```
-- AlFont.mesa  Edited by: Sandman, October 13, 1977  3:20 PM

DIRECTORY
  FontDefs: FROM "FontDefs",
  InlineDefs: FROM "InlineDefs",
  Mopcodes: FROM "Mopcodes",
  SystemDefs: FROM "SystemDefs",
  SegmentDefs: FROM "SegmentDefs";

DEFINITIONS FROM FontDefs;

AlFont: PROGRAM IMPORTS SegmentDefs, SystemDefs EXPORTS FontDefs =
  BEGIN

  FileSegmentHandle: TYPE = SegmentDefs.FileSegmentHandle;

  CR: CHARACTER = 15C;
  SP: CHARACTER = ' ;

  AlFontObject: TYPE = RECORD [
    procs: FontObject,
    seg: FileSegmentHandle,
    lockCount: CARDINAL,
    height: CARDINAL];

  AlFontHandle: TYPE = POINTER TO AlFontObject;

  FHptr: TYPE = POINTER TO FontHeader;
  Fptr: TYPE = POINTER TO Font;
  FCDptr: TYPE = POINTER TO FCD;
  FAptr: TYPE = POINTER TO FontArray;
  FontArray: TYPE = ARRAY [0..255] OF FCDptr;

  Font: TYPE = MACHINE DEPENDENT RECORD [
    header: FontHeader,
    FCDptrs: FontArray,          -- array of self-relative pointers to
      -- FCD's.  Indexed by char value.
      -- font pointer points hear!
    extFCDptrs: FontArray        --  array of self-relative pointers to
      -- FCD's for extentions.  As large an
      -- array as needed.
    ];

  FontHeader: TYPE = MACHINE DEPENDENT RECORD
    [
    maxHeight: CARDINAL,         -- height of tallest char in font (scan lines)
    variableWidth: BOOLEAN,      -- IF TRUE, proportionally spaced font
    blank:       [0..177B],      -- not used
    maxWidth: [0..377B] -- width of widest char in font (raster units).
    ];

  FCD: TYPE = MACHINE DEPENDENT RECORD [
    widthORext: [0..77777B],     -- width or extention index
    hasNoExtension: BOOLEAN,     -- TRUE=> no ext.;prevfield=width
    height: [0..377B],           -- # scan lines to skip for char
    displacement: [0..377B]      -- displacement back to char bitmap
    ];


  CharWidth: PUBLIC PROCEDURE [font: FontHandle, char: CHARACTER] RETURNS [w: CARDINAL] =
    BEGIN
    code: CARDINAL;
    cw: FCDptr;
    fontdesc: FAptr;
  -- checkfor control characters
    IF char = CR THEN char ← SP;
    IF char < SP THEN
      RETURN[CharWidth[font, '↑] +
             CharWidth[font,
                 LOOPHOLE[LOOPHOLE[char,CARDINAL]+100B,CHARACTER]]];
    w ← 0;
    fontdesc ← @LockFont[font].FCDptrs;
    code ← LOOPHOLE[char];
    DO
      cw ← LOOPHOLE[fontdesc[code]+LOOPHOLE[fontdesc,CARDINAL]+code];
      IF cw.hasNoExtension THEN EXIT;
      w ← w+16;
      code ← cw.widthORext;
      ENDLOOP;
    w ← w+cw.widthORext;
    UnlockFont[font];
    RETURN
    END;

  CharHeight: PUBLIC PROCEDURE [font: FontHandle, char: CHARACTER] RETURNS [CARDINAL] =
    BEGIN
    RETURN[LOOPHOLE[font,AlFontHandle].height]
    END;

  CONVERT: MACHINE CODE
    [char: CHARACTER, font: FAptr, destWord: POINTER,
    scanLineLength: CARDINAL, destBit: [0..15]]
    RETURNS [width: CARDINAL, newdestBit: [0..15], newdestWord: POINTER] =
      INLINE [Mopcodes.zCONVERT];

  PaintChar: PROCEDURE
    [font: FontHandle, char: CHARACTER, bmState: POINTER TO BitmapState] =
    -- note funny y-1 due to use of CONVERT!
    BEGIN OPEN bmState;
    dba: CARDINAL = InlineDefs.BITAND[InlineDefs.BITNOT[x], 17B];
    wad: POINTER = origin+(x/16)+(y-1)*wordsPerLine;
    pfont: FAptr = @LockFont[font].FCDptrs;
    cwidth: CARDINAL = CONVERT[char, pfont, wad, wordsPerLine, dba].width;
    UnlockFont[font];
    x ← x + cwidth;
    RETURN
    END;

  ClearChar: PROCEDURE
    [font: FontHandle, char: CHARACTER, bmState: POINTER TO BitmapState] =
    BEGIN OPEN bmState, InlineDefs;
    bit: [0..15];
    xword: CARDINAL;
    scanLines: CARDINAL = LOOPHOLE[font,AlFontHandle].height;
    start,p: POINTER;
    cwidth: INTEGER ← CharWidth[font,char];
    mask: WORD;
    ones: WORD = 177777B;
    IF x < cwidth THEN BEGIN cwidth ← x; x ← 0 END
    ELSE x ← x - cwidth;
    xword ← x/16; bit ← x MOD 16;
    mask ← BITOR[BITSHIFT[ones,16-bit],BITSHIFT[ones,-(bit+cwidth)]];
    start ← origin + xword + y*wordsPerLine-1;
    cwidth ← cwidth + bit;
    DO
      p ← start ← start + 1;
      THROUGH [0..scanLines) DO
        p↑ ← BITAND[p↑,mask];
        p ← p + wordsPerLine;
        ENDLOOP;
      IF (cwidth ← cwidth - 16) <= 0 THEN EXIT;
      mask ← BITSHIFT[ones,-cwidth];
      ENDLOOP;
    RETURN
    END;

  LockFont: PROCEDURE [font: FontHandle] RETURNS [Fptr] =
    BEGIN OPEN SegmentDefs, af: LOOPHOLE[font,AlFontHandle];
    IF (af.lockCount ← af.lockCount + 1) = 1 THEN SwapIn[af.seg];
    RETURN[FileSegmentAddress[af.seg]]
    END;

  UnlockFont: PROCEDURE [font: FontHandle] =
    BEGIN OPEN SegmentDefs, af: LOOPHOLE[font,AlFontHandle];
    IF (af.lockCount ← af.lockCount - 1) = 0 THEN Unlock[af.seg];
    RETURN
    END;

  DestroyFont: PROCEDURE [font: FontHandle] =
    BEGIN
    CloseFont[font];
    SystemDefs.FreeHeapNode[font];
    RETURN
    END;

  CloseFont: PROCEDURE [font: FontHandle] =
    BEGIN OPEN af: LOOPHOLE[font,AlFontHandle];
    IF af.seg.lock = 0 THEN SegmentDefs.SwapOut[af.seg];
    RETURN
    END;

  CreateFont: PUBLIC PROCEDURE
    [fontSegment: FileSegmentHandle] RETURNS [f: FontHandle] =
    BEGIN
    p: AlFontHandle = SystemDefs.AllocateHeapNode[SIZE[AlFontObject]];
    f ←  LOOPHOLE[p];
    p↑ ← [
      procs: [
        paintChar: PaintChar,
        clearChar: ClearChar,
        charWidth: CharWidth,
        charHeight: CharHeight,
        close: CloseFont,
        destroy: DestroyFont,
        lock: LockFont,
        unlock: UnlockFont],
      seg: fontSegment,
      lockCount: 0,
      height: LockFont[f].header.maxHeight];
    UnlockFont[f];
    RETURN
    END;

  END.
```