File formats:

GEM program file formats

Applications (.APP)

APP files are normal .EXE-format files. The GEM Programmer's Toolkit specifies that on loading, they should resize their memory allocation to free any memory they are not using.

Desk Accessories (.ACC)

ACC files are EXE-format files; they are entered at CS:0 rather than CS:IP. They are loaded as overlays rather than programs, so they do not have their own PSP or stack. No useful information is passed in the registers.

A .ACC file will be unloaded without warning, so it cannot allocate resources and free them in - for example - an atexit() routine. This means that DJGPP programs cannot be Desk Accessories.

Accessories are only allowed as much space as the size of their executable file.

VDI drivers (.SYS, .VGA, etc.)

VDI drivers are EXE-format files; they are entered at CS:0 rather than CS:IP. The design is that they should be small-model, with the code segment containing the code and any data common to all instances of the driver, and the data segment containing data unique to one instance.

The driver will be entered with:

- AX = Data segment to use, 0 for the one in the EXE file.
- DS:DX = address of VDI parameter block.

The calls the driver will receive will be very similar to the INT EF interface, except:

- Coordinates will always be in pixels.
- Function 1 should additionally return
  - CONTRL[7] = data segment of this instance of the driver
  - CONTRL[8] = length of the instance data
- Functions 119 and 120 should additionally return
  - CONTRL[7] = segment of linked font list
  - CONTRL[8] = segment of font workspace
  - CONTRL[9] = "font offset"

Font file format

This file format is the same as the "GDOS font" on the Atari; it is usually little-endian even on the otherwise big-endian Atari.

The file starts with a font header:

```
        DEFW    font id         ;Unique to a font; eg 14 for Roman, 2 for
                                ;Sans Serif
        DEFW    point size
        DEFS    32              ;Name, ASCII, 0-terminated
        DEFW    first char defined
        DEFW    last char defined
        DEFW    top,ascent,half,descent,bottom  ;Dimensions
        DEFW    max char width
        DEFW    max cell width
        DEFW    left offset     ;Amount character slants left when skewed
        DEFW    right offset    ;Amount character slants right
        DEFW    thicken         ;No. of pixels to thicken by
        DEFW    ul_size         ;Size of underline
        DEFW    lighten         ;AND with this mask when lightening
        DEFW    skew            ;Mask for skewing
        DEFW    flags           ;Bit 0: Default system font
                                ;Bit 1: Use horizontal offsets table
                                ;Bit 2: Font image is in byteswapped format
                                ;Bit 3: Font is monospaced
                                ;Bit 5: Extended font header
        DD      hoffs           ;Offset of horizontal offsets table from
                                ;start of file (if bit 1 of flags is set)
        DD      coffs           ;Offset of character offsets table from
                                ;start of file
        DD      bmps            ;Offset of bitmaps table from
                                ;start of file
        DEFW    width           ;Width of form
        DEFW    height          ;Height of form
        DD      0               ;Used by the VDI when the font has loaded, as
                                ;pointer to the next font in the linked list.
```

If there is an extended font header, this follows:

```
        DD      next            ;Offset of next section of this font
                                ;from start of file (eg, another character
                                ;range). The next section will have its
                                ;own font header.
        DD      0               ;Reference count when the font is loaded
        DD      offset_tbl      ;File offset of horizontal offset table
        DEFW    offset_len      ;Length of horizontal offset table
        DEFS    14              ;Reserved
        DEFW    dflags          ;Device flags
        DEFS    32              ;Escape sequence buffer
```

If there is a horizontal offsets table, this comes next. It contains two bytes for each character. The first is the number of pixels by which that letter should be moved to the left when it is displayed; the second is the number of pixels by which the next letter printed should be moved to the left. In other words, these two implement proportional spacing by making the letter narrower than the cell size in the header.

The character offsets table consists of one word for each character; this word is the X-coordinate of the glyph in question within the font.

The font itself is stored as a bitmapped image of all the characters side by side. If the image is in byteswapped format, each byte will appear to be swapped with its neighbour (as in a standard GEM device-independent bitmap).

Icon set (.ICN)

Note: This file contains "address" fields. To convert these into offsets within the file, subtract the "load address" value at bytes 2-3.

All values are little-endian.

```
2 bytes:        Address of the strings table; subtract the load address
                to get the file offset.
2 bytes:        Load address when the file was created.
```

There then follow 72 ICONBLK structures:

```
                4 bytes: Mask image number
                4 bytes: Icon image number
                4 bytes: -1  (Caption number; there is never a caption)
                1 byte:  0   (Letter to draw on the icon; there is never one)
                1 byte:  Icon colours. Bits 0-3 are the background colour
                         and bits 7-4 are the foreground.
                2 bytes: X position of drive letter in the icon
                2 bytes: Y position of drive letter in the icon
                2 bytes: X position of icon image relative to containing
                         rectangle. This is normally calculated as
                         (78 - image width) / 2.
                2 bytes: Y position of icon image. Always 0.
                2 bytes: Width of icon image.
                2 bytes: Height of icon image.
                         Normal icon sizes are 32x32 (EGA/VGA) and 48x24 (CGA).
                2 bytes: X position of icon caption. This is normally 0
                         for 32x32 icons and 4 for 48x24 icons.
                2 bytes: Y position of icon caption. This is usually the
                         same as the icon height.
                2 bytes: Width of icon caption. Usually 72.
                2 bytes: Height of icon caption. Usually 10.
```

After the icon table come the icon/mask bitmaps. There are up to 144 of these, and the file format assumes that they are all the same size. Each bitmap is formed of (icon_height) lines; each line is ((icon_width + 15)/16) bytes long.

The bitmap lines are formed of little-endian words rather than bytes, so in a 32x32 icon a line would go like this:

```
    byte 0: Pixels 16-23              (within each byte, bit 7 is the
    byte 1: Pixels 24-31               leftmost pixel and bit 0 is the
    byte 2: Pixels  0- 7               rightmost one).
    byte 3: Pixels  8-15
```

After the bitmaps are 32 0-terminated strings, describing 32 application types. The strings are followed by the strings table:

```
Strings table: 32 entries; each one is formed:
                2 bytes: Address of a string in the preceding section.
                        Subtract the "load address" field to get the file
                        offset.
```

The 72 icons described in the file are split up like this:

```
Icons 0-7: Disc drive types

Icon 0: Floppy drive
Icon 1: Hard drive
Icon 2: Folder
Icon 3: Trashcan (GEM 1.x and recent GPLed versions)
Icon 3 bitmap: Highlight image for selected drive  (GEM 2.x and later)
Icon 3 mask:   Highlight image for selected folder (GEM 2.x and later)
Icon 4: Network drive (ViewMAX and recent GPLed versions)
Icon 5: Other removable drive (recent GPLed versions)
Icon 6: CDROM (recent GPLed versions)
Icons  8-39: Application types
Icons 40-71: Document types, corresponding to the application types
```

The 32 strings in the string table are descriptions for each of these application types.

This text was originally created by John Elliott, and was located on his website www.seasip.info.
This version of the document was packaged by Shane M. Coughlan for the OpenGEM SDK.