Creating a Custom Raster Font

The information in this article applies to:
Microsoft Windows Software Development Kit for Windows version 3.0

Summary:

Fonts are stored as resources in resource-only dynamic-link libraries (DLLs). The process of creating a custom font library involves creating new font resources and inserting them into a DLL that has no code. Fonts must be in a resource-only library.
The Windows 3.0 Font Editor supports editing raster fonts compatible with Windows versions 2.x and Windows version 3.0. FONTEDIT cannot be used to edit vector fonts. (See Chapter 6 of the "Microsoft Windows Software Development Kit Tools" manual for more information.)

It is also possible to create a font-resource DLL that does have a code segment, which alleviates the problem of having to use a special linker.

Sample code for creating a font using the LINK4 linker is available in a file called FONTRES in the Software/Data Library. Sample code that demonstrates using the normal linker is also available in the Software/Data Library, in a file called MYFONT.
FONTRES and MYFONT can be found in the Software/Data Library by searching on the word FONTRES or MYFONT, the Q number of this article, or S13177 or S13181, respectively. FONTRES and MYFONT were archived using the PKware file-compression utility.

More Information:

Basic Steps (Overview)

I. Create font files using the Font Editor.
II. Create a font-resource script.
III. Create a dummy code module.
IV. Create a module-definition file that describes the fonts.
V. Compile and link the sources.

Note: Read Chapter 18 of the "Microsoft Windows Software Development Kit Guide to Programming." The following procedure is very similar.

1. Create a resource script (RC) file.
2. Add one FONT statement per font file created. For example:

```
MyFont1 FONT  MYFONT1.FNT
MyFont2 FONT  MYFONT2.FNT
```

Step I: Create a Font File

1. Invoke the Font Editor.
2. Open an existing font file (FNT).
3. Edit the cell arrays and attributes of the existing font.
4. Save the new font under a different name.

Note 1: It is not possible to generate a new font from scratch; an existing font file must be edited. Two fonts, ATRM1111.FNT and VGASYS.FNT, are supplied with the Windows 3.0 SDK to provide a font on which to base new fonts.
Note 2: The names of the font formats are deceiving. Windows 3.0-compatible format works only in 386 enhanced mode. Window 2.0-compatible format works in all modes; therefore, it is usually better to save fonts in 2.0 format.

Step II: Create a Font Resource Script

1. Create a resource script (RC) file.
2. Add one FONT statement per font file created. For example:

```
MyFont1 FONT  MYFONT1.FNT
MyFont2 FONT  MYFONT2.FNT
```

Step III: Create a Dummy Code Module

1. Write an assembly language procedure that generates no code.
2. Assemble the code to create an object file (OBJ). (This step may seem unnecessary but it is required; otherwise, the linker will complain because the linker will create an executable that does not have any object files. Creating the dummy code module with its null code segment forces the linker to create the required executable DLL).

The code for the dummy code segment might resemble the following:

```
        .xlist
include cmacros.inc
.list
sBegin CODE
sEnd   CODE
end
```

Step IV: Create a Module Definition File

1. Add a LIBRARY statement with the font resource title.
2. Add a DESCRIPTION statement that indicates the font characteristics.
3. Add a STUB statement in case the library is invoked from DOS.
4. Add a DATA statement with the NONE attribute.

The DEF file for a font library might resemble the following:

```
LIBRARY FONTLIB
```

```
DESCRIPTION 'FONTRES 133, 96, 72: MyFont, Terminal (7 point)'
STUB 'WINSTUB.EXE'
DATA NONE
```

Note: The DESCRIPTION statement specifies a string that describes the font attributes, and supplies a comment that is displayed by the Windows Control Panel when the font is loaded.

WINSTUB.EXE is a small file that prints the message "This application requires Microsoft Windows" if the user tries to run the application under DOS.
The NONE attribute indicates that the library does not require its own automatic data segment.

The description string MUST begin with the FONTRES text so that Windows will know that this is a font resource library.
(See the "Microsoft Windows Software Development Kit Guide to Programming" for more information and examples.)

Step V: Building the Font Resource Library

1. Use MASM to assemble the dummy code into an object file. 2. Use LINK4 to generate the library body.
3. Use RC to insert the font into the library.
4. Rename the font library to have the FON extension.

The following is an sample makefile:

```
all: fontlib.exe
fontlib.obj: fontlib.asm
masm fontlib.asm;
fontlib.exe: fontlib.mak fontlib.def fontlib.obj \
fontlib.rc fontlib.fnt
link4 fontlib.obj, fontlib.exe, NUL, /NOD, fontlib.def rc fontlib.rc
rename fontlib.exe fontlib.fon
```

Using LINK Instead of LINK4:

Important note: The specification of LINK4 in the sample above is not an error. The standard linkers supplied with Microsoft C version 5.1 and Microsoft C version 6.0 produce error messages when an attempt is made to create an executable file that has no segments. LINK4.EXE is not shipped with the Windows 3.0 SDK. However, it is shipped with the Windows 2.x SDK and with the Windows 3.0 DDK.

If Steps III, IV, and V of the procedure given above are modified as follows, LINK versions 5.12 and later can be used to create font files:

NEW Step III: Create a Dummy Code Module

Create a code segment in the dummy code module by creating an empty Windows

Exit Procedure (WEP). This code might resemble the following:

```
.xlist
include cmacros.inc
.list
sBegin CODE
cProc WEP,<FAR,PASCAL,PUBLIC>,<si,di>
        parmW EntryCode
cBegin WEP
cEnd WEP
cEnd CODE
end
```

NEW Step IV: Create a Module Definition File

Modify the DEF file provided above to add the following lines:

```
EXETYPE   WINDOWS
CODE      MOVEABLE DISCARDABLE
EXPORTS   WEP @1 RESIDENTNAME
```

NEW Step V: Building the Font Resource Library

Modify the makefile to refer to LINK instead of to LINK4.

Using MASM 6.0 Instead of MASM 5.1

If the font file is built using version 6.0 of the Microsoft Macro Assembler (MASM), use version 5.3 of the CMACROS.INC file included with MASM instead of version 5.2 of the file included with the Windows SDK.

To access the fonts, use AddFontResource() with the DLL name, and RemoveFontResource(). Use CreateFont() or CreateFontIndirect() to retrieve a handle to a font with the specified attributes. Use SelectObject() to put the font into a specified DC.
The face name of the font (for example, "System" or "Helv") can be specified when the font is created using the Font Editor. This same face name is specified as the lpFaceName parameter when calling CreateFont() or CreateFontIndirect(). The face name can be any name desired.

End.