; This is cloned in \STRINGWIDTH.HCPYDISPLAYAUX by straight substitution -- (PUTDEF (QUOTE \STRINGWIDTH.HCPYDISPLAYAUX)
;; (QUOTE FNS) (SUBLIS (QUOTE ((WIDTHS . IMAGEWIDTHS) (\FGETWIDTH . \FGETIMAGEWIDTH) (\FGETCHARWIDTH .
;; \FGETCHARIMAGEWIDTH))) (GETDEF (QUOTE \STRINGWIDTH.GENERIC))))

;; \MAPPNAME uses WIDTHSBASE CSET TOTALWIDTH FONT SPACEWIDTH free, so these become special in bytecompiler

```
(PROG NIL
      [COND
         [(LITATOM STR)
          (if RDTBL
              then (GO SLOW)
            else (RETURN (for C WIDTHSBASE CSET inatom STR sum [COND
                                                                  ((NEQ CSET (\CHARSET C))
                                                                   (SETQ CSET (\CHARSET C))
                                                                   (SETQ WIDTHSBASE (ffetch (CHARSETINFO WIDTHS
                                                                                                        )
                                                                                       of (\GETCHARSETINFO
                                                                                             CSET FONT]
                                                               (COND
                                                                  ((EQ C (CHARCODE SPACE))
                                                                   SPACEWIDTH)
                                                                  (T (\FGETWIDTH WIDTHSBASE (\CHAR8CODE C]
         ((STRINGP STR)
          (RETURN (LET ((TOTAL 0)
                        ESC ESCWIDTH WIDTHSBASE CSET)
                       [COND
                          (RDTBL                            ; Count delimiting quotes and internal escapes
                                 (SETQ TOTAL (UNFOLD (\FGETCHARWIDTH FONT (CHARCODE %"))
                                                     2))
                                 (SETQ ESC (fetch (READTABLEP ESCAPECHAR) of RDTBL))
                                 (SETQ ESCWIDTH (\FGETCHARWIDTH FONT ESC]
                       [for C instring STR
                          do [COND
                                ((NEQ (\CHARSET C)
                                      CSET)                 ; Get the widths vector for this character set
                                 (SETQ CSET (\CHARSET C))
                                 (SETQ WIDTHSBASE (ffetch (CHARSETINFO WIDTHS) of (\GETCHARSETINFO CSET FONT]
                             (add TOTAL (COND
```

```
                                          ((EQ C (CHARCODE SPACE))
                                           SPACEWIDTH)
                                          (T (IPLUS (\FGETWIDTH WIDTHSBASE (\CHAR8CODE C))
                                                    (COND
                                                       ((AND RDTBL (OR (EQ C (CHARCODE %"))
                                                                       (EQ C ESC)))
                                                            ; String char must be escaped
                                                        ESCWIDTH)
                                                       (T 0]
                           TOTAL]
      SLOW
                                                                ; Do the general case here
          (RETURN (LET ((TOTALWIDTH 0)
                        WIDTHSBASE CSET (FONT FONT)
                        (SPACEWIDTH SPACEWIDTH))
                       (DECLARE (SPECVARS TOTALWIDTH WIDTHSBASE CSET FONT SPACEWIDTH))
                       (\MAPPNAME [FUNCTION (LAMBDA (DUMMY CC)
                                            (add TOTALWIDTH (COND
                                                               ((EQ CC (CHARCODE SPACE))
                                                                SPACEWIDTH)
                                                               ((EQ CSET (\CHARSET CC))
                                                                (\FGETWIDTH WIDTHSBASE (\CHAR8CODE CC)))
                                                               (T (SETQ CSET (\CHARSET CC))
                                                                  (SETQ WIDTHSBASE (ffetch (CHARSETINFO WIDTHS)
                                                                                      of (\GETCHARSETINFO
                                                                                          CSET FONT)))
                                                                  (\FGETWIDTH WIDTHSBASE (\CHAR8CODE CC]
                                  STR RDTBL RDTBL *PRINT-LEVEL* *PRINT-LENGTH*)
                       TOTALWIDTH])

)

(DEFINEQ
```

(**DEFAULTFONT**

```
  [LAMBDA (DEVICE FONT NOERRORFLG)                          ; Edited 28-Jul-88 13:15 by rmk:
                                                            ; Edited 24-Mar-87 14:41 by FS
```

;; Returns the default font for an image type. Really only needed to guarantee validity of the display default font for system critical routines, in case ;; the user has smashed the variable DEFAULTFONT. Note that SETFONTCLASSCOMPONENT and FONTCLASS guarantee that the display ;; component is either NIL or a fontdescriptor.

;; FS- If FONT provided set the font descriptor. Do not bother to check if NOERRORFLG is NEW. (old code had (AND FONT (EQ NOERRORFLG ;; 'NEW)))

```
    [OR (type? FONTCLASS DEFAULTFONT)
        (SETQ DEFAULTFONT (FONTCLASS 'DEFAULTFONT]
    (if FONT
        then ;; FS- Not clear the fontclass should be smashed, perhaps instead should make a new FONTCLASS and then rebind DEFAULTFONT.
             ;; Leaving alone for histerical reasons
             (SETFONTCLASSCOMPONENT DEFAULTFONT DEVICE FONT)
      else ;; The code below (not mine!) is messy but is correct (unless weirdness pops up because of deep recursion).
           (COND
              ((\COERCEFONTDESC DEFAULTFONT DEVICE T))
              (NOERRORFLG NIL)
              ((EQ (\DEVICESYMBOL DEVICE T)
                   'DISPLAY)
               ;; If getting for the display and the font can't be found perhaps because of garbage in the display field of the DEFAULTFONTCLASS,
               ;; then the system-guaranteed displayfont. Otherwise, cause the error in the re-coercion. Can never tell when DEVICE is just a
               ;; symbol.
               \GUARANTEEDDISPLAYFONT)
              ((\COERCEFONTDESC DEFAULTFONT DEVICE])
```

(**FONTCLASS**

```
  [LAMBDA (NAME FONTLIST CREATEFORDEVICES)                  (* jds " 9-Sep-86 18:49")
```

;; This builds D style font classes, which are datatypes containing entries for the various known devices.

;; Don't actually set up the for devices not inside CREATEFORDEVICES on the theory that any given user presumably doesn't want all the fonts for ;; all the devices. We wait until he actually asks for the font or the fontmaparray, at which point we note that the fields don't contain FD's, so we ;; then apply FONTCREATE. The actual coercion and caching is done inside \COERCEFONTDESC. However, so as to prevent display crashes, ;; if a display component is specified, we always do the fontcreate before we stick it in.

```
    (PROG (F FC FL)
          (SETQ FL FONTLIST)
          [SETQ FC (create FONTCLASS
                           FONTCLASSNAME _ NAME
                           PRETTYFONT# _ (OR (FIXP (pop FL))
                                             1)
                           DISPLAYFD _ (AND (SETQ F (pop FL))
                                            (FONTCREATE F NIL NIL NIL 'DISPLAY))
                           PRESSFD _ (pop FL)
                           INTERPRESSFD _ (pop FL)
                           OTHERFDS _ (for FSPEC in FL collect (OR (AND (LISTP FSPEC)
                                                                        (ATOM (CAR FSPEC))
                                                                        (CAR FSPEC))
```

```
                                                    (ERROR "illegal font class specification"
                                                           (LIST NAME FONTLIST)))
                                                    ; Copy the alist entry so it can be smashed in
                                                    ; \COERCEFONTDESC
                                                (CONS (CAR FSPEC)
                                                      (CAR (LISTP (CDR FSPEC]
        (for D inside CREATEFORDEVICES do (FONTCREATE FC NIL NIL NIL D))
        (RETURN FC])
```

(FONTCLASSUNPARSE

```
  [LAMBDA (FONTCLASS DEVICE FONT NOERRORFLG)                    (* jds "24-Jan-86 11:58")
                                                                ; Given a font class, unparse it to a form that might be reparsable
    (APPEND (LIST (fetch (FONTCLASS FONTCLASSNAME) of FONTCLASS)
                  (fetch (FONTCLASS PRETTYFONT#) of FONTCLASS)
                  (FONTUNPARSE (ffetch (FONTCLASS DISPLAYFD) of FONTCLASS))
                  (FONTUNPARSE (ffetch (FONTCLASS PRESSFD) of FONTCLASS))
                  (FONTUNPARSE (ffetch (FONTCLASS INTERPRESSFD) of FONTCLASS)))
            (for X in (fetch (FONTCLASS OTHERFDS) of FONTCLASS) collect (LIST (CAR X)
                                                                              (FONTUNPARSE (CDR X])
```

(FONTCLASSCOMPONENT

```
  [LAMBDA (FONTCLASS DEVICE FONT NOERRORFLG)                    (* rmk%: "14-Sep-84 19:34")
    (PROG1 (FONTCREATE FONTCLASS NIL NIL NIL DEVICE NOERRORFLG); This works its way down to \COERCEFONTDESC, where it
                                                                ; needs to be done quickly
        (AND FONT (SETQ FONT (FONTCREATE FONT NIL NIL NIL DEVICE NOERRORFLG))
             (SETFONTCLASSCOMPONENT FONTCLASS DEVICE FONT)))])
```

(SETFONTCLASSCOMPONENT

```
  [LAMBDA (FONTCLASS DEVICE FONT)                               ; Edited 29-Aug-91 12:20 by jds
    (PROG ((NEWFONT (FONTCREATE FONT NIL NIL NIL DEVICE)))
```

;; replaces will barf if FONTCLASS is not a fontclass

```
          (SELECTQ (SETQ DEVICE (FONTPROP NEWFONT 'DEVICE))
               (DISPLAY (replace (FONTCLASS DISPLAYFD) of FONTCLASS with NEWFONT))
               (INTERPRESS (replace (FONTCLASS INTERPRESSFD) of FONTCLASS with NEWFONT))
               (PRESS (replace (FONTCLASS PRESSFD) of FONTCLASS with NEWFONT))
               (RPLACD [OR (SASSOC DEVICE (fetch (FONTCLASS OTHERFDS) of FONTCLASS))
                           (CAR (push (fetch (FONTCLASS OTHERFDS) of FONTCLASS)
                                      (CONS DEVICE]
                       NEWFONT))
          (RETURN NEWFONT])

)

;; Until we pin down the exact interface

(MOVD 'FONTCLASSCOMPONENT 'FONTCOMPONENT)

(MOVD 'SETFONTCLASSCOMPONENT 'SETFONTCOMPONENT)

;; MAPPING FOR DOS FILENAMES

(RPAQ? *DISPLAY-FONT-NAME-MAP*
       '((TIMESROMAN . TR)
         (HELVETICA . HV)
         (TIMESROMAND . TD)
         (HELVETICAD . HD)
         (MODERN . MD)
         (CLASSIC . CL)
         (GACHA . GC)
         (TITAN . TI)
         (LETTERGOTHIC . LG)
         (BOLDPS . BP)
         (TERMINAL . TM)
         (CLASSICTHIN . CT)
         (HIPPO . HP)
         (LOGO . LG)
         (MATH . MA)
         (OLDENGLISH . OE)
         (SYMBOL . SY)))

;; Creation:

(DEFINEQ
```

(FONTCREATE

```
  [LAMBDA (FAMILY SIZE FACE ROTATION DEVICE NOERRORFLG CHARSET) ; Edited 10-Oct-88 09:53 by rmk:
                                                                ; Edited 28-Jul-88 14:43 by rmk:
                                                                ; Edited 10-Nov-87 18:08 by FS
```

;; Create a font descriptor for the specified font. If NOERRORFLG, return NIL if the font doesn't exist; otherwise cause an error.

;; Cache and fonts.widths traffic in uppercase only.

```
;; character set is optional and defaults to \DEFAULTCHARSET (0 in our world)
(DECLARE (GLOBALVARS IMAGESTREAMTYPES \DEFAULTCHARSET))
(PROG (FONTX (CHSET (OR CHARSET \DEFAULTCHARSET)))
      (RETURN (COND
                  ((LISTP FAMILY)
                   (SELECTQ (CAR FAMILY)
                        (FONT (SETQ FONTX (CDR FAMILY)))
                        (CLASS (COND
                                   ((LITATOM (CADR FAMILY))     ; litatom class name
                                    (RETURN (FONTCLASS (CADR FAMILY)
                                                       (CDDR FAMILY)
                                                       DEVICE)))
                                   (T                           ; Allows for a font named CLASS--distinguished cause its size is
                                                                ; not a litatom
                                      (SETQ FONTX FAMILY))))
                        (SETQ FONTX FAMILY))
                   (FONTCREATE (CAR FONTX)
                          (OR (CADR FONTX)
                              SIZE)
                          (OR (CADDR FONTX)
                              FACE)
                          (OR (CADDDR FONTX)
                              ROTATION)
                          (OR (CADR (CDDDR FONTX))
                              DEVICE)
                          NOERRORFLG CHSET))
                  ([SETQ FONTX (COND
                                   ((type? FONTDESCRIPTOR FAMILY)
                                    FAMILY)
                                   ((NULL FAMILY)
                                    (DEFAULTFONT DEVICE))
                                   ((type? FONTCLASS FAMILY)
                                   ;; We know that this won't attempt a cyclic fontcreate in \COERCEFONTDESC, because we are
                                   ;; passing a known class.  Unless NOERROFLG, an error will be caused on the actual device font if it
                                   ;; can't be found.
                                    (\COERCEFONTDESC FAMILY DEVICE NOERRORFLG))
                                   ((OR (IMAGESTREAMP FAMILY)
                                        (type? WINDOW FAMILY))
                                    (DSPFONT NIL FAMILY]
                   ;; FAMILY was a spec for a font descriptor, use it and extend it by the other args.
                   (COND
                      ((OR SIZE FACE ROTATION DEVICE)
                       (FONTCREATE (FONTPROP FONTX 'FAMILY)
                              (OR SIZE (FONTPROP FONTX 'SIZE))
                              (OR FACE (FONTPROP FONTX 'FACE))
                              (OR ROTATION (FONTPROP FONTX 'ROTATION))
                              (OR DEVICE (FONTPROP FONTX 'DEVICE))
                              NOERRORFLG))
                      (T FONTX)))
                  (T (PROG (FONTFACE (DEV DEVICE))
                       RETRY
                           [OR (LITATOM FAMILY)
                               (COND
                                  (NOERRORFLG (RETURN))
                                  (T (LISPERROR "ARG NOT LITATOM" FAMILY T]
                           [OR (AND (FIXP SIZE)
                                    (IGREATERP SIZE 0))
                               (COND
                                  (NOERRORFLG (RETURN NIL))
                                  (T (\ILLEGAL.ARG SIZE]
                           (COND
                              ((NULL ROTATION)
                               (SETQ ROTATION 0))
                              ((AND (FIXP ROTATION)
                                    (IGEQ ROTATION 0)))
                              (NOERRORFLG (RETURN NIL))
                              (T (\ILLEGAL.ARG ROTATION)))
                           [SETQ DEV (COND
                                        ((NULL DEVICE)
                                         'DISPLAY)
                                        ((AND (LITATOM DEVICE)
                                              (NEQ DEVICE T)) ; Maybe wrong case or package, but we bet it's OK and defer
                                                              ; expensive coercion until we've failed.
                                         DEV)
                                        ((SETQ DEV (\GETSTREAM DEVICE 'OUTPUT T))
                                                              ; T coerces here to primary output
                                         (fetch (IMAGEOPS IMFONTCREATE) of (fetch (STREAM IMAGEOPS)
                                                                                  of DEV)))
                                        ((STRINGP DEVICE)
                                         (MKATOM (U-CASE DEVICE)))
                                        (NOERRORFLG (RETURN NIL))
                                        (T (\ILLEGAL.ARG DEVICE]
                                                              ; DEV is now guanteed litatom
                       NEWDEV
                                                              ; Check after device since it is device-dependent
```

```
                (SETQ FONTFACE (OR (\FONTFACE FACE NOERRORFLG DEV)
                                   (RETURN NIL)))  ; Don't truly coerce to \FONTSYMBOL or \DEVICESYMBOL until
                                                   ; we've had a shot at the font cache, since re-interning atoms is
                                                   ; so expensive
                [RETURN (COND
                           ((\LOOKUPFONT FAMILY SIZE FONTFACE ROTATION DEV))
                           [(SETQ FONTX (CDR (ASSOC DEV IMAGESTREAMTYPES)))

                            ;; Device is valid, font just doesn't exist.  FONTFACE, DEV already canonical.  Make FAMILY so,
                            ;; so that each imagestream type doesn't have to.

                              (SETQ FAMILY (\FONTSYMBOL FAMILY))
                              (COND
                                 ((SETQ FONTX (APPLY* (OR (CADR (ASSOC 'FONTCREATE FONTX))
                                                          (FUNCTION NILL))
                                                      FAMILY SIZE FONTFACE ROTATION DEV CHSET))

                                  ;; default creation case.  Use fontcreate method from device, build a fontdescriptor and
                                  ;; use setfontdescriptor to install it.

                                 ;; OBSOLETEd by the CHARSETINFO code (OR (ffetch FONTIMAGEWIDTHS of FONTX)
                                 ;; (freplace FONTIMAGEWIDTHS of FONTX with (ffetch \SFWidths of FONTX)))

                                  ;; the widths fields in the fontdescriptor are obsolete, and shoudln't be updated here.

                                  ;; We should probably force all device implementations to obey these conventions, then
                                  ;; remove these generic updates

                                  (replace (FONTDESCRIPTOR FONTAVGCHARWIDTH) of FONTX
                                     with (\AVGCHARWIDTH FONTX))
                                  (SETFONTDESCRIPTOR FAMILY SIZE FONTFACE ROTATION DEV FONTX))
                                 (T (GO NOTFOUND]
                              ((NEQ DEV (SETQ DEV (U-CASE DEV)))

                             ;; We didn't recognize the device, so check to see whether coercion to U-CASE IL changes anything.
                             ;; Could be slow, but we're heading for an error.

                               (GO NEWDEV))
                              (T (GO NOTFOUND]
            NOTFOUND
                (COND
                   (NOERRORFLG (RETURN NIL))
                   (T (ERROR "FONT NOT FOUND (coerced to)" (LIST FAMILY SIZE FONTFACE ROTATION DEV
                                                                 ))
                      (GO RETRY])
```

(\FONT.SYMBOLMEMB

```
[LAMBDA (USERINPUT LIST)                                    ; Edited  7-Feb-89 15:47 by jds
  (for X on LIST when (\FONT.COMPARESYMBOL USERINPUT (CAR X)) do (RETURN X])
```

(\FONT.SYMBOLASSOC

```
[LAMBDA (USERINPUT LIST)                                    ; Edited 28-Jul-88 16:56 by rmk:
                                                            ; Edited 28-Jul-88 15:15 by rmk:
                                                            ; Edited 28-Jul-88 15:03 by rmk:
                                                            ; Edited 28-Jul-88 14:44 by rmk:
                                                            ; Edited 28-Jul-88 14:16 by rmk:
  (for X FIRSTC (NC _ (NCHARS USERINPUT)) in LIST first (SETQ FIRSTC (CHCON1 USERINPUT))
                                                        [if (AND (IGEQ FIRSTC (CHARCODE a))
                                                                 (ILEQ FIRSTC (CHARCODE z)))
                                                            then (SETQ FIRSTC (IDIFFERENCE FIRSTC
                                                                                           (IDIFFERENCE (CHARCODE a)
                                                                                                        (CHARCODE A]
     when (AND (EQ NC (NCHARS (CAR X)))
               (EQ FIRSTC (CHCON1 (CAR X)))
               (\FONT.COMPARESYMBOL USERINPUT (CAR X)
                      NC FIRSTC))
     do (RETURN X])
```

(\FONT.COMPARESYMBOL

```
[LAMBDA (USERINPUT KEY INPUTNC INPUTFIRSTC)                 ; Edited 24-May-93 16:45 by sybalsky:mv:envos
  ;; An open coded case- and package-insensitive comparison of atom pnames, assuming that KEY is already upper-case but USERINPUT may not
  ;; be.  Maybe there is a simple function that does this.

  ;; INPUTNC and INPUTFIRSTC can be passed in if they are common to lots of calls

  (COND
     ((AND (LITATOM USERINPUT)
           (EQ [CL:AREF *PACKAGE-FROM-INDEX* (fetch (PNAMECELL PACKAGEINDEX) of (PROGN (\PNAMECELL USERINPUT]
               *INTERLISP-PACKAGE*))

      ;; If the user's symbol is in the IL package (which is where all the KEYs are), we can use EQ, which is MUCH faster.

      (OR (EQ USERINPUT KEY)
          (EQ (U-CASE USERINPUT)
              KEY)))
     (T ;; Otherwise, we do the comparison character by character.
        (AND (EQ (OR INPUTNC (NCHARS USERINPUT))
                 (NCHARS KEY))
             [COND
```

```
                (INPUTFIRSTC (EQ INPUTFIRSTC (CHCON1 KEY)))
                ((EQ (SETQ INPUTFIRSTC (CHCON1 USERINPUT))
                     (CHCON1 KEY)))
                ((AND (IGEQ INPUTFIRSTC (CHARCODE a))
                      (ILEQ INPUTFIRSTC (CHARCODE z)))
                 (EQ (IDIFFERENCE INPUTFIRSTC (IDIFFERENCE (CHARCODE a)
                                                           (CHARCODE A)))
                     (CHCON1 KEY]
           (for CHAR1 inatom USERINPUT as CHAR2 inatom KEY
              always (OR (EQ CHAR1 CHAR2)
                         (AND (IGEQ CHAR1 (CHARCODE a))
                              (ILEQ CHAR1 (CHARCODE z))
                              (EQ CHAR2 (IPLUS CHAR1 (CONSTANT (IDIFFERENCE (CHARCODE A)
                                                                            (CHARCODE a])

)

;; Property extraction:

(DEFINEQ

(FONTASCENT
  [LAMBDA (FONTSPEC)                                                    (* lmm "19-NOV-82 00:23")
    (ffetch \SFAscent of (\GETFONTDESC FONTSPEC])

(FONTDESCENT
  [LAMBDA (FONTSPEC)                                                    (* lmm "19-NOV-82 00:24")
                                                                        ; See comment in FONTASCENT
    (ffetch \SFDescent of (\GETFONTDESC FONTSPEC])

(FONTHEIGHT
  [LAMBDA (FONTSPEC)                                                    (* kbr%: " 9-Jan-86 18:29")
    (fetch (FONTDESCRIPTOR \SFHeight) of (\GETFONTDESC FONTSPEC])

(FONTPROP
  [LAMBDA (FONT PROP)                                                   (* kbr%: "13-May-85 22:36")
    (SETQ FONT (\GETFONTDESC FONT))
    (SELECTQ PROP
         (HEIGHT (ffetch \SFHeight of FONT))
         (ASCENT (ffetch \SFAscent of FONT))
         (DESCENT (ffetch \SFDescent of FONT))
         (FAMILY (ffetch FONTFAMILY of FONT))
         (SIZE (ffetch FONTSIZE of FONT))
         (FACE (COPY (ffetch FONTFACE of FONT)))
         (WEIGHT (ffetch WEIGHT of (ffetch FONTFACE of FONT)))
         (SLOPE (ffetch SLOPE of (ffetch FONTFACE of FONT)))
         (EXPANSION (ffetch EXPANSION of (ffetch FONTFACE of FONT)))
         (FORECOLOR (ffetch FORECOLOR of (ffetch FONTFACE of FONT)))
         (BACKCOLOR (ffetch BACKCOLOR of (ffetch FONTFACE of FONT)))
         (ROTATION (ffetch ROTATION of FONT))
         (DEVICE (ffetch FONTDEVICE of FONT))
         (SPEC (LIST (ffetch FONTFAMILY of FONT)
                     (ffetch FONTSIZE of FONT)
                     (COPY (ffetch FONTFACE of FONT))
                     (ffetch ROTATION of FONT)
                     (ffetch FONTDEVICE of FONT)))
         (DEVICESPEC                                                    ; DEVICE fields are for communicating coercions to the particular
                                                                        ; printing device
                     [COND
                        ((ffetch FONTDEVICESPEC of FONT)
                         (COPY (ffetch FONTDEVICESPEC of FONT)))
                        (T (FONTPROP FONT 'SPEC])
         (DEVICEFACE [COPY (COND
                             ((ffetch FONTDEVICESPEC of FONT)
                              (CADDR (ffetch FONTDEVICESPEC of FONT)))
                             (T (ffetch FONTFACE of FONT])
         (DEVICESLOPE [fetch SLOPE of (COND
                                        ((ffetch FONTDEVICESPEC of FONT)
                                         (CADDR (ffetch FONTDEVICESPEC of FONT)))
                                        (T (ffetch FONTFACE of FONT])
         (DEVICEWEIGHT [fetch WEIGHT of (COND
                                          ((ffetch FONTDEVICESPEC of FONT)
                                           (CADDR (ffetch FONTDEVICESPEC of FONT)))
                                          (T (ffetch FONTFACE of FONT])
         (DEVICEEXPANSION
              [fetch EXPANSION of (COND
                                    ((ffetch FONTDEVICESPEC of FONT)
                                     (CADDR (ffetch FONTDEVICESPEC of FONT)))
                                    (T (ffetch FONTFACE of FONT])
         (DEVICESIZE (COND
                       ((ffetch FONTDEVICESPEC of FONT)
                        (CADR (ffetch FONTDEVICESPEC of FONT)))
                       (T (ffetch FONTSIZE of FONT))))
         (DEVICEFAMILY (COND
```

```
                        ((ffetch FONTDEVICESPEC of FONT)
                         (CAR (ffetch FONTDEVICESPEC of FONT)))
                        (T (ffetch FONTFAMILY of FONT))))
      (SCALE (ffetch FONTSCALE of FONT))
      (\ILLEGAL.ARG PROP])
```

(\AVGCHARWIDTH

```
  [LAMBDA (FONT)                                                    (* rmk%: "27-Nov-84 18:40")
    ;; Returns the average width of a character, to be used in units-to-characters approximations, as in fixing the linelength
    (PROG ((W (CHARWIDTH (CHARCODE A)
                         FONT)))
          (RETURN (COND
                     ((NEQ 0 W)
                      W)
                     ([NEQ 0 (SETQ W (FIXR (FTIMES 0.6 (FONTPROP FONT 'HEIGHT]
                      W)
                     (T 1])

)

;; Bitmap editing/manipulation:

(DEFINEQ

(GETCHARBITMAP
  [LAMBDA (CHARCODE FONT)                                           ; Edited 26-Apr-89 21:49 by atm
                                                                    ; returns a bitmap of the character CHARCODE from the font
                                                                    ; descriptor FONTDESC.
    (COND
       ((OR (CHARCODEP CHARCODE)
            (EQ CHARCODE 256))                                      ; bitmap for char 256 is what gets printed if char not found

        )
       ((OR (STRINGP CHARCODE)
            (LITATOM CHARCODE))                                     ; For strings & litatoms, take the first character
        (SETQ CHARCODE (CHCON1 CHARCODE)))
       ((TYPEP CHARCODE 'CL:CHARACTER)                              ; For common-lisp CHARACTERs, convert it to the char code
                                                                    ; first.
        (SETQ CHARCODE (CL:CHAR-INT CHARCODE)))
       (T (\ILLEGAL.ARG CHARCODE)))
    (PROG (CBM (FONTDESC (\GETFONTDESC FONT))
               CSINFO CWDTH CHGHT)
     ;; fetch the csinfo for the character set of this character.  Bitmaps and widths must be fetched from it
          (SETQ CSINFO (\GETCHARSETINFO (\CHARSET CHARCODE)
                                FONTDESC))
     ;; (\\fgetwidth (|fetch| (charsetinfo widths) |of| csinfo) (\\char8code charcode))
          [SETQ CBM (BITMAPCREATE [SETQ CWDTH (if (fetch (CHARSETINFO IMAGEWIDTHS) of CSINFO)
                                                  then (\FGETIMAGEWIDTH (fetch (CHARSETINFO IMAGEWIDTHS)
                                                                               of CSINFO)
                                                                        (\CHAR8CODE CHARCODE))
                                                else (\FGETWIDTH (fetch (CHARSETINFO WIDTHS) of CSINFO)
                                                                 (\CHAR8CODE CHARCODE]
                                  (SETQ CHGHT (FONTPROP FONTDESC 'HEIGHT))
                                  (fetch (BITMAP BITMAPBITSPERPIXEL) of (fetch (CHARSETINFO CHARSETBITMAP) of CSINFO]
          (BITBLT (fetch (CHARSETINFO CHARSETBITMAP) of CSINFO)
                  (\FGETOFFSET (fetch (CHARSETINFO OFFSETS) of CSINFO)
                               (\CHAR8CODE CHARCODE))
                  0 CBM 0 0 CWDTH CHGHT)
          (RETURN CBM])
```

(PUTCHARBITMAP

```
  [LAMBDA (CHARCODE FONT NEWCHARBITMAP NEWCHARDESCENT)              ; Edited 27-Apr-89 11:19 by atm
    ;; stores the bitmap NEWCHARBITMAP as the character CHARCODE from the font descriptor FONTDESC.  If NEWCHARDESCENT is specified,
    ;; it is the descent of the new bitmap, and things may be moved to accomodate it.
    (OR (TYPENAMEP NEWCHARBITMAP 'BITMAP)
        (\ILLEGAL.ARG NEWCHARBITMAP))
    (COND
       ((CHARCODEP CHARCODE))
       ((OR (STRINGP CHARCODE)
            (LITATOM CHARCODE))
        (SETQ CHARCODE (CHCON1 CHARCODE)))
       (T (\ILLEGAL.ARG CHARCODE)))
    (PROG* ((FONTDESC (\GETFONTDESC FONT))
            (CSINFO (\GETCHARSETINFO (\CHARSET CHARCODE)
                             FONTDESC))
            (CDESCENT (fetch (CHARSETINFO CHARSETDESCENT) of CSINFO))
            (CASCENT (fetch (CHARSETINFO CHARSETASCENT) of CSINFO))
            (CHEIGHT (IPLUS CDESCENT CASCENT))
            (OFFSETS (fetch (CHARSETINFO OFFSETS) of CSINFO))
            (WIDTHS (fetch (CHARSETINFO WIDTHS) of CSINFO))
            (IMWIDTHS (fetch (CHARSETINFO IMAGEWIDTHS) of CSINFO))
```

```
        (CIMWIDTH (if IMWIDTHS
                      then (\FGETIMAGEWIDTH IMWIDTHS (\CHAR8CODE CHARCODE))
                    else NIL))
        (CWIDTH (OR CIMWIDTH (CHARWIDTH CHARCODE FONTDESC)))
        (FONTBITMAP (fetch (CHARSETINFO CHARSETBITMAP) of CSINFO))
        (OFWIDTH (fetch (BITMAP BITMAPWIDTH) of FONTBITMAP))
        TEMPBITMAP BWIDTH DW BHEIGHT BASCENT BDESCENT NDESCENT NASCENT NHEIGHT CHAROFFSET
        (BITSPERPIXEL (fetch (BITMAP BITMAPBITSPERPIXEL) of FONTBITMAP)))
```

;; fetch the ascents and descents of the bitmap and the new maximums.

```
    (SETQ BWIDTH (fetch (BITMAP BITMAPWIDTH) of NEWCHARBITMAP))
    (SETQ BHEIGHT (fetch (BITMAP BITMAPHEIGHT) of NEWCHARBITMAP))
    (SETQ BDESCENT (OR NEWCHARDESCENT CDESCENT))
    (SETQ BASCENT (IDIFFERENCE BHEIGHT BDESCENT))
    (SETQ NDESCENT (IMAX BDESCENT CDESCENT))
    (SETQ NASCENT (IMAX BASCENT CASCENT))
    (SETQ NHEIGHT (IPLUS NDESCENT NASCENT))
    (SETQ CHAROFFSET (\FGETOFFSET OFFSETS (\CHAR8CODE CHARCODE)))
```

;; set up a new target bitmap if any of the parameters have changed.

```
    (COND
       ((EQ CHAROFFSET (\FGETOFFSET OFFSETS \MAXTHINCHAR))
```

;; changing the bitmap for a character which formerly pointed at the slug character. Allocate a new bitmap character bitmap for this.

```
        (SETQ TEMPBITMAP (BITMAPCREATE (IPLUS OFWIDTH BWIDTH)
                                NHEIGHT BITSPERPIXEL))
        (BITBLT FONTBITMAP 0 0 TEMPBITMAP 0 (IMAX 0 (IDIFFERENCE NDESCENT CDESCENT))
               OFWIDTH CHEIGHT)                          ; copy the old characters over.
        (SETQ CHAROFFSET OFWIDTH))
       ((NEQ CWIDTH BWIDTH)
```

;; The bitmaps differ in width; create a new bitmap with things at the right places, then update widths and offsets.

```
        (SETQ DW (IDIFFERENCE BWIDTH CWIDTH))            ; Difference in character widths
        (SETQ TEMPBITMAP (BITMAPCREATE (IPLUS OFWIDTH DW)
                                NHEIGHT BITSPERPIXEL))   ; this may also be a taller bitmap if NHEIGHT is larger than
                                                         ; CHEIGHT.
        (BITBLT FONTBITMAP 0 0 TEMPBITMAP 0 (IMAX 0 (IDIFFERENCE NDESCENT CDESCENT))
               CHAROFFSET CHEIGHT)                       ; Copy that portion to the left of the character.
        (BITBLT FONTBITMAP (IPLUS CHAROFFSET CWIDTH)
               0 TEMPBITMAP (IPLUS CHAROFFSET BWIDTH)
               (IMAX 0 (IDIFFERENCE NDESCENT CDESCENT))
               (ADD1 (IDIFFERENCE OFWIDTH (IPLUS CHAROFFSET CWIDTH)))
               CHEIGHT)                                  ; Copy that portion to the right of the new character.

        )
       ((OR (IGREATERP BASCENT CASCENT)
            (IGREATERP BDESCENT CDESCENT))
```

;; The new character is TALLER than the existing bitmap. Make a larger bitmap.

```
        (SETQ TEMPBITMAP (BITMAPCREATE OFWIDTH NHEIGHT BITSPERPIXEL))
        (BITBLT FONTBITMAP 0 0 TEMPBITMAP 0 (IMAX 0 (IDIFFERENCE NDESCENT CDESCENT))
               OFWIDTH CHEIGHT)
```

;; Copy the existing bitmap into it, adjusting for a larger descent in the new character (if there is one)

```
        ))
```

;; copy the new bitmap in and update parameters.

```
    (BITBLT NEWCHARBITMAP 0 0 (OR TEMPBITMAP FONTBITMAP)
           CHAROFFSET
           (IMAX 0 (IDIFFERENCE NDESCENT BDESCENT))
           BWIDTH BHEIGHT)
    [COND
       (TEMPBITMAP (UNINTERRUPTABLY
```

;; update the parameters for this character set.

```
                        (\FSETWIDTH WIDTHS (\CHAR8CODE CHARCODE)
                               BWIDTH)                   ; The new character's correct width
                                                         ; Make sure that we update imagewidths also
                        (if IMWIDTHS
                            then (\FSETIMAGEWIDTH IMWIDTHS (\CHAR8CODE CHARCODE)
                                         BWIDTH))
                        (\FSETOFFSET OFFSETS (\CHAR8CODE CHARCODE)
                              CHAROFFSET)
                        [COND
                           (DW (for I from 0 to \MAXCHAR
                                  do                     ; Run thru the offsets of later characters, adjusting them for the
                                                         ; changed width of this character
                                      (if (IGREATERP (\FGETOFFSET OFFSETS I)
                                                     CHAROFFSET)
                                          then (\FSETOFFSET OFFSETS I (IPLUS DW (\FGETOFFSET OFFSETS I]
                        (replace (CHARSETINFO CHARSETBITMAP) of CSINFO with TEMPBITMAP)
                        (replace (CHARSETINFO CHARSETDESCENT) of CSINFO with NDESCENT)
                        (replace (CHARSETINFO CHARSETASCENT) of CSINFO with NASCENT)
```

;; update the properties for the font as a whole.

```
                        [SETQ NASCENT (IMAX NASCENT (FONTPROP FONTDESC 'ASCENT]
                        [SETQ NDESCENT (IMAX NDESCENT (FONTPROP FONTDESC 'DESCENT]
```

```
                              (replace (FONTDESCRIPTOR \SFAscent) of FONTDESC with NASCENT)
                              (replace (FONTDESCRIPTOR \SFDescent) of FONTDESC with NDESCENT)
                              (replace (FONTDESCRIPTOR \SFHeight) of FONTDESC with (IPLUS NDESCENT NASCENT)))]
            (RETURN NEWCHARBITMAP])
```

(MOVECHARBITMAP

```
  [LAMBDA (SRCECODE SRCEFONT DESTCODE DESTFONT CLIP)            ; Edited 14-Dec-86 18:04 by Shih
```

;;; moves a character from one font to another, clipping if necessary.

```
    (PROG ((SRCEDESC (\GETFONTDESC SRCEFONT))
           (DESTDESC (\GETFONTDESC DESTFONT))
           SRCEASCENT SRCEDESCENT DESTASCENT DESTDESCENT CHARBITMAP TEMPBITMAP NEWASCENT NEWDESCENT)
          (SETQ CHARBITMAP (GETCHARBITMAP SRCECODE SRCEFONT))
          (SETQ SRCEASCENT (FONTPROP SRCEDESC 'ASCENT))
          (SETQ DESTASCENT (FONTPROP DESTDESC 'ASCENT))
          (SETQ SRCEDESCENT (FONTPROP SRCEDESC 'DESCENT))
          (SETQ DESTDESCENT (FONTPROP DESTDESC 'DESCENT))
          [SETQ NEWASCENT (COND
                             (CLIP DESTASCENT)
                             (T (IMAX SRCEASCENT DESTASCENT]
          [SETQ NEWDESCENT (COND
                              (CLIP DESTDESCENT)
                              (T (IMAX SRCEDESCENT DESTDESCENT]
          [COND
             ((OR (NEQ SRCEASCENT NEWASCENT)
                  (NEQ SRCEDESCENT NEWDESCENT))
              (SETQ TEMPBITMAP (BITMAPCREATE (BITMAPWIDTH CHARBITMAP)
                                             (IPLUS NEWASCENT NEWDESCENT)))
              (BITBLT CHARBITMAP 0 (IMAX 0 (IDIFFERENCE SRCEDESCENT NEWDESCENT))
                     TEMPBITMAP 0 (IMAX 0 (IDIFFERENCE NEWDESCENT SRCEDESCENT))
                     (BITMAPWIDTH CHARBITMAP)
                     (IMIN (IPLUS SRCEASCENT SRCEDESCENT)
                           (IPLUS NEWASCENT NEWDESCENT]
          (PUTCHARBITMAP DESTCODE DESTFONT (OR TEMPBITMAP CHARBITMAP)
                 NEWDESCENT])

)

(DEFINEQ
```

(FONTCOPY

```
  [LAMBDA FONTSPECS                                             ; Edited 10-Nov-87 17:12 by FS
                                                                ; makes a copy of a font changing the specified fields.
    (PROG (NOERROR ERROR FAMILY FACE SIZE ROTATION DEVICE OLDFONT)
```

;; Set NOERROR if we find it as a prop, but set ERROR if we find a PROP which is illegal. Then just return NIL if NOERROR and ERROR, ;; otherwise, call FONTCREATE.

```
          [SETQ OLDFONT (\GETFONTDESC (ARG FONTSPECS 1)
                                (AND (type? FONTCLASS (ARG FONTSPECS 1))
                                     (COND
                                        ((AND (EQ FONTSPECS 2)
                                              (LISTP (ARG FONTSPECS 2)))
                                         (LISTGET (ARG FONTSPECS 2)
                                                  'DEVICE))
                                        (T (for I from 2 by 2 to FONTSPECS
                                              do (COND
                                                    ((AND (NEQ I FONTSPECS)
                                                          (EQ (ARG FONTSPECS I)
                                                              'DEVICE))
                                                     (RETURN (ARG FONTSPECS (ADD1 I]
          (SETQ FAMILY (fetch (FONTDESCRIPTOR FONTFAMILY) of OLDFONT))
          (SETQ SIZE (fetch (FONTDESCRIPTOR FONTSIZE) of OLDFONT))
          (SETQ FACE (fetch (FONTDESCRIPTOR FONTFACE) of OLDFONT))
          (SETQ ROTATION (fetch (FONTDESCRIPTOR ROTATION) of OLDFONT))
          (SETQ DEVICE (fetch (FONTDESCRIPTOR FONTDEVICE) of OLDFONT))
          [for I VAL from 2 by 2 to FONTSPECS
             do [SETQ VAL (COND
                             ((NOT (EQ I FONTSPECS))
                              (ARG FONTSPECS (ADD1 I]
                (SELECTQ (ARG FONTSPECS I)
                     (FAMILY (SETQ FAMILY VAL))
                     (SIZE (SETQ SIZE VAL))
                     (FACE (SETQ FACE (\FONTFACE VAL)))
                     (WEIGHT (SETQ FACE (create FONTFACE using FACE WEIGHT _ VAL)))
                     (SLOPE (SETQ FACE (create FONTFACE using FACE SLOPE _ VAL)))
                     (EXPANSION (SETQ FACE (create FONTFACE using FACE EXPANSION _ VAL)))
                     (BACKCOLOR (SETQ FACE (create FONTFACE using FACE BACKCOLOR _ VAL)))
                     (FORECOLOR (SETQ FACE (create FONTFACE using FACE FORECOLOR _ VAL)))
                     (ROTATION (SETQ ROTATION VAL))
                     (DEVICE (SETQ DEVICE VAL))
                     (NOERROR (SETQ NOERROR VAL))
                     (COND
                        [(AND (EQ I 2)
                              (EQ FONTSPECS 2)
                              (LISTP (ARG FONTSPECS 2)))
```

```
                        (for J on (ARG FONTSPECS 2) by (CDDR J)
                           do (SETQ VAL (CADR J))
                              (SELECTQ (CAR J)
                                   (FAMILY (SETQ FAMILY VAL))
                                   (SIZE (SETQ SIZE VAL))
                                   (FACE (SETQ FACE (\FONTFACE VAL)))
                                   (WEIGHT (SETQ FACE (create FONTFACE using FACE WEIGHT _ VAL)))
                                   (SLOPE (SETQ FACE (create FONTFACE using FACE SLOPE _ VAL)))
                                   (EXPANSION (SETQ FACE (create FONTFACE using FACE EXPANSION _ VAL)))
                                   (BACKCOLOR (SETQ FACE (create FONTFACE using FACE BACKCOLOR _ VAL)))
                                   (FORECOLOR (SETQ FACE (create FONTFACE using FACE FORECOLOR _ VAL)))
                                   (ROTATION (SETQ ROTATION VAL))
                                   (DEVICE (SETQ DEVICE VAL))
                                   (NOERROR (SETQ NOERROR VAL))
                                   (COND
                                      (NOERROR
                                             ;; Fell through the SELECTQ, so an illegal PROP.  But, if NOERROR, just note the error,
                                             ;; otherwise
                                             (SETQ ERROR T))
                                      (T (\ILLEGAL.ARG (CAR J]
                     (T (if NOERROR
                            then (SETQ ERROR T)
                          else (\ILLEGAL.ARG (ARG FONTSPECS I]
          (RETURN (if (AND NOERROR ERROR)
                      then NIL
                    else (FONTCREATE FAMILY SIZE FACE ROTATION DEVICE NOERROR])
```

(FONTSAVAILABLE

```
  [LAMBDA (FAMILY SIZE FACE ROTATION DEVICE CHECKFILESTOO?)       (* rrb " 7-Nov-84 15:41")
```

;;; returns a list of the fonts fitting a description that are available. FAMILY SIZE FACE or ROTATION can be * which means get them all. if
;;; LOADEDONLYFLG is non-NIL, only fonts in core will be considered.

```
    (DECLARE (GLOBALVARS IMAGESTREAMTYPES))
    (PROG (FONTX DEV)
          [SETQ DEV (COND
                       ((type? STREAM DEVICE)
                        (COND
                           ((LISTP (SETQ DEV (IMAGESTREAMTYPE DEVICE)))
                            (CAR DEV))
                           (T DEV)))
                       (DEVICE)
                       (T 'DISPLAY]
          (RETURN (COND
                     ((LISTP FAMILY)
                      (COND
                         ((EQ (CAR FAMILY)
                              'FONT)
                          (SETQ FONTX (CDR FAMILY)))
                         (T (SETQ FONTX FAMILY)))
                      (FONTSAVAILABLE (CAR FONTX)
                             (OR (CADR FONTX)
                                 SIZE)
                             (OR (CADDR FONTX)
                                 FACE)
                             (OR (CADDDR FONTX)
                                 ROTATION)
                             DEV CHECKFILESTOO?))
                     ([SETQ FONTX (COND
                                     ((type? FONTDESCRIPTOR FAMILY)
                                      FAMILY)
                                     ((NULL FAMILY)
                                      (DEFAULTFONT DEV))
                                     ((type? FONTCLASS FAMILY)
                                             ;; We know that this won't attempt a cyclic fontcreate in \COERCEFONTDESC, because we are
                                             ;; passing a known class.  Unless NOERROFLG, an error will be caused on the actual device font if it
                                             ;; can't be found.                    ; I don't know what to do in this case- rrb.
                                      (\COERCEFONTDESC FAMILY DEV T))
                                     ((OR (IMAGESTREAMP FAMILY)
                                          (type? WINDOW FAMILY))
                                      (DSPFONT NIL FAMILY]       ; FAMILY was a spec for a font descriptor, use it and extend it by
                                                                 ; the other args.
                      (FONTSAVAILABLE (FONTPROP FONTX 'FAMILY)
                             (OR SIZE (FONTPROP FONTX 'SIZE))
                             (OR FACE (FONTPROP FONTX 'FACE))
                             (OR ROTATION (FONTPROP FONTX 'ROTATION))
                             (OR DEVICE (FONTPROP FONTX 'DEVICE))
                             CHECKFILESTOO?))
                     (T (PROG ((FONTFACE FACE))
                              RETRY
                              (OR (LITATOM FAMILY)
                                  (LISPERROR "ARG NOT LITATOM" FAMILY T))
                              (OR (AND (FIXP SIZE)
                                       (IGREATERP SIZE 0))
                                  (EQ SIZE '*)
```

```
                    (\ILLEGAL.ARG SIZE))
                [OR (EQ FONTFACE '*)
                    (SETQ FONTFACE (OR (\FONTFACE FACE T)
                                       (RETURN NIL]
                (OR (U-CASEP FAMILY)
                    (SETQ FAMILY (U-CASE FAMILY)))
                (COND
                   ((NULL ROTATION)
                    (SETQ ROTATION 0))
                   ((AND (FIXP ROTATION)
                         (IGEQ ROTATION 0)))
                   ((EQ ROTATION '*))
                   (T (\ILLEGAL.ARG ROTATION)))
                (RETURN (UNION (\LOOKUPFONTSINCORE FAMILY SIZE FONTFACE ROTATION DEV)
                               (COND
                                  ((NOT CHECKFILESTOO?)
                                   NIL)
                                  [(EQ DEV '*)     ; map thru all the devices.
                                   (for EXTANTDEV in IMAGESTREAMTYPES
                                      join (APPLY* (OR (CADR (ASSOC 'FONTSAVAILABLE (CDR EXTANTDEV)
                                                                                    ))
                                                       (FUNCTION NILL))
                                                   FAMILY SIZE FONTFACE ROTATION (CAR EXTANTDEV]
                                  (T               ; apply the device font lookup function.
                                     (APPLY* (OR [CADR (ASSOC 'FONTSAVAILABLE
                                                              (CDR (ASSOC DEV IMAGESTREAMTYPES]
                                                 (FUNCTION NILL))
                                           FAMILY SIZE FONTFACE ROTATION DEV])
```

(FONTFILEFORMAT

```
[LAMBDA (STRM LEAVEOPEN)                                    (* rmk%: "11-Sep-84 17:16")
                                                            ; Returns the font format of STRM

   [OR (OPENP STRM 'INPUT)
       (SETQ STRM (OPENSTREAM STRM 'INPUT 'OLD]
   (PROG1 (SELECTC (\WIN STRM)
                ((LIST (LLSH 1 15)
                       (LOGOR (LLSH 1 15)
                              (LLSH 1 13)))
                                        ;; If high bit of type is on, then must be strike.  If 2nd bit is on, must be strike-index, and we
                                        ;; punt.  We don't care about the 3rd bit
                    ;; first word has high bits (onebit index fixed).  Onebit means 'new-style font' , index is 0 for simple strike, 1 for index, and
                    ;; fixed is if all chars have max width.  Lisp doesn't care about 'fixed'
                    'STRIKE)
                ((LOGOR (LLSH 16 8)
                        12)  ;; This is the length of a standard index header.  Other files could also have this value, but it's a pretty good
                             ;; discriminator
                    ;; Skip to byte 25;  do it with BINS so works for non-randaccessp devices.  This skips the standard name header, then look for
                    ;; type 3 in the following header
                    (FRPTQ 22 (\BIN STRM))                  ; (SETFILEPTR STRM 25)
                    (AND (EQ 3 (LRSH (\BIN STRM)
                                     4))
                         'AC))
                NIL)
       (OR LEAVEOPEN (CLOSEF STRM)))])
```

(FONTP

```
[LAMBDA (X)                                                 (* rmk%: "13-Sep-84 09:04")
                                                            ; is X a FONTDESCRIPTOR?

   (COND
      ((OR (type? FONTDESCRIPTOR X)
           (type? FONTCLASS X))
       X])
```

(FONTUNPARSE

```
[LAMBDA (FONT)                                              (* kbr%: "25-Feb-86 19:40")
```

;; Produces a minimal specification of the font or fontclass specification, for dumping by Tedit, imageobjects.

```
   (PROG (FACE SPEC)
         (SETQ SPEC (COND
                       ((type? FONTDESCRIPTOR FONT)
                        (FONTPROP FONT 'SPEC))
                       [(type? FONTCLASS FONT)
                        (RETURN (CONS 'CLASS (FONTCLASSUNPARSE FONT]
                       (T  ;; Could be a non-instantiated specification in a fontclass, just use it as the spec without creating the font.
                          FONT)))
         (OR SPEC (RETURN))
         (SETQ FACE (CADDR SPEC))                           ; FACE and rotation can be NIL for a non-fontdescriptor fontclass
                                                            ; component
         [SETQ FACE (COND
                       ([OR (NULL FACE)
```

```
                        (EQUAL FACE '(MEDIUM REGULAR REGULAR]
                    NIL)
                   ((LITATOM FACE)
                    FACE)
                   [(LISTP FACE)
                    (PACK (LIST* (NTHCHAR (fetch (FONTFACE WEIGHT) of FACE)
                                          1)
                                 (NTHCHAR (fetch (FONTFACE SLOPE) of FACE)
                                          1)
                                 (NTHCHAR (fetch (FONTFACE EXPANSION) of FACE)
                                          1)
                                 (COND
                                    ((fetch (FONTFACE COLOR) of FACE)
                                     (LIST "-" (fetch (FONTFACE BACKCOLOR) of FACE)
                                           "-"
                                           (fetch (FONTFACE FORECOLOR) of FACE]
                   (T (SHOULDNT]                                  ; Don't return device, or any trailing defaults
        (RETURN (CONS (CAR SPEC)
                      (CONS (CADR SPEC)
                            (COND
                               ([AND (CADDDR SPEC)
                                     (NOT (EQ 0 (CADDDR SPEC]
                                (LIST (OR FACE 'MRR)
                                      (CADDDR SPEC)))
                               (FACE (CONS FACE])
```

(**SETFONTDESCRIPTOR**

```
[LAMBDA (FAMILY SIZE FACE ROTATION DEVICE FONT)                   ; Edited  1-Aug-88 16:16 by rmk:
                                                                  ; Edited  5-Mar-87 19:28 by FS
```

;; saves a font descriptor under a family/size/face/rotation/device key so that it will be retreived by FONTCREATE. This is a user entry.

```
(DECLARE (GLOBALVARS \FONTSINCORE))
(SETQ DEVICE (\DEVICESYMBOL DEVICE))                              ; Unpackageify
(AND FONT (SETQ FONT (\COERCEFONTDESC FONT DEVICE)))              ; NIL is used to clobber existing font so that next use will reread
                                                                  ; it.
(SETQ FAMILY (\FONTSYMBOL FAMILY))                                ; Unpackageify
(SETQ FACE (\FONTFACE FACE NIL DEVICE))
(OR ROTATION (SETQ ROTATION 0))
(OR (AND (FIXP SIZE)
         (IGEQ SIZE 0))
    (\ILLEGAL.ARG SIZE))
(PROG [(X (OR (FASSOC FAMILY \FONTSINCORE)
              (CAR (push \FONTSINCORE (LIST FAMILY]
      [SETQ X (OR (FASSOC SIZE (CDR X))
                  (CAR (push (CDR X)
                             (LIST SIZE]
      [SETQ X (OR (SASSOC FACE (CDR X))
                  (CAR (push (CDR X)
                             (LIST FACE]                          ; SASSOC cause FACE is listp
      [SETQ X (OR (FASSOC ROTATION (CDR X))
                  (CAR (push (CDR X)
                             (LIST ROTATION]
      [SETQ X (OR (FASSOC DEVICE (CDR X))
                  (CAR (push (CDR X)
                             (LIST DEVICE]
      (RPLACD X FONT)
      (RETURN FONT])
```

(**CHARCODEP**

```
[LAMBDA (CHCODE)                                                  (* gbn "22-Jul-85 16:35")
                                                                  ; is CHCODE a legal character code?
  (AND (SMALLP CHCODE)
       (IGEQ CHCODE 0)
       (ILEQ CHCODE \MAXNSCHAR])
```

(**EDITCHAR**

```
[LAMBDA (CHARCODE FONT)                                           (* rrb "24-MAR-82 12:22")
                                                                  ; calls the bitmap editor on a character of a font
  (PROG ((FONTDESC (\GETFONTDESC FONT)))
        (RETURN (PUTCHARBITMAP CHARCODE FONTDESC (EDITBM (GETCHARBITMAP CHARCODE FONTDESC])
```

(**\STREAMCHARWIDTH**

```
[LAMBDA (CHARCODE STREAM TTBL)                                    (* JonL " 8-NOV-83 03:31")
```

;; Returns the width that the printed representation of CHARCODE would occupy if printed on STREAM, allowing for the various escape
;; sequences. Used by \ECHOCHAR

```
(SETQ CHARCODE (LOGAND CHARCODE \CHARMASK))
((LAMBDA (WIDTHSVECTOR)
```

;; Note in following that if the DDWIDTHSCACHE exists and has a 0 entry for some character, that may someday mean that the character's
;; glyph simply isn't loaded; e.g., it may want #^A

```
   (SETQ WIDTHSVECTOR (OR (AND (DISPLAYSTREAMP STREAM)
                               (SETQ WIDTHSVECTOR (ffetch IMAGEDATA of STREAM))
```

```
                                    (ffetch DDWIDTHSCACHE of WIDTHSVECTOR))
                                \UNITWIDTHSVECTOR))
        (SELECTC (fetch CCECHO of (\SYNCODE (fetch (TERMTABLEP TERMSA) of (OR (TERMTABLEP TTBL)
                                                                             \PRIMTERMTABLE))
                                       CHARCODE))
            (INDICATE.CCE ([LAMBDA (CC)
                              (IPLUS (if (IGEQ CHARCODE (CHARCODE %#^@))
                                         then                          ; A META charcode -- implies that the 8th bit is non-zero
                                              (SETQ CC (LOADBYTE CHARCODE 0 7))
                                              (\FGETWIDTH WIDTHSVECTOR (CHARCODE %#))
                                       else 0)
                                     (if (ILESSP CC (CHARCODE SPACE))
                                         then                          ; A CONTROL charcode
                                              (add CC (CONSTANT (LLSH 1 6)))
                                              (\FGETWIDTH WIDTHSVECTOR (CHARCODE ^))
                                       else 0)
                                     (\FGETWIDTH WIDTHSVECTOR CC]
                           CHARCODE))
            (SIMULATE.CCE (SELCHARQ CHARCODE
                                ((EOL CR LF BELL)
                                   NIL)
                                (ESCAPE (\FGETWIDTH WIDTHSVECTOR (CHARCODE $)))
                                (TAB (PROG ((SPACEWIDTH (\FGETWIDTH WIDTHSVECTOR (CHARCODE SPACE)))
                                            (NEWXPOSITON (DSPXPOSITION NIL STREAM))
                                            TABWIDTH)
                                           (SETQ TABWIDTH (UNFOLD SPACEWIDTH 8))
                                           [add NEWXPOSITON (SETQ TABWIDTH (IDIFFERENCE TABWIDTH
                                                                                 (IMOD (IDIFFERENCE NEWXPOSITON
                                                                                                    (DSPLEFTMARGIN
                                                                                                     NIL STREAM))
                                                                                       TABWIDTH]
                                           (RETURN (if (IGREATERP NEWXPOSITON (DSPRIGHTMARGIN NIL STREAM))
                                                       then          ; tab was past rightmargin, force cr.
                                                            NIL
                                                     else TABWIDTH))))
                                (\FGETWIDTH WIDTHSVECTOR CHARCODE)))
            (REAL.CCE (SELECTC CHARCODE
                           ((CHARCODE (EOL CR LF))
                               NIL)
                           (ERASECHARCODE
                               NIL)
                           (\FGETWIDTH WIDTHSVECTOR CHARCODE)))
            (IGNORE.CCE 0)
            (SHOULDNT])
```

(\UNITWIDTHSVECTOR

```
[LAMBDA NIL                                                    (* JonL " 7-NOV-83 19:23")
  (SETQ \UNITWIDTHSVECTOR (\ALLOCBLOCK (UNFOLD (IPLUS \MAXCHAR 3)
                                               WORDSPERCELL)))
  (for I from 0 to (IPLUS \MAXCHAR 2) do (\PUTBASE \UNITWIDTHSVECTOR I 1))
  \UNITWIDTHSVECTOR])
```

(\CREATEDISPLAYFONT

```
[LAMBDA (FAMILY SIZE FACE ROTATION DEVICE CHARSET)             (* gbn%: "25-Jan-86 18:02")
  (PROG [(FONTDESC (create FONTDESCRIPTOR
                           FONTDEVICE _ DEVICE
                           FONTFAMILY _ FAMILY
                           FONTSIZE _ SIZE
                           FONTFACE _ FACE
                           \SFAscent _ 0
                           \SFDescent _ 0
                           \SFHeight _ 0
                           ROTATION _ ROTATION
                           FONTDEVICESPEC _ (LIST FAMILY SIZE FACE ROTATION DEVICE]
        (RETURN (COND
                   ((\GETCHARSETINFO CHARSET FONTDESC T)
                    FONTDESC)
                   (T NIL])
```

(\CREATECHARSET.DISPLAY

```
[LAMBDA (FAMILY SIZE FACE ROTATION DEVICE CHARSET FONTDESC NOSLUG?)
                                                               ; Edited 14-Jan-88 23:42 by FS
```

;; Color Stuff removed -FS.

;; Replace Cond below with

;; (PROG (XCSINFO)

;; (SETQ XCSINFO &)

;; (COND ((FMEMB DEVICE \\COLORDISPLAYSTREAMTYPES) (SETQ XCSINFO (\\SFMAKECOLOR XCSINFO (OR (|fetch| (FONTFACE
;; BACKCOLOR) |of| FACE) 0) (OR (|fetch| (FONTFACE FORECOLOR) |of| FACE) (MAXIMUMCOLOR (\\DISPLAYSTREAMTYPEBPP DEVICE)))
;; (\\DISPLAYSTREAMTYPEBPP DEVICE)))))

;; (RETURN XCSINFO)))

;;; tries to build the csinfo required for CHARSET. Does the necessary coercions.

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

```
(DECLARE (GLOBALVARS DISPLAYFONTCOERCIONS MISSINGDISPLAYFONTCOERCIONS))
;; DISPLAYFONTCOERCIONS is a list of font coercions, in the form ((user-font real-font) (user-font real-font) ...). Each user-font is a list of
;; FAMILY, and optionally SIZE and CHARSET, (e.g., (GACHA) or (GACHA 10) or (GACHA 10 143)), and each real-font is a similar list.
(COND
   ((PROG1 (for TRANSL in DISPLAYFONTCOERCIONS bind NEWCSINFO UFONT REALFONT
              when (AND (SETQ UFONT (CAR TRANSL))
                        (EQ FAMILY (CAR UFONT))
                        (OR (NOT (CADR UFONT))
                            (EQ SIZE (CADR UFONT)))
                        (OR (NOT (CADDR UFONT))
                            (EQ CHARSET (CADDR UFONT)))
                        (SETQ REALFONT (CADR TRANSL))
                        (SETQ NEWCSINFO (\CREATECHARSET.DISPLAY (OR (CAR REALFONT)
                                                                    FAMILY)
                                                                (OR (CADR REALFONT)
                                                                    SIZE)
                                                                FACE ROTATION DEVICE (OR (CADDR REALFONT)
                                                                                         CHARSET)
                                                                FONTDESC NOSLUG?)))
              do (RETURN NEWCSINFO))                                ; Just recursively call ourselves to handle entries in
                                                                    ; DISPLAYFONTCOERCIONS

         ))
   (T ;; One weirdness is, if you have a coercion, and the real-font is missing, you can't get a missingfont coercion on the user-font because the
      ;; real-font missingfont coercion shadows it out.
      (\CREATE-REAL-CHARSET.DISPLAY FAMILY SIZE FACE ROTATION DEVICE CHARSET FONTDESC NOSLUG?])
```

(\CREATE-REAL-CHARSET.DISPLAY

```
[LAMBDA (FAMILY SIZE FACE ROTATION DEVICE CHARSET FONTDESC NOSLUG?)
                                                                    ; Edited 26-Jun-2022 12:37 by rmk
                                                                    ; Edited 15-Jan-88 00:02 by FS
   (COND
      [(AND (EQ ROTATION 0)
            (PROG1 (\READDISPLAYFONTFILE FAMILY SIZE FACE ROTATION 'DISPLAY CHARSET)
                                                                    ; If it is available, this will force the appropriate file to be read to
                                                                    ; fill in the charset entry

                   ]
      (T ;; if we get here, the font is not directly available, either it needs to be rotated, boldified, or italicised 'by hand'. Past that point, we do not
         ;; allow DISPLAYFONTCOERCIONS, only MISSINGxxxxDISPLAYFONTCOERCIONS.
         (PROG (NEWFONT XFONT XLATEDFAM CSINFO)
               (RETURN (COND
                          [(NEQ ROTATION 0)
                           ;; to make a rotated font (even if it is bold or whatnot), recursively call fontcreate to get the unrotated font (maybe
                           ;; bold, etc), then call \SFMAKEROTATEDFONT on the csinfo. If its still missing, then search for missing display
                           ;; font coercions (e.g. no avail. charset, *but*, do not recurse (avoid getting into infinite loops). This allows partial
                           ;; permutations of fonts.
                           (OR (MEMB ROTATION '(90 270))
                               (ERROR "only implemented rotations are 0, 90 and 270." ROTATION))
                           (COND
                              ((SETQ XFONT (\CREATEDISPLAYFONT FAMILY SIZE FACE 0 'DISPLAY CHARSET))
                               ;; Do not call FONTCREATE here. The user might have modified (via PUTCHARBITMAP, etc.) the
                               ;; in-memory version of the source. This also fixes a bug in which several font descriptors ended up sharing
                               ;; bitmaps or charsetvectors, causing havoc when the user modifies either fontdescriptor.
                               (if (SETQ CSINFO (\GETCHARSETINFO CHARSET XFONT T))
                                   then (\SFROTATECSINFO CSINFO ROTATION)
                                 else NIL]
                          ((AND (EQ (fetch WEIGHT of FACE)
                                    'BOLD)
                                (SETQ XFONT (\CREATEDISPLAYFONT FAMILY SIZE (create FONTFACE
                                                                                   using FACE WEIGHT _
                                                                                         'MEDIUM)
                                                                0
                                                                'DISPLAY CHARSET)))
                           ;; if we want a bold font, and the medium weight font is available, build the medium weight version then call
                           ;; \SFMAKEBOLD on the csinfo
                           (if (SETQ CSINFO (\GETCHARSETINFO CHARSET XFONT T))
                               then (\SFMAKEBOLD CSINFO)
                             else NIL))
                          ((AND (EQ (fetch (FONTFACE SLOPE) of FACE)
                                    'ITALIC)
                                (SETQ XFONT (\CREATEDISPLAYFONT FAMILY SIZE (create FONTFACE
                                                                                   using FACE SLOPE _ 'REGULAR)
                                                                0
                                                                'DISPLAY CHARSET)))
                           (if (SETQ CSINFO (\GETCHARSETINFO CHARSET XFONT T))
```

```
                    then (\SFMAKEITALIC CSINFO)
                  else NIL))
            [(AND CHARSET (NOT (EQL CHARSET 0))
                  (for TRANSL in MISSINGCHARSETDISPLAYFONTCOERCIONS bind NEWCSINFO UFONT REALFONT
                     when (AND (SETQ UFONT (CAR TRANSL))
                               (EQ FAMILY (CAR UFONT))
                               (OR (NOT (CADR UFONT))
                                   (EQ SIZE (CADR UFONT)))
                               (OR (NOT (CADDR UFONT))
                                   (EQ CHARSET (CADDR UFONT)))
                               (SETQ REALFONT (CADR TRANSL))
                               (SETQ NEWCSINFO (\CREATE-REAL-CHARSET.DISPLAY (OR (CAR REALFONT)
                                                                                 FAMILY)
                                                                             (OR (CADR REALFONT)
                                                                                 SIZE)
                                                                             FACE ROTATION DEVICE (OR (CADDR REALFONT)
                                                                                                      CHARSET)
                                                                             FONTDESC NOSLUG?)))
                     do (RETURN NEWCSINFO]
            ((for TRANSL in MISSINGDISPLAYFONTCOERCIONS bind NEWCSINFO UFONT REALFONT
                when (AND (SETQ UFONT (CAR TRANSL))
                          (EQ FAMILY (CAR UFONT))
                          (OR (NOT (CADR UFONT))
                              (EQ SIZE (CADR UFONT)))
                          (OR (NOT (CADDR UFONT))
                              (EQ CHARSET (CADDR UFONT)))
                          (SETQ REALFONT (CADR TRANSL))
                          (SETQ NEWCSINFO (\CREATE-REAL-CHARSET.DISPLAY (OR (CAR REALFONT)
                                                                            FAMILY)
                                                                        (OR (CADR REALFONT)
                                                                            SIZE)
                                                                        FACE ROTATION DEVICE (OR (CADDR REALFONT)
                                                                                                 CHARSET)
                                                                        FONTDESC NOSLUG?)))
                do (RETURN NEWCSINFO)))
            ((NOT NOSLUG?)
             (\BUILDSLUGCSINFO (fetch (FONTDESCRIPTOR FONTAVGCHARWIDTH) of FONTDESC)
                    (FONTPROP FONTDESC 'ASCENT)
                    (FONTPROP FONTDESC 'DESCENT)
                    (FONTPROP FONTDESC 'DEVICE])
```

(\BUILDSLUGCSINFO

```
  [LAMBDA (WIDTH ASCENT DESCENT DEVICE SCALE)                    ; Edited  9-May-93 23:12 by rmk:
```

;;; builds a csinfo which contains only the slug (black rectangle) character. Called only for display.

```
    (SETQ SCALE (OR SCALE 1))
    (PROG ((CSINFO (create CHARSETINFO
                           CHARSETASCENT _ ASCENT
                           CHARSETDESCENT _ DESCENT))
           WIDTHS OFFSETS BITMAP IMAGEWIDTHS)
          (SETQ WIDTHS (fetch (CHARSETINFO WIDTHS) of CSINFO))
          (for I from 0 to \MAXTHINCHAR do (\FSETWIDTH WIDTHS I WIDTH))
          (REPLACE IMAGEWIDTHS OF CSINFO WITH WIDTHS)
          (replace (CHARSETINFO OFFSETS) of CSINFO with (SETQ OFFSETS (\CREATECSINFOELEMENT)))
          (for I from 0 to \MAXTHINCHAR do (\FSETOFFSET OFFSETS I 0))
          [replace (CHARSETINFO CHARSETBITMAP) of CSINFO with (SETQ BITMAP (BITMAPCREATE
                                                                             (ROUND (QUOTIENT WIDTH SCALE))
                                                                             (ROUND (QUOTIENT (IPLUS ASCENT DESCENT
                                                                                                     )
                                                                                              SCALE]
          [BLTSHADE BLACKSHADE BITMAP 1 NIL (SUB1 (ROUND (QUOTIENT WIDTH SCALE]
          (RETURN CSINFO])
```

(\SEARCHDISPLAYFONTFILES

```
  [LAMBDA (FAMILY SIZE FACE ROTATION DEVICE)                     ; Edited  5-Mar-87 18:55 by FS
```

;; This function called via APPLY in IMAGESTREAMTYPES.

;; Returns a list of the fonts that can be read in for displaylike devices. Rotation is ignored because it is assumed that all devices support 0 90 ;; and 270.

;; Note we *allow* a device that is not 'DISPLAY for guys like 4DISPLAY, 8DISPLAY, 24DISPLAY, and also possibly for FX80, etc. (guys that ;; want DISPLAYFONTS anyway). Should have some hook though for FONTEXTENSIONS, FONTDIRECTORIES??

```
    (DECLARE (GLOBALVARS DISPLAYFONTEXTENSIONS DISPLAYFONTDIRECTORIES))
    (SELECTQ (SYSTEMTYPE)
         (D (\SEARCHFONTFILES FAMILY SIZE FACE ROTATION DEVICE DISPLAYFONTDIRECTORIES DISPLAYFONTEXTENSIONS))
         (J                                                      (* OLD J code from \READDISPLAYFONT
                                                                 (PROG ((FONTFILE (\FONTFILENAME FAMILY SIZE FACE))
                                                                 FONTDESC STRM) (COND ((SETQ STRM
                                                                 (AND FONTDIRECTORIES (FINDFILE FONTFILE T
                                                                 FONTDIRECTORIES))) (SETQ STRM
                                                                 (OPENSTREAM FONTFILE (QUOTE INPUT)))
                                                                 (SETQ FONTDESC (\READJERICHOFONTFILE FAMILY SIZE
            FACE STRM)) (CLOSEF STRM))) (RETURN FONTDESC)))
            NIL)
```

```
        (SHOULDNT])
```

(\SEARCHFONTFILES

```
[LAMBDA (FAMILY SIZE FACE ROTATION DEVICE DIRLST EXTLST)        ; Edited 14-Sep-96 10:54 by rmk:
                                                                ; Edited  6-Oct-89 12:34 by bvm
```

;; GENERIC FUNCTION

;; returns a list of the fonts that can be read in for a device. Rotation is ignored because it is assumed that all devices support 0 90 and 270.

```
(SETQ FAMILY (\FONTSYMBOL FAMILY))
(SETQ DEVICE (\FONTSYMBOL DEVICE))
(SETQ FACE (\FONTFACE FACE))
(BIND (FILING.ENUEMRATION.DEPTH _ 1)
      FONTSFOUND THISFONT THISFACE FOR E INSIDE EXTLST
   DO [FOR DIR INSIDE DIRLST BIND (FILEPATTERN _ (IF (FMEMB E *OLD-FONT-EXTENSIONS*)
                                                     THEN (\FONTFILENAME.OLD FAMILY SIZE FACE E)
                                                   ELSE (\FONTFILENAME FAMILY SIZE FACE E)))
         DO ;; Hack above to handle both old and new font file names. The variable *OLD-FONT-EXTENSIONS* can be set to suppress
            ;; using the old-style lookup. If set to a list of extensions, just those will be looked up with old-style conventions
            (FOR FONTFILE IN (DIRECTORY (PACKFILENAME.STRING 'DIRECTORY DIR 'BODY FILEPATTERN))
               WHEN [PROGN (SETQ THISFONT (\FONTINFOFROMFILENAME FONTFILE DEVICE))
                           (SETQ THISFACE (CADDR THISFONT))

                           ;; make sure the face, size, and family really match.

                           (AND (NOT (MEMBER THISFONT FONTSFOUND))
                                (OR (EQ FAMILY '*)
                                    (EQ FAMILY (CAR THISFONT)))
                                (OR (EQ SIZE '*)
                                    (EQ SIZE (CADR THISFONT)))
                                (OR (EQ FACE '*)
                                    (EQUAL FACE THISFACE)
                                    (AND (OR (EQ (CAR FACE)
                                                 '*)
                                             (EQ (CAR FACE)
                                                 (CAR THISFACE)))
                                         (OR (EQ (CADR FACE)
                                                 '*)
                                             (EQ (CADR FACE)
                                                 (CADR THISFACE)))
                                         (OR (EQ (CADDR FACE)
                                                 '*)
                                             (EQ (CADDR FACE)
                                                 (CADDR THISFACE]
                  DO (SETQ FONTSFOUND (CONS THISFONT FONTSFOUND]
   FINALLY (RETURN FONTSFOUND])
```

(\FINDFONTFILE

```
[LAMBDA (FAMILY SIZE FACE ROTATION DEVICE CHARSET DIRLST EXTLST)
                                                                ; Edited 14-Sep-96 10:53 by rmk:
                                                                ; Edited  6-Oct-89 11:18 by bvm
```

;; Find any font file on any directory with any naming convention with any extension. Note that ROTATION and DEVICE are just place holders.
;; DEVICE is irrelevant because DIRLST already incorporates the device information. The variable *OLD-FONT-EXTENSIONS* can be set to
;; suppress using the old-style lookup. If set to a list of extensions, just those will be looked up with old-style conventions.

```
(BIND FONTFILE FOR EXT INSIDE EXTLST WHEN (SETQ FONTFILE
                                                (FINDFILE (IF (FMEMB EXT *OLD-FONT-EXTENSIONS*)
                                                              THEN (\FONTFILENAME.OLD FAMILY SIZE FACE EXT
                                                                                      CHARSET)
                                                            ELSE (\FONTFILENAME FAMILY SIZE FACE EXT CHARSET))
                                                          T DIRLST))
   DO (RETURN FONTFILE])
```

(\FONTSYMBOL

```
[LAMBDA (X ElseReturnXFlg)                                      ; Edited 28-Jul-88 11:59 by rmk:
                                                                ; Edited 24-Mar-87 14:32 by FS
```

;; Return a symbol in IL package and is in uppercase. Currently the function IL:U-CASE is believed to do this, but if it changes, this is the font
;; hook. ElseReturnXFlg is if you want an IL symbol if X is a symbol or string, otherwise just X.

```
(COND
   ((LITATOM X)
    (U-CASE X))
   ((STRINGP X)
    (MKATOM (U-CASE X)))
   (ElseReturnXFlg X)
   (T (ERROR "Want an IL symbol"])
```

(\DEVICESYMBOL

```
[LAMBDA (X ElseReturnXFlg)                                      ; Edited  7-Oct-88 20:07 by rmk:
                                                                ; Edited 28-Jul-88 14:43 by rmk:
                                                                ; Edited 24-Mar-87 14:33 by FS
```

;; Return a canonicalized atom good for comparing with DEVICE symbols

```
(LET ((STRM (\GETSTREAM X 'OUTPUT T)))
```

```
        (COND
           (STRM (fetch (IMAGEOPS IMFONTCREATE) of (fetch (STREAM IMAGEOPS) of STRM)))
           ((NULL X)
            'DISPLAY)
           (T                                                    ; because its used in ASSOC.
              (\FONTSYMBOL X ElseReturnXFlg])
```

(\FONTFACE

```
  [LAMBDA (FACE NOERRORFLG DEV)                                  ; Edited  1-Aug-88 09:44 by rmk:
                                                                 ; Edited 28-Jul-88 15:50 by rmk:
                                                                 ; Edited 28-Jul-88 15:49 by rmk:
                                                                 ; Edited 28-Jul-88 15:41 by rmk:
                                                                 ; Edited 28-Jul-88 15:38 by rmk:
                                                                 ; Edited 28-Jul-88 14:44 by rmk:
                                                                 ; Edited 25-Feb-87 22:58 by FS
```

;; Coerces FACE into standard FONTFACE record, usually returns a CONSTANT (so you'd better not RPLACD or REPLACE fields!!)

```
(PROG (UNKNOWN (WEIGHT 'MEDIUM)
               (SLOPE 'REGULAR)
               (EXPANSION 'REGULAR)
               (OLDFACE FACE))
```

;; On error, can signal, or return NIL, or return REGULAR face.

```
      [SETQ UNKNOWN (COND
                       ((EQ NOERRORFLG 'REGULAR)
                        'REGULAR)
                       (T 'ERROR]
      [COND
         ((type? FONTFACE FACE)
```

;; List Case. Unpack because want to validate fields

```
          (SETQ WEIGHT (fetch (FONTFACE WEIGHT) of FACE))
          (SETQ SLOPE (fetch (FONTFACE SLOPE) of FACE))
          (SETQ EXPANSION (fetch (FONTFACE EXPANSION) of FACE))
```

;; Handle unknown faces

```
          [OR (\FONT.SYMBOLMEMB WEIGHT '                         (* BOLD MEDIUM LIGHT))
              (SETQ WEIGHT (COND
                              ((\FONT.COMPARESYMBOL WEIGHT 'REGULAR)
```

;; Clean up WEIGHT REGULAR vs. MEDIUM

```
                               (SETQ WEIGHT 'MEDIUM))
                              (T UNKNOWN]
          (OR (\FONT.SYMBOLMEMB SLOPE '                          (* REGULAR ITALIC))
              (SETQ SLOPE UNKNOWN))
          (OR (\FONT.SYMBOLMEMB EXPANSION '                      (* COMPRESSED REGULAR EXPANDED))
              (SETQ EXPANSION UNKNOWN)))
         ((OR (LITATOM FACE)
              (STRINGP FACE))
          (COND
             ((NULL FACE)                                        ; Fast vanilla default

              )
             ((EQ (NCHARS FACE)
                  3)                                             ; 3 char notation case
              (SETQ WEIGHT (SELCHARQ (CHCON1 FACE)
                                ((B b)
                                     'BOLD)
                                ((M m R r)
                                     'MEDIUM)
                                ((L l)
                                     'LIGHT)
                                UNKNOWN))
              (SETQ SLOPE (SELCHARQ (NTHCHARCODE FACE 2)
                                ((R r)
                                     'REGULAR)
                                ((I i)
                                     'ITALIC)
                                UNKNOWN))
              (SETQ EXPANSION (SELCHARQ (NTHCHARCODE FACE 3)
                                     ((R r)
                                          'REGULAR)
                                     ((C c)
                                          'COMPRESSED)
                                     ((E e)
                                          'EXPANDED)
                                     UNKNOWN)))
             ((SELECTQ FACE
                   (BOLD (SETQ WEIGHT 'BOLD))
                   (ITALIC (SETQ SLOPE 'ITALIC))
                   (BOLDITALIC (SETQ WEIGHT 'BOLD)
                               (SETQ SLOPE 'ITALIC))
                   ((STANDARD REGULAR)
                       T)
                   NIL))
             ((\FONT.COMPARESYMBOL FACE 'BOLD)
```

```
                (SETQ WEIGHT 'BOLD))
               ((\FONT.COMPARESYMBOL FACE 'ITALIC)
                (SETQ SLOPE 'ITALIC))
               ((\FONT.COMPARESYMBOL FACE 'BOLDITALIC)
                (SETQ WEIGHT 'BOLD)
                (SETQ SLOPE 'ITALIC))
               ((\FONT.SYMBOLMEMB FACE '(STANDARD REGULAR NIL NNN))
                                                          ; Vanilla case

                )
               ((STRPOS "-" FACE)                         ; Color fontface spec!
                (SETQ FACE (\FONTFACE.COLOR FACE NOERRORFLG DEV))
                (RETURN FACE))
               ((\FONT.SYMBOLMEMB FACE '                  (* ***))
                                                          ; Wildcard case
                (SETQ WEIGHT '*)
                (SETQ SLOPE '*)
                (SETQ EXPANSION '*))
               (T                                         ; Other litatom error case
                  (SETQ WEIGHT UNKNOWN)
                  (SETQ SLOPE UNKNOWN)
                  (SETQ EXPANSION UNKNOWN]
        (if (OR (EQ WEIGHT 'ERROR)
                (EQ SLOPE 'ERROR)
                (EQ EXPANSION 'ERROR))
            then (if NOERRORFLG
                     then (RETURN NIL)
                   else (\ILLEGAL.ARG OLDFACE)))
```

;; Avoid consing by returning constant faces (historical: really, would have been better to return MRR, but users have know about this for too long ;; (rmk))

```
        (RETURN (COND
                   ((AND (EQ WEIGHT 'MEDIUM)
                         (EQ SLOPE 'REGULAR)
                         (EQ EXPANSION 'REGULAR))         ; MRR
                    (CONSTANT (create FONTFACE)))
                   [(AND (EQ WEIGHT 'BOLD)
                         (EQ SLOPE 'REGULAR)
                         (EQ EXPANSION 'REGULAR))         ; BRR
                    (CONSTANT (create FONTFACE
                                      WEIGHT _ 'BOLD]
                   [(AND (EQ WEIGHT 'MEDIUM)
                         (EQ SLOPE 'ITALIC)
                         (EQ EXPANSION 'REGULAR))         ; MIR
                    (CONSTANT (create FONTFACE
                                      SLOPE _ 'ITALIC]
                   [(AND (EQ WEIGHT 'BOLD)
                         (EQ SLOPE 'ITALIC)
                         (EQ EXPANSION 'REGULAR))         ; BIR
                    (CONSTANT (create FONTFACE
                                      WEIGHT _ 'BOLD
                                      SLOPE _ 'ITALIC]
                   (T                                     ; Otherwise, cons up
                      (create FONTFACE
                              WEIGHT _ WEIGHT
                              SLOPE _ SLOPE
                              EXPANSION _ EXPANSION])
```

(\FONTFACE.COLOR

```
  [LAMBDA (FACE NOERRORFLG DEV)                           ; Edited 28-Jul-88 14:51 by rmk:
                                                          ; Edited 28-Jul-88 13:09 by rmk:
                                                          ; Edited 24-Mar-87 17:03 by FS
```

;; This used to be \FONTFACE. Renamed \FONTFACE.COLOR, and \FONTFACE rewritten. The section below should also be redone

;; Takes a variety of user specifications and converts them to a standard FONTFACE record.

;; b/w fontfaces are extended by an optional '-backcolor-forecolor'

;; the atom NNN is interpreted the same as NIL or MRR to cover up a bug described in AR 3025, the FONTNNN bug

```
(DECLARE (GLOBALVARS \COLORDISPLAYSTREAMTYPES))
(SETQ DEV (\DEVICESYMBOL DEV))
(PROG (BWFACE POS OLDPOS BITSPERPIXEL BACKCOLOR FORECOLOR ANSWER)
```

;; First get a FONTFACE ANSWER.

```
        [SETQ ANSWER (COND
                        ((type? FONTFACE FACE)
                         FACE)
                        ((LITATOM FACE)
                         (OR (U-CASEP FACE)
                             (SETQ FACE (U-CASE FACE)))
                         (SETQ POS (STRPOS "-" FACE))
                         (COND
                            [POS (SETQ BWFACE (SUBATOM FACE 1 (SUB1 POS]
                            (T (SETQ BWFACE FACE)))
                         [SETQ ANSWER (SELECTQ BWFACE
                                              ((* ***)
                                                   (CONSTANT (create FONTFACE
```

```
                                                    WEIGHT _ '*
                                                    SLOPE _ '*
                                                    EXPANSION _ '*)))
                                        ((NIL MRR STANDARD NNN)
                                            (CONSTANT (create FONTFACE)))
                                        ((ITALIC MIR)
                                            (CONSTANT (create FONTFACE
                                                    SLOPE _ 'ITALIC)))
                                        ((BOLD BRR)
                                            (CONSTANT (create FONTFACE
                                                    WEIGHT _ 'BOLD)))
                                        ((BOLDITALIC BIR)
                                            (CONSTANT (create FONTFACE
                                                    WEIGHT _ 'BOLD
                                                    SLOPE _ 'ITALIC)))
                                        (create FONTFACE
                                               WEIGHT _ (SELCHARQ (NTHCHARCODE FACE 1)
                                                           (M 'MEDIUM)
                                                           (B 'BOLD)
                                                           (L 'LIGHT)
                                                           (GO ERROR))
                                               SLOPE _ (SELCHARQ (NTHCHARCODE FACE 2)
                                                          (R 'REGULAR)
                                                          (I 'ITALIC)
                                                          (GO ERROR))
                                               EXPANSION _ (SELCHARQ (NTHCHARCODE FACE 3)
                                                                (R 'REGULAR)
                                                                (C 'COMPRESSED)
                                                                (E 'EXPANDED)
                                                                (GO ERROR]
                                (COND
                                   (POS                                  ; Color FONTFACE. *
                                        (SETQ OLDPOS POS)
                                        (SETQ POS (STRPOS "-" FACE (ADD1 OLDPOS)))
                                        (COND
                                           ((NULL POS)
                                            (GO ERROR)))
                                        (SETQ BITSPERPIXEL (\DISPLAYSTREAMTYPEBPP DEV))
                                        (SETQ BACKCOLOR (COLORNUMBERP (SUBATOM FACE (ADD1 OLDPOS)
                                                                                   (SUB1 POS))
                                                                      BITSPERPIXEL))
                                        (SETQ OLDPOS POS)
                                        (SETQ FORECOLOR (COLORNUMBERP (SUBATOM FACE (ADD1 OLDPOS)
                                                                                   -1)
                                                                      BITSPERPIXEL))
                                                                          ; COPY ANSWER to avoid smashing constants.
                                        (SETQ ANSWER (COPY ANSWER))
                                        (replace (FONTFACE BACKCOLOR) of ANSWER with BACKCOLOR)
                                        (replace (FONTFACE FORECOLOR) of ANSWER with FORECOLOR)))
                                 ANSWER)
                                (T (GO ERROR]
```

;; Coerce on or off COLOR.

```
      (SETQ ANSWER (COND
                       ((AND (NOT (FMEMB DEV \COLORDISPLAYSTREAMTYPES))
                             (fetch (FONTFACE COLOR) of ANSWER))
                        (SETQ ANSWER (COPY ANSWER))
                        (replace (FONTFACE COLOR) of ANSWER with NIL)
                        ANSWER)
                       ((AND (FMEMB DEV \COLORDISPLAYSTREAMTYPES)
                             (NULL (fetch (FONTFACE COLOR) of ANSWER)))
                        (SETQ FACE (COPY FACE))
                        (replace (FONTFACE BACKCOLOR) of ANSWER with 0)
                        (replace (FONTFACE FORECOLOR) of ANSWER with (MAXIMUMCOLOR (\DISPLAYSTREAMTYPEBPP DEV))
                         )
                        ANSWER)
                       (T ANSWER)))
      (RETURN ANSWER)
    ERROR
      (COND
         (NOERRORFLG (RETURN NIL))
         (T (\ILLEGAL.ARG FACE])
```

(**FONTFILENAME**

```
  [LAMBDA (FAMILY SIZE FACE EXTENSION CHARSET)                    ; Edited  5-Mar-93 16:10 by rmk:
```

;; Strike file naming convention (w/o dashes, no charset) no longer supported. New name is of the form "familysize-face-Ccharset.ext", e.g., ;; MODERN12-MRR-C357.WD

;; **bvm 10/5/89 Slight change: partition fonts into subdirectories by charset, e.g., all Charset zero fonts are in subdirectory C0>. This significantly ;; speeds up any font operation that requires any local directory work (e.g., NFS servers on both Sun and D machine), and FONTSAVAILABLE on ;; any device (since it doesn't have to wade thru all those charsets). This behavior is conditioned on the value of *USEOLDFONTDIRECTORIES*

```
    (SETQ FACE (\FONTFACE FACE))                                  ; Validate face
    (LET* ([SIZEPATT (COND
                         ((EQ SIZE '*)
                          SIZE)
                         ((FIXP SIZE)
```

```
                        (if (< SIZE 10)
                            then (CONCAT 0 SIZE)
                          else SIZE))
                      (T (\ILLEGAL.ARG SIZE]
          (CSETNAME (COND
                      ((OR (NULL CHARSET)
                           (EQ CHARSET 0))                    ; Charset defaults to zero.
                       "0")
                      ((FIXP CHARSET)
                       (LET ((*PRINT-BASE* 8)
                             (*PRINT-RADIX* NIL))             ; Longhand for (cl:write-to-string charset :radix nil :base 8), which
                                                              ; is twice as slow, due to lousy keyword handling
                            (\PRINDATUM.TO.STRING CHARSET)))
                      (T                                      ; Somebody made the string already?
                         CHARSET)))
          [FACESPEC (LIST (CHCON1 (fetch (FONTFACE WEIGHT) of FACE))
                          (CHCON1 (fetch (FONTFACE SLOPE) of FACE))
                          (CHCON1 (fetch (FONTFACE EXPANSION) of FACE]
          (TAIL FACESPEC))
         [if (OR (EQ (CAR TAIL)
                     (CHARCODE *))
                 (EQ (CAR (SETQ TAIL (CDR TAIL)))
                     (CHARCODE *)))
             then                                             ; Avoid adjacent wildcards because some devices (notably DSK)
                                                              ; get exponentially slower.
                  (while (EQ (CADR TAIL)
                             (CHARCODE *))
                     do (RPLACD TAIL (CDDR TAIL]
         ;; Fortunately, CONCAT ignores packages.
         (PACKFILENAME.STRING 'NAME (CONCAT (CL:IF *USEOLDFONTDIRECTORIES*
                                                  ""
                                                  (CONCAT (PROGN ; Lowercase because it's in a directory name, so maybe Unix will
                                                                 ; find it sooner?
                                                                 "c")
                                                          CSETNAME ">"))
                                            FAMILY SIZEPATT "-" (CONCATCODES FACESPEC)
                                            "-C" CSETNAME)
                'EXTENSION EXTENSION])
```

(**FONTFILENAME.OLD**

```
  [LAMBDA (FAMILY SIZE FACE EXTENSION CHARSET)                ; Edited 23-Sep-92 18:22 by jds
     ;; Returns old style font file names.  They were ambiguous because you could not ask for e.g. FACE (MEDIUM * REGULAR) because it maps to
     ;; FamilySize-*-Charset, which also matches (BOLD * COMPRESSED), etc.  Keep this function around though for user's who don't rename their
     ;; files.                                                ; Returns the name of the file that should contain the information
                                                             ; for a font.
     (SETQ FACE (\FONTFACE FACE))                            ; Force legal canonical face
     (SETQ FACE (COND
                   ((AND (EQ (CAR FACE)
                             '*)
                         (EQ (CADR FACE)
                             '*))
                    ;; Avoid  adjacent wildcards because DSK gets slower exponentially (can take loooong tiiiiime).  No need to check
                    ;; compression.
                    '*)
                   (T FACE)))
     (PACKFILENAME.STRING 'NAME [PROGN                       ;; DISPLAYFONT AC WD and the default case
                                      (CONCAT (CDR (SASSOC FAMILY *DISPLAY-FONT-NAME-MAP*))
                                              (COND
                                                 ((EQ SIZE '*)
                                                  SIZE)
                                                 ((FIXP SIZE)
                                                  (COND
                                                     ((< SIZE 10)
                                                      (CONCAT 0 SIZE))
                                                     (T SIZE)))
                                                 (T (\ILLEGAL.ARG SIZE)))
                                              [COND
                                                 ((EQ FACE '*)
                                                  '*)
                                                 (T (SELECTQ (fetch WEIGHT of FACE)
                                                         (BOLD (SELECTQ (fetch SLOPE of FACE)
                                                                    (ITALIC "D")
                                                                    "B"))
                                                         (SELECTQ (fetch SLOPE of FACE)
                                                              (ITALIC "I")
                                                              "R"]
                                              (COND
                                                 ((FIXP CHARSET)
                                                  (LET ((*PRINT-BASE* 8))
                                                       (CL:FORMAT NIL "~O" CHARSET)))
                                                 (T "000"]
          'EXTENSION EXTENSION])
```

(\FONTFILENAME.NEW

```
  [LAMBDA (FAMILY SIZE FACE EXTENSION CHARSET)                    ; Edited 30-Mar-87 20:00 by FS
    ;; Strike file naming convention (w/o dashes, no charset) no longer supported.
    (LET (NAME SIZEPATT)
         (SETQ FACE (\FONTFACE FACE))                             ; Validate face
         [SETQ SIZEPATT (COND
                           ((EQ SIZE '*)
                            SIZE)
                           ((FIXP SIZE)
                            (if (< SIZE 10)
                                then (CONCAT 0 SIZE)
                              else SIZE))
                           (T (\ILLEGAL.ARG SIZE]
        ;; Avoid adjacent wildcards because some devices (notably DSK) get exponentially slower.  Nicely, PACK & CONCAT ignore packages.
        (PACKFILENAME.STRING 'NAME (CONCAT FAMILY SIZEPATT "-" [COND
                                                                  ((EQUAL FACE '
                                                                                '*)
                                                                  (T (CONCAT (NTHCHAR (fetch (FONTFACE WEIGHT)
                                                                                         of FACE)
                                                                                      1)
                                                                             (NTHCHAR (fetch (FONTFACE SLOPE)
                                                                                         of FACE)
                                                                                      1)
                                                                             (NTHCHAR (fetch (FONTFACE EXPANSION)
                                                                                         of FACE)
                                                                                      1]
                                                               (COND
                                                                  [(FIXP CHARSET)
                                                                   (LET ((*PRINT-BASE* 8))
                                                                        (CONCAT "-C" (\PRINDATUM.TO.STRING CHARSET]
                                                                  (CHARSET (CONCAT "-C" CHARSET))
                                                                  (T "-C0")))
                                                               '.EXTENSION EXTENSION])
          (* * *)
)
```

(\FONTINFOFROMFILENAME

```
  [LAMBDA (FONTFILE DEVICE)                                       ; Edited 14-Sep-96 10:23 by rmk:
                                                                  ; Edited  5-Oct-89 18:28 by bvm
    ;; returns a list of the family size face rotation device of the font stored in the file name FONTFILE.  Rotation is 0 always.  Parses both new & old
    ;; format files.
    (LET ((FILENAMELIST (UNPACKFILENAME.STRING FONTFILE))
          CH SIZEBEG SIZEND NAME FAMILY SIZE FACE EXT)
         (SETQ NAME (LISTGET FILENAMELIST 'NAME))                 ; find where the name and size are.  MUST check for ch nil
                                                                  ; below or possible infinite loop
         (SETQ SIZEBEG (for CH# from 1 when (OR (NUMBERP (SETQ CH (NTHCHAR NAME CH#)))
                                                (NULL CH))
                          do (RETURN CH#)))
        ;; Get Family
        [SETQ FAMILY (MKATOM (U-CASE (SUBSTRING NAME 1 (SUB1 SIZEBEG]
        ;; Get Size
        [SETQ SIZEND (find CH# from SIZEBEG suchthat (NOT (NUMBERP (NTHCHAR NAME CH#]
        [SETQ SIZE (MKATOM (SUBSTRING NAME SIZEBEG (SUB1 SIZEND]
        (if (EQ (NTHCHAR NAME SIZEND)
                '-)
            then (SETQ SIZEND (ADD1 SIZEND)))
        ;; Get Face
        (SETQ NAME (U-CASE NAME))                                 ; don't need name, but checks for lowercase face
        [SETQ FACE (LIST (COND
                            ((STRPOS "B" NAME SIZEND NIL T NIL UPPERCASEARRAY)
                             'BOLD)
                            ((STRPOS "L" NAME SIZEND NIL T NIL UPPERCASEARRAY)
                             'LIGHT)
                            (T 'MEDIUM))
                         (COND
                            ((STRPOS "I" NAME SIZEND NIL NIL NIL UPPERCASEARRAY)
                             'ITALIC)
                            (T 'REGULAR))
                         (COND
                            ((STRPOS "E" NAME SIZEND NIL NIL NIL UPPERCASEARRAY)
                             'EXPANDED)
                            ((STRPOS "C-" NAME SIZEND NIL NIL NIL UPPERCASEARRAY)
                             'COMPRESSED)
                            (T 'REGULAR]
        (LIST FAMILY SIZE FACE 0 (COND
                                    ((STREAMP DEVICE)
                                     (IMAGESTREAMTYPE DEVICE))
                                    ((NULL DEVICE)
```

```
                                    [SETQ EXT (MKATOM (U-CASE (LISTGET FILENAMELIST 'EXTENSION]
                                    (SELECTQ EXT
                                         (WD 'INTERPRESS)
                                         ((STRIKE AC DISPLAYFONT)
                                             'DISPLAY)
                                         EXT))
                                   ((LITATOM DEVICE)
                                    (\FONTSYMBOL DEVICE))
                                   (T DEVICE])
```

(\FONTINFOFROMFILENAME.OLD

```
[LAMBDA (FONTFILE DEVICE)                                   ; Edited  1-Jan-87 01:29 by FS
```

;; returns a list of the family size face rotation device of the font stored in the file name FONTFILE.

```
(PROG ((FILENAMELIST (UNPACKFILENAME FONTFILE))
       SIZEBEG SIZEND NAME FAMILY SIZE)
      (SETQ NAME (LISTGET FILENAMELIST 'NAME))              ; find where the name and size are.
      (SETQ SIZEBEG (for CH# from 1 when (NUMBERP (NTHCHAR NAME CH#)) do (RETURN CH#)))
      [SETQ FAMILY (MKATOM (SUBSTRING NAME 1 (SUB1 SIZEBEG]
      (SETQ SIZEND (for CH# from SIZEBEG when (NOT (NUMBERP (NTHCHAR NAME CH#))) do (RETURN CH#)))
      [SETQ SIZE (MKATOM (SUBSTRING NAME SIZEBEG (SUB1 SIZEND]
      (RETURN (LIST FAMILY SIZE (SELECTQ (LISTGET FILENAMELIST 'EXTENSION)
                                         ((DISPLAYFONT AC WD)
                                             (LIST (COND
                                                       ((STRPOS "-B" NAME SIZEND NIL T)
                                                        'BOLD)
                                                       (T 'MEDIUM))
                                                   (COND
                                                       ((STRPOS "-I" NAME SIZEND NIL)
                                                        'ITALIC)
                                                       (T 'REGULAR))
                                                   'REGULAR))
                                         (LIST (COND
                                                   ((STRPOS "B" NAME SIZEND NIL T)
                                                    'BOLD)
                                                   (T 'MEDIUM))
                                               (COND
                                                   ((STRPOS "I" NAME SIZEND NIL)
                                                    'ITALIC)
                                                   (T 'REGULAR))
                                               'REGULAR))
                         0 DEVICE])
```

(\GETFONTDESC

```
[LAMBDA (SPEC DEVICE NOERRORFLG)                            (* J.Gibbons " 5-Dec-82 16:53")
```

;; Coerces SPEC to a fontdescriptor ; \GETFONTDESC HAS MACRO, BUT OLD CALLS STILL EXIST

```
(\COERCEFONTDESC SPEC DEVICE NOERRORFLG])
```

(\COERCEFONTDESC

```
[LAMBDA (SPEC STREAM NOERRORFLG)                            ; Edited 29-Aug-91 12:19 by jds
```

;; Coerces SPEC to a fontdescriptor appropriate for STREAM. Go back thru FONTCREATE for various coercions in order to make sure that the ;; cache gets set up

```
(DECLARE (GLOBALVARS DEFAULTFONT))
(PROG (FONT DEVICE)
      [COND
         ((type? FONTDESCRIPTOR SPEC)
          (SETQ FONT SPEC))
         [(type? FONTCLASS SPEC)
          [SETQ DEVICE (COND
                           ((NULL STREAM)                   ; Default is display
                            ;; COULDN'T THIS BRANCH BE INTENDED TO MEAN 4DISPLAY, 8DISPLAY, 24DISPLAY?  PEOPLE
                            ;; PROBABLY SHOULDN'T BE CALLING \COERCEFONTDESC WITH STREAM = NIL.
                            'DISPLAY)
                           ((IMAGESTREAMP STREAM)
                            (IMAGESTREAMTYPE STREAM))
                           ((LITATOM STREAM)
                            (\DEVICESYMBOL STREAM))
                           (STREAM STREAM)
                           (T                               ;; I don't think this case should be allowed.
                              'DISPLAY]
          [SETQ FONT (SELECTQ DEVICE
                          (DISPLAY (fetch (FONTCLASS DISPLAYFD) of SPEC))
                          (INTERPRESS (fetch (FONTCLASS INTERPRESSFD) of SPEC))
                          (PRESS (fetch (FONTCLASS PRESSFD) of SPEC))
                          (CDR (SASSOC DEVICE (fetch (FONTCLASS OTHERFDS) of SPEC]
          (RETURN (COND
                      ((type? FONTDESCRIPTOR FONT)
                       ;; We don't always create FD's for devices before they are needed, so do it now and save result
                       FONT)
```

```
                        [(NULL FONT)
                         ;; NIL means create FONT but don't cache.
                         (COND
                            ((AND (FMEMB DEVICE \DISPLAYSTREAMTYPES)
                                  (SETQ FONT (\COERCEFONTDESC SPEC 'DISPLAY NOERRORFLG))
                                  (SETQ FONT (FONTCOPY FONT 'DEVICE STREAM 'NOERROR NOERRORFLG)))
                             ;; Coerce existing black & white font to color font, but don't cache.
                             FONT)
                            [(EQ SPEC DEFAULTFONT)
                             ;; Break cycles with NIL in the defaultfont
                             (COND
                                (NOERRORFLG NIL)
                                ((EQ DEVICE 'DISPLAY)
                                 ;; Function DEFAULTFONT guarantees system integrity
                                 (DEFAULTFONT 'DISPLAY))
                                ((EQUAL DEVICE '(HARDCOPY DISPLAY))
                                 ;; MAKE DISPLAY-HARDCOPY FONTS default to the corresponding display font, copied....
                                 (FONTCOPY (DEFAULTFONT 'DISPLAY)
                                        'DEVICE STREAM 'NOERROR NOERRORFLG))
                                (T (ERROR (CONCAT DEVICE " component for DEFAULTFONT undefined"]
                            (T (FONTCREATE DEFAULTFONT NIL NIL NIL STREAM NOERRORFLG]
                        ((SETQ FONT (FONTCREATE FONT NIL NIL NIL STREAM NOERRORFLG))
                         ;; Might get NIL if NOERRORFLG
                         (SETFONTCLASSCOMPONENT SPEC DEVICE FONT]
                ((NULL SPEC)
                 (RETURN (\COERCEFONTDESC DEFAULTFONT STREAM NOERRORFLG)))
                ((OR (IMAGESTREAMP SPEC)
                     (type? WINDOW SPEC))
                 (SETQ FONT (DSPFONT NIL SPEC)))
                (T ;; If called with NOERRORFLG=T (e.g. from DSPFONT) we want to suppress invalid arg errors as well as font not found, so we
                   ;; can move on to other possible coercions.
                   (RETURN (FONTCREATE SPEC NIL NIL NIL STREAM NOERRORFLG]
    ;; Here if arg was a fontdescriptor or imagestream
          (RETURN (COND
                     ((NULL STREAM)
                      ;; NIL device doesn't default to display if a fully-specified font was found
                      FONT)
                     ([OR (EQ STREAM (fetch (FONTDESCRIPTOR FONTDEVICE) of FONT))
                          (AND (type? STREAM STREAM)
                               (EQ (fetch (IMAGEOPS IMFONTCREATE) of (fetch (STREAM IMAGEOPS) of STREAM))
                                   (fetch (FONTDESCRIPTOR FONTDEVICE) of FONT]
                      FONT)
                     (T ;; Here if doesn't match or if DEVICE is not explicitly a stream.
                        (FONTCOPY FONT 'DEVICE STREAM 'NOERROR NOERRORFLG])
```

(\LOOKUPFONT

```
  [LAMBDA (FAMILY SIZE FACE ROTATION DEVICE)                    ; Edited 28-Jul-88 17:05 by rmk:
                                                                ; Edited 28-Jul-88 17:04 by rmk:
                                                                ; Edited 28-Jul-88 14:44 by rmk:
                                                                ; Edited 28-Jul-88 14:02 by rmk:
                                                                ; Edited 28-Jul-88 13:54 by rmk:
                                                                ; Edited 26-Feb-87 00:20 by FS
    ;; looks up a font in the internal cache.  SASSOC for listp FACE
    (DECLARE (GLOBALVARS \FONTSINCORE))
    ;; Someone had better have already made FACE canonical
    (LET [(X (CDR (FASSOC ROTATION (CDR (SASSOC FACE (CDR (FASSOC SIZE (CDR (OR (FASSOC FAMILY \FONTSINCORE)
                                                                                 (\FONT.SYMBOLASSOC FAMILY
                                                                                        \FONTSINCORE]
         (CDR (OR (FASSOC DEVICE X)
                  (\FONT.SYMBOLASSOC DEVICE X])
```

(\LOOKUPFONTSINCORE

```
  [LAMBDA (FAMILY SIZE FACE ROTATION DEVICE)                    ; Edited 25-Apr-93 13:07 by rmk:
                                                                (* rrb "25-Sep-84 12:10")
```

;;; returns a list of the fonts that are available in core. * is used to match anything.

```
    (DECLARE (GLOBALVARS \FONTSINCORE))
    (for FAMBUCKET in \FONTSINCORE when (OR (EQ FAMILY '*)
                                           (EQ FAMILY (CAR FAMBUCKET)))
       join (for SIZEBUCKET in (CDR FAMBUCKET) when (OR (EQ SIZE '*)
                                                        (EQ SIZE (CAR SIZEBUCKET)))
               join (for FACEBUCKET in (CDR SIZEBUCKET) when (OR (EQ FACE '*)
```

```
                                                    (EQUAL FACE (CAR FACEBUCKET)))
                        join (for ROTBUCKET in (CDR FACEBUCKET) when (OR (EQ ROTATION '*)
                                                                         (EQ ROTATION (CAR ROTBUCKET)))
                                  join (for DEVBUCKET in (CDR ROTBUCKET) when (AND (OR (EQ DEVICE '*)
                                                                                       (EQ DEVICE (CAR DEVBUCKET)))
                                                                                   (TYPE? FONTDESCRIPTOR (CDR DEVBUCKET)
                                                                                          ))
                                        collect (LIST (CAR FAMBUCKET)
                                                      (CAR SIZEBUCKET)
                                                      (CAR FACEBUCKET)
                                                      (CAR ROTBUCKET)
                                                      (CAR DEVBUCKET])
```

(**READDISPLAYFONTFILE**

```
[LAMBDA (FAMILY SIZE FACE ROTATION DEVICE CHARSET)          ; Edited  8-Oct-96 10:17 by rmk:
                                                            ; Edited 30-Sep-96 12:03 by kaplan
                                                            ; Edited  2-Jan-87 17:55 by FS
```

;; Look for new filename convention, then old file name convention, with extensions. If CACHEDISPLAYFONTS, this keeps a cache of what was
;; read, on the canonical filename's property list, so that NSDISPLAYSIZES and SMALLSCREEN size coercions can be done and undone without
;; always going out to the directories.

```
(DECLARE (GLOBALVARS DISPLAYFONTEXTENSIONS DISPLAYFONTDIRECTORIES CACHEDISPLAYFONTS))
(BIND FONTFILE CSINFO STRM
    FIRST ;; Cache is indexed by canonical font file name, without the extension fields.
        (CL:WHEN [AND CACHEDISPLAYFONTS
                      (FIND EXT INSIDE DISPLAYFONTEXTENSIONS
                         SUCHTHAT (SETQ CSINFO
                                        (GETP (L-CASE (FILENAMEFIELD (IF (FMEMB EXT *OLD-FONT-EXTENSIONS*)
                                                                         THEN (\FONTFILENAME.OLD FAMILY SIZE
                                                                                                 FACE EXT CHARSET)
                                                                       ELSE (\FONTFILENAME FAMILY SIZE FACE
                                                                                           EXT CHARSET))
                                                                     'NAME))
                                              'CACHEDCHARSET]
            (RETURN (AND (NEQ CSINFO T)
                         (COPYALL CSINFO))))
    FOR EXT INSIDE DISPLAYFONTEXTENSIONS WHEN (SETQ FONTFILE (\FINDFONTFILE FAMILY SIZE FACE ROTATION DEVICE
                                                                            CHARSET DISPLAYFONTDIRECTORIES
                                                                            (LIST EXT)))
    DO ;; Cache is indexed by canonical font file name, without the directory or extension fields
        (SETQ STRM (OPENSTREAM FONTFILE 'INPUT))
        (RESETLST
            [SETQ CSINFO (SELECTQ (FONTFILEFORMAT STRM T)
                                  (STRIKE (RESETSAVE NIL (LIST (FUNCTION CLOSEF)
                                                               STRM))
                                          (\READSTRIKEFONTFILE STRM FAMILY SIZE FACE))
                                  (AC ;; CLOSEF is guaranteed inside \READACFONTFILE, against the possibility that we have to copy to
                                      ;; make randaccessp
                                      (\READACFONTFILE STRM FAMILY SIZE FACE))
                                  (PROG1 (CLOSEF STRM)          ; This would get done by RESETSAVE if AC's were read
                                                                ; sequentially and we could factor the RESETSAVE
                                      (SHOULDNT))])
        (CL:WHEN CACHEDISPLAYFONTS
            (PUTPROP (L-CASE (FILENAMEFIELD FONTFILE 'NAME))
                     'CACHEDCHARSET CSINFO)
            (SETQ CSINFO (COPYALL CSINFO)))
        ;; If not a recognizable format, I guess we should keep looking for another possible extension, altho it would also be nice to tell the user
        ;; that he has a bogus file.
        (RETURN CSINFO)
    FINALLY
            ;; Didn't find the file, cache T to suppress future lookups
            (CL:WHEN CACHEDISPLAYFONTS
                (PUTPROP (L-CASE (FILENAMEFIELD (IF (FMEMB (CAR (MKLIST DISPLAYFONTEXTENSIONS))
                                                           *OLD-FONT-EXTENSIONS*)
                                                    THEN (\FONTFILENAME.OLD FAMILY SIZE FACE
                                                                            (CAR (MKLIST DISPLAYFONTEXTENSIONS))
                                                                            CHARSET)
                                                  ELSE (\FONTFILENAME FAMILY SIZE FACE (CAR (MKLIST
                                                                                              DISPLAYFONTEXTENSIONS
                                                                                                  ))
                                                                      CHARSET))
                                                'NAME))
                         'CACHEDCHARSET T))])

)

;; \FINDFONTFILE \FONTFILENAME \SEARCHFONTFILES \FONTINFOFROMFILENAME are redefined to deal with character-set directories. That
;; behavior is conditioned on the setting of the global variable *USEOLDFONTDIRECTORIES*, T at PARC, maybe NIL most other places.

(ADDTOVAR **OLD-FONT-EXTENSIONS*** STRIKE)
```

```
(RPAQ? *USEOLDFONTDIRECTORIES* NIL)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS *OLD-FONT-EXTENSIONS* *USEOLDFONTDIRECTORIES*)
)
```

;; Establishes DISPLAYFONTFILECACHE to avoid rereading charsets when size coercions are done (e.g. for nsdisplaysizes or smallscreen)

;; Establishes DISPLAYFONTFILECACHE to avoid rereading charsets when size coercions are done (e.g. for nsdisplaysizes or smallscreen)

```
(RPAQ? CACHEDISPLAYFONTS )

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS CACHEDISPLAYFONTS)
)
```

;; STRIKE format file support

```
(DEFINEQ
```

(\READSTRIKEFONTFILE

```
  [LAMBDA (STRM FAMILY SIZE FACE)                                    ; Edited 12-Jul-2022 09:19 by rmk
                                                                     ; Edited  4-Dec-92 12:11 by jds
                                                                     ; STRM has already been determined to be a vanilla
                                                                     ; strike-format file.
                                                                     ; returns a charsetinfo
    (COND
       ((NEQ 2 (GETFILEPTR STRM))
        (SETFILEPTR STRM 2)))
    (LET (CSINFO NUMBCODES RW BITMAP OFFSETS FIRSTCHAR LASTCHAR HEIGHT WIDTHS)
         (SETQ CSINFO (create CHARSETINFO))
         (SETQ FIRSTCHAR (\WIN STRM))                                ; minimum ascii code
         (SETQ LASTCHAR (\WIN STRM))                                 ; maximum ascii code
         (\WIN STRM)                                                 ; MaxWidth which isn't used by anyone.
         (\WIN STRM)                                                 ; number of words in this StrikeBody
         (replace (CHARSETINFO CHARSETASCENT) of CSINFO with (\WIN STRM))
                                                                     ; ascent in scan lines (=FBBdy+FBBoy)
         (replace (CHARSETINFO CHARSETDESCENT) of CSINFO with (\WIN STRM))
                                                                     ; descent in scan-lines (=FBBoy)
         (\WIN STRM)                                                 ; offset in bits (<0 for kerning, else 0, =FBBox)
         (SETQ RW (\WIN STRM))                                       ; raster width of bitmap
                                                                     ; height of bitmap
         ;; JDS 12/4/92:  Apparently, these fields can be signed values, if all chars, e.g., ride above the base line.
         (SETQ HEIGHT (IPLUS (SIGNED (fetch (CHARSETINFO CHARSETASCENT) of CSINFO)
                                     16)
                             (SIGNED (fetch (CHARSETINFO CHARSETDESCENT) of CSINFO)
                                     16)))
         (SETQ BITMAP (BITMAPCREATE (UNFOLD RW BITSPERWORD)
                                    HEIGHT))
         (\BINS STRM (fetch BITMAPBASE of BITMAP)
                0
                (UNFOLD (ITIMES RW HEIGHT)
                        BYTESPERWORD))                               ; read bits into bitmap
         (replace (CHARSETINFO CHARSETBITMAP) of CSINFO with BITMAP)
         (SETQ NUMBCODES (IPLUS (IDIFFERENCE LASTCHAR FIRSTCHAR)
                                3))                                  ; (SETQ OFFSETS (ARRAY (IPLUS \MAXCHAR 3) (QUOTE
                                                                     ; SMALLPOSP) 0 0))
         (SETQ OFFSETS (fetch (CHARSETINFO OFFSETS) of CSINFO))      ; initialise the offsets to 0
         (for I from 0 to (IPLUS \MAXTHINCHAR 2) do (\FSETOFFSET OFFSETS I 0))
                                                                     ; (AIN OFFSETS FIRSTCHAR NUMBCODES STRM)
         (for I from FIRSTCHAR as J from 1 to NUMBCODES do (\FSETOFFSET OFFSETS I (\WIN STRM)))
         (SETQ WIDTHS (fetch (CHARSETINFO WIDTHS) of CSINFO))
         (for I from 0 to (IPLUS \MAXTHINCHAR 2) do (\FSETWIDTH WIDTHS I 0))
                                                                     ; (replace WIDTHS of (CHARSETINFO CSINFO) with (ARRAY
                                                                     ; (IPLUS \MAXCHAR 3) (QUOTE SMALLPOSP) 0 0))
         (\FONTRESETCHARWIDTHS CSINFO FIRSTCHAR LASTCHAR)
         (replace (CHARSETINFO IMAGEWIDTHS) of CSINFO with (fetch (CHARSETINFO WIDTHS) of CSINFO))
        CSINFO])
```

(\SFMAKEBOLD

```
  [LAMBDA (CSINFO)                                                   (* gbn "25-Jul-85 04:52")
    (PROG* ((OLDCHARBITMAP (fetch (CHARSETINFO CHARSETBITMAP) of CSINFO))
            (WIDTHS (fetch (CHARSETINFO WIDTHS) of CSINFO))
            (OFFSETS (fetch (CHARSETINFO OFFSETS) of CSINFO))
            (HEIGHT (IPLUS (fetch (CHARSETINFO CHARSETASCENT) of CSINFO)
                           (fetch (CHARSETINFO CHARSETDESCENT) of CSINFO)))
            NEWCHARBITMAP OFFSET UNKNOWNOFFSET UNKNOWNWIDTH)
           (SETQ NEWCHARBITMAP (BITMAPCREATE (fetch BITMAPWIDTH of OLDCHARBITMAP)
                                             (fetch BITMAPHEIGHT of OLDCHARBITMAP)))
           (SETQ UNKNOWNOFFSET (\FGETOFFSET OFFSETS (ADD1 \MAXCHAR)))
           (SETQ UNKNOWNWIDTH (\FGETWIDTH WIDTHS (ADD1 \MAXCHAR)))
           [for I from 0 to \MAXCHAR do (COND
                                          ((EQ (SETQ OFFSET (\FGETOFFSET OFFSETS I))
```

```
                                        UNKNOWNOFFSET)      ; if this is the magic charcode with the slug image (charcode 256)
                                                            ; then leave it alone
                                     NIL)
                                     (T                     ; overlap two blts to produce bold effect
                                        (BITBLT OLDCHARBITMAP OFFSET 0 NEWCHARBITMAP OFFSET 0
                                               (\FGETWIDTH WIDTHS I)
                                               HEIGHT
                                               'INPUT
                                               'REPLACE)
                                        (BITBLT OLDCHARBITMAP OFFSET 0 NEWCHARBITMAP (ADD1 OFFSET)
                                               0
                                               (SUB1 (\FGETWIDTH WIDTHS I))
                                               HEIGHT
                                               'INPUT
                                               'PAINT]      ; fill in the slug for the magic charcode
          (BITBLT OLDCHARBITMAP UNKNOWNOFFSET 0 NEWCHARBITMAP UNKNOWNOFFSET 0 UNKNOWNWIDTH HEIGHT 'INPUT
                 'REPLACE)
          (RETURN (create CHARSETINFO using CSINFO CHARSETBITMAP _ NEWCHARBITMAP])
```

(\SFMAKEITALIC

```
  [LAMBDA (CSINFO)                                          (* gbn "18-Sep-85 17:57")
    (PROG ((WIDTHS (fetch (CHARSETINFO WIDTHS) of CSINFO))
           (OFFSETS (fetch (CHARSETINFO OFFSETS) of CSINFO))
           (ASCENT (fetch (CHARSETINFO CHARSETASCENT) of CSINFO))
           (DESCENT (fetch (CHARSETINFO CHARSETDESCENT) of CSINFO))
           (OLDBITMAP (fetch (CHARSETINFO CHARSETBITMAP) of CSINFO))
           HEIGHT OFFSET NEWBITMAP WIDTH UNKNOWNOFFSET UNKNOWNWIDTH N M R XN XX YN YX)
          (SETQ HEIGHT (IPLUS ASCENT DESCENT))
          (SETQ NEWBITMAP (BITMAPCREATE (fetch BITMAPWIDTH of OLDBITMAP)
                                 (fetch BITMAPHEIGHT of OLDBITMAP)))
          (SETQ UNKNOWNOFFSET (\FGETOFFSET OFFSETS (ADD1 \MAXTHINCHAR)))
          (SETQ UNKNOWNWIDTH (\FGETWIDTH WIDTHS (ADD1 \MAXTHINCHAR)))
          (SETQ N (IDIFFERENCE 0 (IQUOTIENT (IPLUS DESCENT 3)
                                            4)))
          (SETQ M (IQUOTIENT (IPLUS ASCENT 3)
                             4))
          [for I from 0 to \MAXTHINCHAR
             do (COND
                   ((EQ (SETQ OFFSET (\FGETOFFSET OFFSETS I))
                        UNKNOWNOFFSET)                      ; if this is the magic charcode with the slug image (charcode 256)
                                                            ; then leave it alone
                    NIL)
                   (T (SETQ WIDTH (\FGETWIDTH WIDTHS I))
                      (for J from N to M do (SETQ R (IPLUS OFFSET WIDTH))
                                            (SETQ XN (IMIN R (IMAX (IPLUS OFFSET J)
                                                                   0)))
                                            (SETQ XX (IMIN R (IMAX (IPLUS R J)
                                                                   0)))
                                            [SETQ YN (IMAX 0 (IPLUS DESCENT (ITIMES J 4]
                                            [SETQ YX (IMIN HEIGHT (IPLUS DESCENT (IPLUS (ITIMES J 4)
                                                                                        4]
                                            (COND
                                               ((AND (IGREATERP XX XN)
                                                     (IGREATERP YX YN))
                                                (BITBLT OLDBITMAP OFFSET YN NEWBITMAP XN YN (IDIFFERENCE XX XN)
                                                       (IDIFFERENCE YX YN)
                                                       'INPUT
                                                       'REPLACE]
          (BITBLT OLDBITMAP UNKNOWNOFFSET 0 NEWBITMAP UNKNOWNOFFSET 0 UNKNOWNWIDTH HEIGHT 'INPUT 'REPLACE)
          (RETURN (create CHARSETINFO using CSINFO CHARSETBITMAP _ NEWBITMAP])
```

(\SFMAKEROTATEDFONT

```
  [LAMBDA (FONTDESC ROTATION)                               ; Edited 30-Mar-87 20:35 by FS
    ;; takes a fontdecriptor and rotates it.
    ;; 1/5/86 JDS.  Masterscope claims nobody calls this.  Let's find out....
    (HELP "ROTATED fonts need to be fixed for NS Chars & New FONTDESCRIPTOR fields")
                                                            (* (create FONTDESCRIPTOR using FONTDESC
                                                            (SETQ CHARACTERBITMAP (
                                                            \SFROTATEFONTCHARACTERS
                                                            (fetch (FONTDESCRIPTOR CHARACTERBITMAP) of
                                                            FONTDESC) ROTATION)) (SETQ ROTATION ROTATION)
                                                            (SETQ \SFOffsets (\SFFIXOFFSETSAFTERROTATION
                                                            FONTDESC ROTATION)) (SETQ FONTCHARSETVECTOR
                                                            (\ALLOCBLOCK (ADD1 \MAXCHARSET) T))))
    ;; If you uncomment out the code above, remove this comment and the NIL below
    NIL])
```

(\SFROTATECSINFO

```
  [LAMBDA (CSINFO ROTATION)                                 (* gbn "15-Sep-85 14:38")
    ;; takes a CHARSETINFO and rotates it and produces a rotated equivalent one.
    (create CHARSETINFO using CSINFO CHARSETBITMAP _ (\SFROTATEFONTCHARACTERS (fetch (CHARSETINFO CHARSETBITMAP)
```

```
                                                                    of CSINFO)
                                                ROTATION)
                        OFFSETS _ (\SFROTATECSINFOOFFSETS CSINFO ROTATION])
```

(\SFROTATEFONTCHARACTERS

```
  [LAMBDA (CHARBITMAP ROTATION)                                    ; Edited 22-Sep-87 10:38 by Snow
```

;;; rotate a bitmap either 90 or 270 for fonts.

```
    (CASE ROTATION
        (0 CHARBITMAP)
        (90 (ROTATE-BITMAP-LEFT CHARBITMAP))
        (180 (ROTATE-BITMAP (ROTATE-BITMAP CHARBITMAP)))
        (270 (ROTATE-BITMAP CHARBITMAP)))])
```

(\SFFIXOFFSETSAFTERROTATION

```
  [LAMBDA (FONTDESC ROTATION)                                      ; Edited 30-Mar-87 20:35 by FS
```

;; adjusts offsets in case where rotation turned things around.

```
    (HELP "NEED TO UPDATE THIS FN TO NSCHARS & NEW FONT FIELDS") (* (COND ((EQ ROTATION 270)
                                                                (PROG ((OFFSETS (fetch (FONTDESCRIPTOR \SFOffsets) of
    FONTDESC)) (WIDTHS (fetch (FONTDESCRIPTOR \SFWidths) of FONTDESC)) (BITMAPHEIGHT (BITMAPWIDTH
                                                                (fetch (FONTDESCRIPTOR CHARACTERBITMAP) of
                                                                FONTDESC))) NEWOFFSETS) (SETQ NEWOFFSETS
                                                                (COPYARRAY OFFSETS)) (for CHARCODE from 0 to
                                                                \MAXCHAR do (SETA NEWOFFSETS CHARCODE
                                                                (IDIFFERENCE BITMAPHEIGHT
                                                                (IPLUS (ELT OFFSETS CHARCODE)
                                                                (ELT WIDTHS CHARCODE))))) (* ;
                                                                "may be some problem with dummy character representation.")
                                                                (RETURN NEWOFFSETS))) (T (fetch
                                                                (FONTDESCRIPTOR \SFOffsets) of FONTDESC))))
```

;; If you uncomment out the code above, remove this comment and the NIL below

```
    NIL])
```

(\SFROTATECSINFOOFFSETS

```
  [LAMBDA (CSINFO ROTATION)                                        (* gbn "15-Sep-85 14:36")
                                                                   ; adjusts offsets in case where rotation turned things around.
    (COND
       ((EQ ROTATION 270)
        (PROG ((OFFSETS (fetch (CHARSETINFO OFFSETS) of CSINFO))
               (WIDTHS (fetch (CHARSETINFO WIDTHS) of CSINFO))
               (BITMAPHEIGHT (BITMAPWIDTH (fetch (CHARSETINFO CHARSETBITMAP) of CSINFO)))
               NEWOFFSETS)
              (SETQ NEWOFFSETS (\CREATECSINFOELEMENT))
              [for CHARCODE from 0 to \MAXCHAR do (\FSETOFFSET NEWOFFSETS CHARCODE (IDIFFERENCE
                                                                                    BITMAPHEIGHT
                                                                                    (IPLUS (\FGETOFFSET OFFSETS
                                                                                                  CHARCODE)
                                                                                           (\FGETWIDTH WIDTHS
                                                                                                  CHARCODE]
                                                                   ; may be some problem with dummy character representation.
              (RETURN NEWOFFSETS)))
       (T (fetch (CHARSETINFO OFFSETS) of CSINFO])
```

(\SFMAKECOLOR

```
  [LAMBDA (BWCSINFO BACKCOLOR FORECOLOR BITSPERPIXEL)              (* kbr%: " 6-Feb-86 18:17")
```

;; makes a csinfo that has a character bitmap that is colorized.

```
    (PROG (CHARACTERBITMAP COLORCSINFO)
          [COND
             ((IMAGESTREAMP BITSPERPIXEL)
              (OR BACKCOLOR (SETQ BACKCOLOR (DSPBACKCOLOR NIL BITSPERPIXEL)))
              (OR FORECOLOR (SETQ FORECOLOR (DSPCOLOR NIL BITSPERPIXEL)))
              (SETQ BITSPERPIXEL (IMAGESTREAMTYPE BITSPERPIXEL]
          [SETQ BITSPERPIXEL (COND
                                ((NUMBERP BITSPERPIXEL)
                                 BITSPERPIXEL)
                                (T (\DISPLAYSTREAMTYPEBPP BITSPERPIXEL]
          (SETQ BACKCOLOR (COLORNUMBERP BACKCOLOR BITSPERPIXEL))
          (SETQ FORECOLOR (COLORNUMBERP FORECOLOR BITSPERPIXEL))
          (SETQ CHARACTERBITMAP (COLORIZEBITMAP (fetch (CHARSETINFO CHARSETBITMAP) of BWCSINFO)
                                       BACKCOLOR FORECOLOR BITSPERPIXEL))
          (SETQ COLORCSINFO (create CHARSETINFO using BWCSINFO CHARSETBITMAP _ CHARACTERBITMAP))
          (RETURN COLORCSINFO])

)

(DEFINEQ
```

(WRITESTRIKEFONTFILE

```
  [LAMBDA (FONT CHARSET FILE)                                      ; Edited 12-Jul-2022 14:36 by rmk
```

```
                                                                    (* kbr%: "21-Oct-85 15:08")
                                                                    ; Write strike FILE using info in FONT. *
    (CL:UNLESS (FONTP FONT)
          (LISPERROR "ILLEGAL ARG" FONT))
    (CL:UNLESS CHARSET (SETQ CHARSET 0))
    (CL:UNLESS (AND (IGEQ CHARSET 0)
                    (ILEQ CHARSET \MAXCHARSET))
          (LISPERROR "ILLEGAL ARG" CHARSET))
    (LET (STREAM CSINFO FIRSTCHAR LASTCHAR WIDTHS MAXWIDTH LENGTH RASTERWIDTH DUMMYCHAR DUMMYOFFSET
                 PREVIOUSOFFSET OFFSETS)
         (SETQ CSINFO (\GETCHARSETINFO CHARSET FONT T))
         (CL:UNLESS CSINFO (ERROR "Couldn't find charset " CHARSET))
         (SETQ WIDTHS (fetch (CHARSETINFO WIDTHS) of CSINFO))
         (SETQ OFFSETS (fetch (CHARSETINFO OFFSETS) of CSINFO))
         (SETQ DUMMYOFFSET (\FGETOFFSET OFFSETS DUMMYINDEX))
         [SETQ FIRSTCHAR (for I from 0 to MAXCODE thereis (NOT (EQ (\FGETOFFSET OFFSETS I)
                                                                   DUMMYOFFSET]
         [SETQ LASTCHAR (for I from MAXCODE to 0 by -1 thereis (NOT (EQ (\FGETOFFSET OFFSETS I)
                                                                        DUMMYOFFSET]
         (SETQ DUMMYCHAR (ADD1 LASTCHAR))
         [SETQ STREAM (OPENSTREAM FILE 'OUTPUT 'NEW '((TYPE BINARY]
         (\WOUT STREAM 32768)                                       ; STRIKE HEADER. *
         (\WOUT STREAM FIRSTCHAR)
         (\WOUT STREAM LASTCHAR)
         (SETQ MAXWIDTH 0)
         [for I from 0 to DUMMYINDEX do (SETQ MAXWIDTH (IMAX MAXWIDTH (\FGETWIDTH WIDTHS I]
         (\WOUT STREAM MAXWIDTH)                                    ; STRIKE BODY. *
                                                                    ; Length. *
         (SETQ RASTERWIDTH (fetch (BITMAP BITMAPRASTERWIDTH) of (fetch (CHARSETINFO CHARSETBITMAP) of CSINFO)))
         (SETQ LENGTH (IPLUS 8 (IDIFFERENCE LASTCHAR FIRSTCHAR)
                             (ITIMES (fetch (FONTDESCRIPTOR \SFHeight) of FONT)
                                     RASTERWIDTH)))
         (\WOUT STREAM LENGTH)                                      ; Ascent, Descent, Xoffset (no longer used) and Rasterwidth. *
         (\WOUT STREAM (fetch (CHARSETINFO CHARSETASCENT) of CSINFO))
         (\WOUT STREAM (fetch (CHARSETINFO CHARSETDESCENT) of CSINFO))
         (\WOUT STREAM 0)
         (\WOUT STREAM RASTERWIDTH)                                 ; Bitmap. *
         [\BOUTS STREAM (fetch (BITMAP BITMAPBASE) of (fetch (CHARSETINFO CHARSETBITMAP) of CSINFO))
                0
                (ITIMES 2 RASTERWIDTH (IPLUS (fetch (CHARSETINFO CHARSETASCENT) of CSINFO)
                                             (fetch (CHARSETINFO CHARSETDESCENT) of CSINFO]
                                                                    ; Offsets. *
         (for I WIDTH OFFSET (CODE _ 0) from FIRSTCHAR to DUMMYCHAR first (\WOUT STREAM CODE)
            do (SETQ OFFSET (\FGETOFFSET OFFSETS I))
               (SETQ WIDTH (\FGETWIDTH WIDTHS I))
               (CL:UNLESS (AND (IEQP OFFSET DUMMYOFFSET)
                               (NOT (IEQP I DUMMYCHAR)))
                      (ADD CODE WIDTH))
               (\WOUT STREAM CODE))
         (CLOSEF STREAM])
```

(**STRIKECSINFO**

```
  [LAMBDA (CSINFO)                                                  ; Edited 27-Apr-89 13:39 by atm
    ;; Returns a STRIKE type font descriptor (EQ WIDTHS IMAGEWIDTHS), cause we know how to write those guys out (they read quicker  but
    ;; display slower).  If (EQ WIDTHS IMAGEWIDTHS), just return original.
    (PROG (WIDTHS OFFSETS IMWIDTHS OLDBM BMWIDTH BMHEIGHT NEWBM NEWOFFSET NEWWIDTH OLDOFFSET DUMMYOFFSET
                  NEWOFFSETS)
          (SETQ WIDTHS (fetch (CHARSETINFO WIDTHS) of CSINFO))
          (SETQ IMWIDTHS (fetch (CHARSETINFO IMAGEWIDTHS) of CSINFO))
          (if (EQ WIDTHS IMWIDTHS)
              then (RETURN CSINFO))
          (SETQ OFFSETS (fetch (CHARSETINFO OFFSETS) of CSINFO))
          (SETQ OLDBM (fetch (CHARSETINFO CHARSETBITMAP) of CSINFO))
          (SETQ DUMMYOFFSET (\FGETOFFSET OFFSETS 256))
          (SETQ BMHEIGHT (BITMAPHEIGHT OLDBM))
          [SETQ BMWIDTH (for I from 0 to \MAXTHINCHAR sum (if (IEQP DUMMYOFFSET (\FGETOFFSET OFFSETS I))
                                                              then 0
                                                            else (IMAX (\FGETIMAGEWIDTH IMWIDTHS I)
                                                                       (\FGETWIDTH WIDTHS I]
     ;;
     ;; Initialize new offsets vector
     ;;
          (SETQ NEWOFFSETS (\CREATECSINFOELEMENT))
          (for I from 0 to (IPLUS \MAXTHINCHAR 2) do (\FSETOFFSET NEWOFFSETS I 0))
          (\FSETOFFSET NEWOFFSETS (ADD1 \MAXTHINCHAR)
                 BMWIDTH)
     ;;
     ;; Adjust bitmap with so width = imagewidth, fill offsets
     ;;
          (SETQ NEWBM (BITMAPCREATE BMWIDTH BMHEIGHT 1))
          (SETQ NEWOFFSET 0)
```

```
          [for I from 0 to 255 do (SETQ OLDOFFSET (\FGETOFFSET OFFSETS I))
                                  (if (IEQP DUMMYOFFSET OLDOFFSET)
                                      then (\FSETOFFSET NEWOFFSETS I BMWIDTH)
                                    else (\FSETOFFSET NEWOFFSETS I NEWOFFSET)
                                         (SETQ NEWWIDTH (IMAX (\FGETIMAGEWIDTH IMWIDTHS I)
                                                              (\FGETWIDTH WIDTHS I)))
                                         (BITBLT OLDBM OLDOFFSET 0 NEWBM NEWOFFSET 0 (\FGETWIDTH IMWIDTHS I)
                                                BMHEIGHT
                                                'REPLACE)
                                         (SETQ NEWOFFSET (IPLUS NEWOFFSET NEWWIDTH]
     ;;
     ;; Make new CSInfo record withs IMAGEWIDTHS, WIDTHS the same
     ;;
          (SETQ WIDTHS (COPYALL WIDTHS))
          [for I from 0 to \MAXTHINCHAR do (\FSETWIDTH WIDTHS I (IMAX (\FGETWIDTH WIDTHS I)
                                                                      (\FGETIMAGEWIDTH IMWIDTHS I]
          (RETURN (create CHARSETINFO
                          WIDTHS _ WIDTHS
                          OFFSETS _ NEWOFFSETS
                          IMAGEWIDTHS _ WIDTHS
                          CHARSETBITMAP _ NEWBM
                          YWIDTHS _ (fetch (CHARSETINFO YWIDTHS) of CSINFO)
                          CHARSETASCENT _ (fetch (CHARSETINFO CHARSETASCENT) of CSINFO)
                          CHARSETDESCENT _ (fetch (CHARSETINFO CHARSETDESCENT) of CSINFO])

)

(/DECLAREDATATYPE 'FONTCLASS '(BYTE POINTER POINTER POINTER POINTER POINTER)
       ;; ---field descriptor list elided by lister---
       '12)

(/DECLAREDATATYPE 'FONTDESCRIPTOR
       '(POINTER POINTER POINTER POINTER WORD WORD WORD WORD SIGNEDWORD SIGNEDWORD SIGNEDWORD SIGNEDWORD POINTER
              POINTER POINTER POINTER POINTER (BITS 8)
              WORD POINTER POINTER POINTER)
       ;; ---field descriptor list elided by lister---
       '34)

(/DECLAREDATATYPE 'CHARSETINFO '(POINTER POINTER POINTER POINTER POINTER WORD WORD POINTER)
       ;; ---field descriptor list elided by lister---
       '14)

(ADDTOVAR SYSTEMRECLST
          (DATATYPE FONTCLASS ((PRETTYFONT# BYTE)
                               DISPLAYFD PRESSFD INTERPRESSFD OTHERFDS FONTCLASSNAME))
          (DATATYPE FONTDESCRIPTOR ((FONTDEVICE POINTER)
                                    (FONTFAMILY POINTER)
                                    (FONTSIZE POINTER)
                                    (FONTFACE POINTER)
                                    (\SFAscent WORD)
                                    (\SFDescent WORD)
                                    (\SFHeight WORD)
                                    (ROTATION WORD)
                                    (FBBOX SIGNEDWORD)
                                    (FBBOY SIGNEDWORD)
                                    (FBBDX SIGNEDWORD)
                                    (FBBDY SIGNEDWORD)
                                    (\SFLKerns POINTER)
                                    (\SFRWidths POINTER)
                                    (FONTDEVICESPEC POINTER)
                                    (OTHERDEVICEFONTPROPS POINTER)
                                    (FONTSCALE POINTER)
                                    (\SFFACECODE BITS 8)
                                    (FONTAVGCHARWIDTH WORD)
                                    (FONTIMAGEWIDTHS POINTER)
                                    (FONTCHARSETVECTOR POINTER)
                                    (FONTEXTRAFIELD2 POINTER)))
          (DATATYPE CHARSETINFO (WIDTHS OFFSETS IMAGEWIDTHS CHARSETBITMAP YWIDTHS (CHARSETASCENT WORD)
                                        (CHARSETDESCENT WORD)
                                        LEFTKERN)))

(RPAQ? \FONTSINCORE )

(RPAQ? \DEFAULTDEVICEFONTS )

(RPAQ? \UNITWIDTHSVECTOR )

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS DISPLAYFONTDIRECTORIES \DEFAULTDEVICEFONTS \UNITWIDTHSVECTOR)
)
```

```
(DECLARE%: DONTEVAL@LOAD DOCOPY

(\UNITWIDTHSVECTOR)
)

(DECLARE%: EVAL@COMPILE

(RPAQQ NORUNCODE 255)

(CONSTANTS (NORUNCODE 255))
)
```

;; FOLLOWING DEFINITIONS EXPORTED

```
(DEFOPTIMIZER FONTPROP (&REST ARGS)
                       (SELECTQ (AND (EQ (CAADR ARGS)
                                         'QUOTE)
                                     (CADADR ARGS))
                            (ASCENT (LIST 'FONTASCENT (CAR ARGS)))
                            (DESCENT (LIST 'FONTDESCENT (CAR ARGS)))
                            (HEIGHT (LIST 'FONTHEIGHT (CAR ARGS)))
                            'IGNOREMACRO))
```

;; END EXPORTED DEFINITIONS

```
(DECLARE%: DONTCOPY
```

;; FOLLOWING DEFINITIONS EXPORTED

```
(DECLARE%: EVAL@COMPILE

(DATATYPE FONTCLASS ((PRETTYFONT# BYTE)
                     DISPLAYFD PRESSFD INTERPRESSFD OTHERFDS FONTCLASSNAME))

(DATATYPE FONTDESCRIPTOR ((FONTDEVICE POINTER)
                          (FONTFAMILY POINTER)
                          (FONTSIZE POINTER)
                          (FONTFACE POINTER)
                          (\SFAscent WORD)
                          (\SFDescent WORD)
                          (\SFHeight WORD)
                          (ROTATION WORD)
                          (FBBOX SIGNEDWORD)
                          (FBBOY SIGNEDWORD)
                          (FBBDX SIGNEDWORD)
                          (FBBDY SIGNEDWORD)
                          (\SFLKerns POINTER)
                          (\SFRWidths POINTER)
                          (FONTDEVICESPEC POINTER)              ; Holds the spec by which the font is known to the printing
                                                                ; device, if coercion has been done
                          (OTHERDEVICEFONTPROPS POINTER)        ; For individual devices to hang special information
                          (FONTSCALE POINTER)
                          (\SFFACECODE BITS 8)
                          (FONTAVGCHARWIDTH WORD)               ; Set in FONTCREATE, used to fix up the linelength when
                                                                ; DSPFONT is called
                          (FONTIMAGEWIDTHS POINTER)             ; This is the image width, as opposed to the advanced width;
                                                                ; initial hack for accents, kerning.  Fields is referenced by
                                                                ; FONTCREATE.
                          (FONTCHARSETVECTOR POINTER)           ; A 256-pointer block, with one pointer per 'character set' --each
                                                                ; group of 256 character codes.  Each pointer is either NIL if
                                                                ; there's no info for that charset, or is a CHARSETINFO,
                                                                ; containing widths, char bitmap, etc for the characters in that
                                                                ; charset.
                          (FONTEXTRAFIELD2 POINTER))
       FONTCHARSETVECTOR _ (\CREATEFONTCHARSETVECTOR))

(RECORD FONTFACE (WEIGHT SLOPE EXPANSION)
       [ACCESSFNS ((COLOR (CDDDR DATUM)
                          (RPLACD (CDDR DATUM)
                                  NEWVALUE))
                   (BACKCOLOR [COND
                                  ((CDDDR DATUM)
                                   (CAR (CDDDR DATUM]
                                  (PROGN [COND
                                             ((NULL (CDDDR DATUM))
                                              (RPLACD (CDDR DATUM)
                                                      (LIST NIL NIL]
                                             (RPLACA (CDDDR DATUM)
                                                     NEWVALUE)))
                   (FORECOLOR [COND
                                  ((CDDDR DATUM)
                                   (CADR (CDDDR DATUM]
                                  (PROGN [COND
                                             ((NULL (CDDDR DATUM))
                                              (RPLACD (CDDR DATUM)
```

```
                                        (LIST NIL NIL]
                                    (RPLACA (CDR (CDDDR DATUM))
                                           NEWVALUE]
       WEIGHT _ 'MEDIUM SLOPE _ 'REGULAR EXPANSION _ 'REGULAR (TYPE? LISTP))

(DATATYPE CHARSETINFO (WIDTHS                                   ; The advance-width of each character, an array indexed by
                                                                ; charcode.  Usually the same as the imagewidth, but can differ
                                                                ; for accents, kerns kerns.  This is what should be used for
                                                                ; stringwidth calculations.
                             OFFSETS                            ; Offset of each character into the image bitmap;  X value of left
                                                                ; edge
                             IMAGEWIDTHS                        ; imagewidths is not automagically allocated since it is not
                                                                ; always needed
                             CHARSETBITMAP                      ; Bitmap containing the character images, indexed by OFFSETS
                             YWIDTHS
                             (CHARSETASCENT WORD)               ; Max ascent for all characters in this CHARSET
                             (CHARSETDESCENT WORD)              ; Max descent for all characters in this CHARSET
                             LEFTKERN)
       WIDTHS _ (\CREATECSINFOELEMENT)
       OFFSETS _ (\CREATECSINFOELEMENT))
)

(/DECLAREDATATYPE 'FONTCLASS '(BYTE POINTER POINTER POINTER POINTER POINTER)
       ;; ---field descriptor list elided by lister---
       '12)

(/DECLAREDATATYPE 'FONTDESCRIPTOR
       '(POINTER POINTER POINTER POINTER WORD WORD WORD WORD SIGNEDWORD SIGNEDWORD SIGNEDWORD SIGNEDWORD POINTER
              POINTER POINTER POINTER POINTER (BITS 8)
              WORD POINTER POINTER POINTER)
       ;; ---field descriptor list elided by lister---
       '34)

(/DECLAREDATATYPE 'CHARSETINFO '(POINTER POINTER POINTER POINTER POINTER WORD WORD POINTER)
       ;; ---field descriptor list elided by lister---
       '14)

(DECLARE%: EVAL@COMPILE

(PUTPROPS FONTASCENT MACRO ((FONTSPEC)
                            (ffetch \SFAscent of (\GETFONTDESC FONTSPEC))))

(PUTPROPS FONTDESCENT MACRO ((FONTSPEC)
                             (ffetch \SFDescent of (\GETFONTDESC FONTSPEC))))

(PUTPROPS FONTHEIGHT MACRO ((FONTSPEC)
                            (ffetch \SFHeight of (\GETFONTDESC FONTSPEC))))

(PUTPROPS \FGETOFFSET DMACRO ((OFFSETSBLOCK CHAR8CODE)
                              (\GETBASE OFFSETSBLOCK CHAR8CODE)))

(PUTPROPS \FSETOFFSET DMACRO ((OFFSETSBLOCK CHAR8CODE OFFSET)
                              (\PUTBASE OFFSETSBLOCK CHAR8CODE OFFSET)))

(PUTPROPS \FGETWIDTH DMACRO ((WIDTHSBLOCK CHAR8CODE)
                             (\GETBASE WIDTHSBLOCK CHAR8CODE)))

(PUTPROPS \FSETWIDTH DMACRO ((WIDTHSBLOCK INDEX WIDTH)
                             (\PUTBASE WIDTHSBLOCK INDEX WIDTH)))

(PUTPROPS \FGETCHARWIDTH MACRO (OPENLAMBDA (FONTDESC CHARCODE)
                                  (\FGETWIDTH (ffetch (CHARSETINFO WIDTHS) of (\GETCHARSETINFO (\CHARSET
                                                                                                CHARCODE)
                                                                                              FONTDESC))
                                         (\CHAR8CODE CHARCODE))))

(PUTPROPS \FSETCHARWIDTH MACRO (OPENLAMBDA (FONTDESC CHARCODE WIDTH)
                                  (\FSETWIDTH (ffetch (CHARSETINFO WIDTHS) of (\GETCHARSETINFO (\CHARSET CHARCODE
                                                                                                )
                                                                                              FONTDESC))
                                         (\CHAR8CODE CHARCODE)
                                         WIDTH)))

(PUTPROPS \FGETIMAGEWIDTH MACRO ((IMAGEWIDTHSBLOCK CHAR8CODE)
                                 (\GETBASE IMAGEWIDTHSBLOCK CHAR8CODE)))

(PUTPROPS \FSETIMAGEWIDTH DMACRO ((WIDTHSBLOCK INDEX WIDTH)
                                  (\PUTBASE WIDTHSBLOCK INDEX WIDTH)))

(PUTPROPS \GETCHARSETINFO MACRO ((CHARSET FONTDESC NOSLUG?)
                                 ;; fetches the charsetinfo for charset CHARSET in fontdescriptor FONTDESC.  If NIL, then creates the
                                 ;; required charset.
                                 ;; NOSLUG?  means don't create an empty (slug) csinfo if the charset is not found, just return NIL
```

```
                                    (OR (\GETBASEPTR (ffetch FONTCHARSETVECTOR of FONTDESC)
                                              (UNFOLD CHARSET 2))
                                        (\CREATECHARSET CHARSET FONTDESC NOSLUG?))))

(PUTPROPS \CREATECSINFOELEMENT MACRO (NIL (\ALLOCBLOCK (FOLDHI (IPLUS \MAXTHINCHAR 3)
                                                               WORDSPERCELL))))

(PUTPROPS \CREATEFONTCHARSETVECTOR MACRO (NIL                ; Allocates a block for the character set records
                                            (\ALLOCBLOCK (ADD1 \MAXCHARSET)
                                                   T)))
)


(DEFMACRO \CREATEKERNELEMENT ()
   `(CL:MAKE-ARRAY (IPLUS \MAXTHINCHAR 3)
           :ELEMENT-TYPE
           '(SIGNED-BYTE 16)
           :INITIAL-ELEMENT 0))


(DEFMACRO \FSETLEFTKERN (LEFTKERNBLOCK INDEX KERNVALUE)
   `(CL:SETF (CL:AREF ,LEFTKERNBLOCK ,INDEX)
           ,KERNVALUE))


(DEFMACRO \FGETLEFTKERN (LEFTKERNBLOCK CHAR8CODE)
   `(CL:AREF ,LEFTKERNBLOCK ,CHAR8CODE))

(DECLARE%: EVAL@COMPILE

(RPAQQ \MAXNSCHAR 65535)

(CONSTANTS (\MAXNSCHAR 65535))
)
)
```

;; END EXPORTED DEFINITIONS

;; NS Character specific code

```
(DEFINEQ
```

(\CREATECHARSET

```
  [LAMBDA (CHARSET FONT NOSLUG?)                                    ; Edited 12-Jul-2022 14:37 by rmk
                                                                    ; Edited  8-May-93 23:42 by rmk:
                                                                    ; Edited  4-Dec-92 11:43 by jds
```

;; Creates and returns the CHARSETINFO for charset CHARSET in fontdesc FONT, installing it in fonts FONTCHARSETVECTOR

```
                                                                    ; NOSLUG?  means don't create an empty (slug) csinfo if the
                                                                    ; charset is not found, just return NIL
    (DECLARE (GLOBALVARS \DISPLAYSTREAMTYPES))
    (CL:WHEN (OR (ILESSP CHARSET 0)
                 (IGREATERP CHARSET \MAXCHARSET))
           (\ILLEGAL.ARG CHARSET))
    (LET [CSINFO (CREATEFN (COND
                             ((FMEMB (FONTPROP FONT 'DEVICE)
                                     \DISPLAYSTREAMTYPES)
                              (FUNCTION \CREATECHARSET.DISPLAY))
                             (T (CADR (ASSOC 'CREATECHARSET (CDR (ASSOC (FONTPROP FONT 'DEVICE)
                                                                        IMAGESTREAMTYPES]
```

;; Create a descriptor of info for that charset, and use it to fill things in.

```
        (CL:WHEN [SETQ CSINFO (APPLY CREATEFN (APPEND (FONTPROP FONT 'DEVICESPEC)
                                                      (LIST CHARSET FONT NOSLUG?]
                                                                    ; the create method did not return NIL--NOSLUG? must be T.
            (\INSTALLCHARSETINFO FONT CSINFO CHARSET))])
```

(\INSTALLCHARSETINFO

```
  [LAMBDA (FONT CSINFO CHARSET)                                     ; Edited 12-Jul-2022 15:08 by rmk
    (replace \SFAscent of FONT with (IMAX (fetch \SFAscent of FONT)
                                          (SIGNED (fetch CHARSETASCENT of CSINFO)
                                                  16)))
    (replace (FONTDESCRIPTOR \SFDescent) of FONT with (IMAX (fetch (FONTDESCRIPTOR \SFDescent) of FONT)
                                                            (SIGNED (fetch (CHARSETINFO CHARSETDESCENT)
                                                                           of CSINFO)
                                                                    16)))
                                                                    ; jtm: height = ascent + descent, not (IMAX fontHeight
                                                                    ; charSetHeight)
    (replace (FONTDESCRIPTOR \SFHeight) of FONT with (IPLUS (fetch (FONTDESCRIPTOR \SFAscent) of FONT)
                                                            (ffetch (FONTDESCRIPTOR \SFDescent) of FONT)))
    (\SETCHARSETINFO (fetch (FONTDESCRIPTOR FONTCHARSETVECTOR) of FONT)
           CHARSET CSINFO)
```

;; \AVGCHARWIDTH has to be confused after the CSINFO is stuck in.

```
    (replace (FONTDESCRIPTOR FONTAVGCHARWIDTH) of FONT with (\AVGCHARWIDTH FONT))
    CSINFO])
```

```
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS DISPLAYFONTCOERCIONS MISSINGDISPLAYFONTCOERCIONS MISSINGCHARSETDISPLAYFONTCOERCIONS CHARSETERRORFLG)
)

(RPAQ? DISPLAYFONTCOERCIONS NIL)

(RPAQ? MISSINGCHARSETDISPLAYFONTCOERCIONS
       '(((GACHA)
          (TERMINAL))
         ((MODERN)
          (CLASSIC))
         ((TIMESROMAN)
          (CLASSIC))
         ((HELVETICA)
          (MODERN))
         ((TERMINAL 6)
          (MODERN 6))
         ((TERMINAL 8)
          (MODERN 8))
         ((TERMINAL 10)
          (MODERN 10))
         ((TERMINAL 12)
          (MODERN 12))))

(RPAQ? MISSINGDISPLAYFONTCOERCIONS '(((GACHA)
                                      (TERMINAL))
                                     ((MODERN)
                                      (CLASSIC))
                                     ((TIMESROMAN)
                                      (CLASSIC))
                                     ((HELVETICA)
                                      (MODERN))
                                     ((TERMINAL)
                                      (MODERN))))

(RPAQ? CHARSETERRORFLG NIL)

(RPAQ? \DEFAULTCHARSET 0)

(DEFINEQ

(\FONTRESETCHARWIDTHS
  [LAMBDA (CSINFO FIRSTCHAR LASTCHAR)                    (* AJB " 6-Dec-85 14:42")
                                                         ; sets the widths array from the offsets array
    (PROG ((mincharcode FIRSTCHAR)
           (maxcharcode LASTCHAR)
           (offsets (fetch (CHARSETINFO OFFSETS) of CSINFO))
           (widths (fetch (CHARSETINFO WIDTHS) of CSINFO))
           left right charoffset dummycharoffset dummycharwidth)
          (SETQ dummycharoffset (\FGETOFFSET offsets (ADD1 maxcharcode)))
          (SETQ dummycharwidth (IDIFFERENCE (\FGETOFFSET offsets (IPLUS maxcharcode 2))
                                            dummycharoffset))
          [for charcode from 0 to \MAXCHAR do (COND
                                                ((OR (ILESSP charcode mincharcode)
                                                     (IGREATERP charcode maxcharcode))
                                                 (\FSETOFFSET offsets charcode dummycharoffset)
                                                 (\FSETWIDTH widths charcode dummycharwidth))
                                                (T (SETQ left (\FGETWIDTH offsets charcode))
                                                   (SETQ right (\FGETWIDTH offsets (ADD1 charcode)))
                                                   (COND
                                                      ((EQ left right)
                                                       (\FSETOFFSET offsets charcode dummycharoffset)
                                                       (\FSETWIDTH widths charcode dummycharwidth))
                                                      (T (\FSETWIDTH widths charcode (IDIFFERENCE right left]
          (\FSETWIDTH widths (ADD1 \MAXCHAR)
                 dummycharwidth)
          (\FSETOFFSET offsets (ADD1 \MAXCHAR)
                 dummycharoffset])

)

(DECLARE%: DONTEVAL@LOAD

(RPAQ? DISPLAYFONTEXTENSIONS 'DISPLAYFONT)

(RPAQ? DISPLAYFONTDIRECTORIES '({DSK}/USR/LOCAL/LDE/FONTS/DISPLAY/PRESENTATION/
                                {dsk}/usr/local/lde/fonts/display/publishing/))
)

(DECLARE%: EVAL@COMPILE

(PUTPROPS \FGETCHARIMAGEWIDTH MACRO (OPENLAMBDA (FONT CHARCODE)
                                      (\FGETWIDTH (ffetch (CHARSETINFO IMAGEWIDTHS) of (\GETCHARSETINFO
                                                                                         (\CHARSET CHARCODE)
```

```
                                                                                    FONT))
                                                    (\CHAR8CODE CHARCODE))))

(PROGN (PUTPROPS \GETFONTDESC DMACRO [X (COND
                                           ((CDR X)
                                            (CONS '\COERCEFONTDESC X))
                                           (T `(\DTEST ,(CAR X)
                                                       'FONTDESCRIPTOR])
       (PUTPROPS \GETFONTDESC MACRO (= . \COERCEFONTDESC)))

(PUTPROPS \SETCHARSETINFO MACRO ((CHARSETVECTOR CHARSET CSINFO)
                                 (\RPLPTR CHARSETVECTOR (UNFOLD CHARSET 2)
                                        CSINFO)))
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(LOCALVARS . T)
)

(PUTPROPS FONT FILETYPE :FAKE-COMPILE-FILE)

(DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS

(ADDTOVAR NLAMA )

(ADDTOVAR NLAML )

(ADDTOVAR LAMA FONTCOPY)
)

(PUTPROPS FONT COPYRIGHT ("Venue & Xerox Corporation" 1981 1982 1983 1984 1985 1986 1987 1988 1989 1990 1991
                                1992 1993 1994 1999 2021))
```

FUNCTION INDEX

VARIABLE INDEX

MACRO INDEX

RECORD INDEX

CONSTANT INDEX

PROPERTY INDEX

OPTIMIZER INDEX