Adobe Type Manager Version 2.0 Information

CREATED: 31-MAR-1992 MODIFIED: 01-APR-1992
3.10 WINDOWS

Summary:

Application name: Adobe Type Manager version 2.0
Application type: Font utility
Manufacturer: Adobe Systems, Inc.
RAM required: 1MB
Disk space needed: 950K
Changes to MS-DOS files: n/a
Changes to Windows files:

SYSTEM.INI:
[boot]
system.drv=atmsys.drv
atm.system.drv=system.drv
WIN.INI:
[PostScript,LPT1]
feed1=1
feed15=1
ATM=placeholder
softfonts=10
softfont1=d:[path]\atm2\pfm\bs_____.pfm softfont2=d:[path]\atm2\pfm\ca_____.pfm softfont3=d:[path]\atm2\pfm\dc_____.pfm softfont4=d:[path]\atm2\pfm\fgc_____.pfm softfont5=d:[path]\atm2\pfm\lg_____.pfm softfont6=d:[path]\atm2\pfm\shal____.pfm softfont7=d:[path]\atm2\pfm\mtr_____.pfm softfont8=d:[path]\atm2\pfm\mtb_____.pfm softfont9=d:[path]\atm2\pfm\mtbi____.pfm softfont10=d:[path]\atm2\pfm\mti_____.pfm

Files copied to Windows root directory:

ATM.INI
ATMCNTRL.EXE

Files copied to WINDOWS\SYSTEM directory:

ATM16.DLL
ATM32.DLL
ATMSYS.DRV

Things that are helpful to know about the application:

For Hewlett-Packard (HP) LaserJet printers with 512K, the installation gives the option of installing soft fonts. If you have problems displaying the fonts correctly, make sure the SYSTEM.INI is correct and ATM is turned on. To turn on ATM, run ATMCNTRL, click the ON button, then restart Windows.
Additional reference word(s): 3.10

End.

RLE Sample Application
Multimedia Systems Group

If you did not get this from Microsoft Sources, then it may not be the most current version. This sample code in particular will be updated and include more documentation.
Sources are:

The MM Sys BBS:	The phone number is 206 936-4082.
CompuServe:	WINSDK forum, MDK section.
Anonymous FTP from:	ftp.uu.net vendors\microsoft\multimedia\samples
Version 1.2	Released 02/28/92

This sample ONLY works on the following:

- a 386 or better
- Windows 3.1 (or Windows 3.0 + MME 1.0 + COMMDLG)

File	Description
df.asm	RLE routines including delta-frame. Very usefull. This file is the heart of this sample code.
df.h	Include file for df.asm
dib.c	General DIB handling routines.

WARNING: there are many different versions of this file contained in the sample code.

dib.h	Include file for dib.c
gmem.h	Protect mode global memory optimizations
makefile	The infamous makefile
mem.asm	General memory routines
rle.c	RLE and Delta-frame interfaces and general routines
rle.h	Include file for rle.c
rlea.asm	386 version of some routines in rle.c
rleapp.c	The main functions/ where the app lives
rleapp.def	Our app's DEF
rleapp.dlg	Our app's dialogs
rleapp.h	Our app's main include file
rleapp.ico	Our app's icon
rleapp.rc	Our app's resource file
rledlg.c	Dialog handling routines
rledlg.h	Include for rledlg.c
rlefile.c	Animation file I/O

Notes about this sample

This sample app is called rle.exe even though the ZIP file is RLEAPP.ZIP.
The application can be used to load, generate, play and save animations using the DIB RLE

format for delta-frames.

The RLE format is described in the Windows 3.0 SDK docs in the section on the BITMAPINFOHEADER structure. It is described in the Multimedia SDK (MDK) in the Programmer's Reference in the section entitled "Windows 3.0 Bitmap Compression Formats".

This app only handles 8bpp DIBS, even though the same technology applies to 4bpp DIBs (RLE, delta-frame, etc).

Sample animations can be obtained from the MM System's BBS at 206 936-4082 in the VIDEOS library, or look where you obtained this sample.
This sample code includes many optimizations for 386-specific machines. This is a very usefull way to increase speed because the MPC spec calls for a 386 as a minimum, so in most cases, the 286 support can be left out.

This program is very dependant on blit speed of the video card and driver. In most cases, the video driver is very sssllloooowww doing blits. Complain to your card manufacturer if you think it should be faster.

You should be able to blit 160x120 256 color frames (with the same palette) at at least 15 frames/second. Bad drivers/card will be able to do about 1 frame/second.

I am planning on releasing an AutodeskR file converter to RLE and RL0 format converter soon. The capability to read these files will be built in to future versions of this sample code.

If you have any comments/questions/requests about this sample, you can reach me at the BBS, or by FAX at 206 93MSFAX.

Please let me know if this sample code is usefull to you.
Matt Saettler
76150,2523

End.

Animation in Windows

By: Herman Rodent, Nell, and Olivia the Wonder Dog

Abstract

This article is aimed at people who would like to create a Windows application which does some form of animation or who would like to understand how to improve the performance of an existing application. The main focus of the article is on using DIBs for the images and the DIB driver (DIB.DRV) for the off-screen image buffer. The article is written around a sample application (SPRITES) which is included with the article. Some knowledge of animation techniques is assumed. If you're looking for a "How to do animation" article this isn't it. The following points are covered:

- Using DIBs
- Using the DIB driver
- Palettes
- Measuring and Improving Performance
- Lots of Useful Little Tips and Hints

Introduction

A common misconception is that Windows is too slow to do any sort of animation effectively. This concept has hindered the porting of many good MS-DOS applications (notably games) to Windows. In addition to the theory that Windows is too slow, many MS-DOS programmers fear having to learn how to program in the Windows environment and recreate an existing application from scratch using strange new function calls and techniques. To top this off, there are a number of existing applications which while adequate at what they do, do not show what the system is capable of.

The Video for Windows technology recently released to developers shows what really can be achieved - reasonable quality video in a window. The frame rates are not excessive and the window sizes are not huge but the net effect is very impressive and not difficult to achieve.

If you are considering porting an MS-DOS application to Windows or would like to add some animation to an existing Windows application, then this note and its accompanying sample code should help you along the way.

For myself, I do not even pretend to be an animation wizard. Before I wrote this article I thought I had some pretty good ideas about how all this should be done and a few of them were even right! By experimenting with my own ideas and those of others I have discovered a lot of useful tips which I thought were worth passing on. Much of the information about palettes is reproduced form Ron Gery's articles available elsewhere on the Developer Network CD. Huge volumes of information also came from Todd Laney, without whom frame animation in Windows might never have got beyond one frame per second.

Architecture

When I decided to write the SPRITES sample application, I made two major choices which determined the architecture of the entire application and hence this article. These were to use DIBs for all the images and to use the DIB driver for the off-screen image buffer. Since these things are so fundamental to what follows some discussion of these two decisions is in order. This section also includes some notes on the use of palettes.

Using DIBs

Before the advent of Device Independent Bitmaps (DIBs) in Windows 3.0, we had to create applications with different DDBs (Device Dependent Bitmaps) for every screen resolution/color depth we wanted the application to run on. DIBs provide a way around that problem by packaging together information about the image size, a color table and the bits themselves into a single file. DIBs are inherently portable between different Windows environments. The only problem being that there are no Windows functions to render them directly which makes them seem somewhat difficult to use. I chose to use them in my sample application because of the portability they provide. It turns out that the code required to manipulate them is not complicated and the difficulty in writing the code is easily outweighed by the advantages of using DIBs.

DIB File Formats

One irritating complexity of the DIB file format is that there are two different versions of it. One was designed for Windows 3.0 and the other for the Presentation Manger in OS/2. The two formats essentially do the same thing but in slightly different ways.

The Windows format uses four bytes for each color table entry which keeps the color table elements DWORD aligned and is efficient to access on a 32 bit platform. The Presentation Manager format uses three bytes for each color table entry resulting in a smaller color table (big deal compared with the image size!) but inefficient access.

Both formats exist in applications designed for Windows. The Windows system handles both formats internally but may not handle the Presentation Manager format in the future. A good application which reads DIB files should be able to handle both, the only complexity being creating a palette from the color table. More about recognizing which format a DIB is in and dealing with it later.

The other irritation of the DIB format which is a hangover from the Presentation Manager design is that the scan lines in the DIB are 'upside down' with respect to their address in the file. In other words the bits for the last scan line are first and the bits for the first scan line are last. This requires some mental juggling when you manipulate or blt the bits in your code. Figure 1 shows the problem.

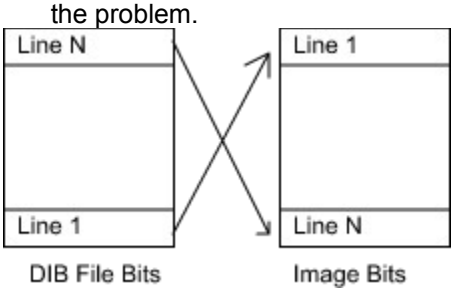

Figure 1. DIB scan lines are inverted.

The DIB file structure consists of two blocks: a header and the bits. The header is actually three

structures packed together so the overall picture looks like figure 2.

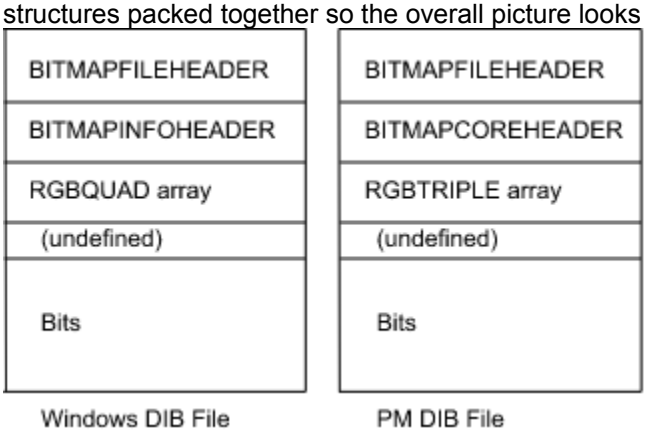

Figure 2. Windows and Presentation Manager DIB File Formats

Note: In the case of the Windows format the BITMAPINFOHEADER and RGBQUAD array are both contained within a BITMAPINFO structure. In the case of the PM format file, the BITMAPCOREHEADER and RGBTRIPLE array are contained within a BITMAPCOREINFO structure. All of these structures are documented in the Windows SDK.

DIBs in memory

If you examine the file format structures carefully, you will see that the file header contains a pointer to the position of the image bits within the file implying that the block containing the bits need not be contiguous with the header. This is a convenience for the file format but a nuisance for a memory image format since we don't want to have to manage two memory blocks for the DIB and we don't want to allocate memory which doesn't contain any useful information. So when we read a DIB file into memory we calculate the total memory required for the header (excluding the BITMAPFILEHEADER which is not required) and the bits, allocate a single block for the whole thing and read the header and bits separately into this block so that they end up contiguous within it.

But what about the two different formats? Instead of dealing with both formats in the application, I chose always to convert Presentation Manager DIBs to Windows DIB format internally giving only one format to deal with. This is easily done when the DIB image is created in memory. This does mean that a DIB which is subsequently saved to file later may end up being converted from Presentation Manager format to Windows format but this is not a problem for any Windows application :).

A Windows format DIB which is contained in a single memory block with its header immediately preceding the bits is called a packed DIB and is used to transfer DIBs through the Clipboard. Its Clipboard format name is CF_DIB.

Handles or Pointers?

Since we will allocate a block of memory for the packed DIB information, should we retain the handle to the memory or a pointer to it? In days of old when knights were bold and memory wasn't addressed by descriptor tables (real mode) we always worked with handles. Since we now live in a protected mode environment and are rapidly approaching the happy days of 32 bit, flat model Windows programming I have chosen to keep a pointer to all my global objects rather than a handle. This avoids thousands of unnecessary calls to GlobalLock and GlobalUnlock which both simplifies and speeds up the code. To make the code as portable as possible I use macros (ALLOCATE() and FREE()) to allocate and free memory blocks.

When I was writing the sample application I found that I needed to use things like the height and width of the DIB a lot so I initially created a structure which held all the information about the DIB I used a lot and a pointer to the packed DIB itself. This wasn't really as efficient as it might have been since we were back to having two memory blocks to describe one DIB. I decided to implement a set of macros to do all the pointer dereferencing (DIB_HEIGHT etc.) and these were used throughout the application. There is some issue here regarding the code speed. It could be argued that all the pointer dereferencing used in the macros leads to slower code than could be achieved by having the commonly used DIB parameters cached in a single place. The macros helped make the code a bit simpler to look at and resulted in a DIB being just a single memory block in CF_DIB format which on balance I prefer. Figure 3 shows the format of a packed DIB in memory.

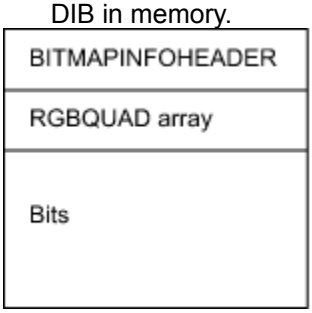

Figure 3. Packed DIB memory organization

General DIB Notes

When accessing the bits of a DIB in memory be very careful that you use the correct type of pointer. If the DIB is less than 64k in size, you can use a FAR pointer to access all of it. If the DIB is larger than 64k then it's very important that you use a HUGE pointer so that the address arithmetic will be performed correctly. Bitmaps of 100x640, 200x320 or 250x250 are all just under 64k.

FAR pointers only have 64k offsets so if you attempt to go beyond 64k the offset will wrap and the address will have a low offset value rather than the big one you expect. This problem is most noticeable when writing to the DIB memory since an address wrap while writing the DIB bits will cause the BITMAPINFO header to be overwritten and consequently the integrity of the DIB trashed. I know about this - I've done it. The code always looks just fine , but something keeps trashing the header. Be warned!

When accessing the bits in a DIB, also be aware that the width of a scan line is always divisible by 4 (DWORD aligned) and consequently the pixel position is not simply determined by the scan line number and image width. The SPRITES header file GLOBAL.H includes a set of macros for accessing information on a DIB given a pointer to it. The DIB_STORAGEWIDTH macro returns the physical length of the scan line. The DIB_WIDTH macro returns the width of the image.

The DIB Driver

The DIB driver was developed in response to very many requests from software developers to be able to manipulate the bits of a bitmap directly in memory. The DIB driver was originally developed as a part of the Multimedia Extensions to Windows and subsequently shipped as a standard component of Windows 3.1.

Direct Bitmap Memory Access

It is not possible to directly access the bits of a Windows DDB. This is not simply because we (Microsoft) won't tell you how to find it, it's because allocation of the bitmap memory is done by the video device driver and the driver can choose to use any memory available to it including memory on the video adapter card. In the case of the 8514 driver, there is quite a bit of spare video memory in some video modes and it makes good sense for the video driver to make use of it if it can. If the memory for a bitmap is allocated on the adapter card then there is no direct access to it possible since that memory is not mapped into the processor address space.

In addition to the problem of where the memory is located, there is an additional problem that the format of the memory allocated for the bitmap is also defined by the device driver. This makes a lot of sense since it allows the driver to choose a memory organization which makes it easy to transfer (blt) chunks to and from the video memory. Since this format can vary from driver to driver, even if you could access the memory, there is no way you could know how it was organized.

The DIB driver solves the problem of memory access and bitmap organization by requiring the application to allocate the memory for the bits and provide the information on how the bitmap bits are organized. The application does this by creating a packed DIB in memory and passing the address of this structure to the DIB driver when it (the application) requests the creation of a DC (Device Context). The packed DIB structure contains a BITMAPINFO structure at its start which gives the width and height of the bitmap and also describes its color table and pixel color organization. Directly following the header are the bitmap bits, organized the same way as the bitmap bits in a DIB file.

What the DIB Driver Can do for You

By using the DIB driver to create a DC in a piece of memory you have allocated and understand the organization of, you can choose to create images in that DC in two ways: use GDI operations in the same way as for any other DC or manipulate the bits directly in memory.

For an animation application this means that you can use the DIB Driver to manage the off-screen image buffer. You create a packed DIB the same size and organization as the window in which the animation will play and then use the DIB Driver to create a DC for it. Thereafter you can do all your image rendering to the off-screen DIB and use the StretchDIBits function to move areas of the off-screen DIB to the display window DC.

Drawing to the DIB Driver DC

As mentioned above, you can perform regular GDI operations on a DIB Driver DC. Since the DIB Driver is a non-palletized device there is no point in selecting/realizing a palette in the DC. There is no harm in doing this however as the driver simply ignores the request. Generic code which selects and realizes a palette when drawing to an arbitrary DC will still work OK if used on a DIB Driver DC.

DIB Blt Functions

There is no function to directly blt a packed DIB to a DC but there are a small number of functions designed to move DIB bits between a DIB and a DDB or the screen DC. Table 1 lists the functions.

Name	Description

SetDIBits	Move DIB bits to a DDB
GetDIBits	Move DDB bits to a DIB
SetDIBitsToDevice	Move DIB bits to a device
StretchDIBits	Move DIB bits to a device and optionally stretch them.

Table 1. DIB Functions

The SetDIBits and GetDIBits functions are directly implemented by screen and printer device drivers. The SetDIBitsToDevice function is a hangover from the Windows 3.0 DIB development and should not be used in new projects. Use StretchDIBits instead of SetDIBitsToDevice. StretchDIBits is the 'do all' function used to transfer DIB images between DCs. It is optionally implemented by the screen device driver. If not implemented in the device driver, GDI uses combinations of SetDIBits, GetDIBits and BitBlt to emulate it. Having GDI emulate StretchDIBits means awful performance. More about performance issues later.

StretchDIBits

Understanding what is going on with the color table entries when StretchDIBits is called can be quite a problem. The following set of figures show what happens when StretchDIBIts is used to draw DIBs to DC's of both palletized and non-paletized devices. The source DIB color table can be either RGB values (the DIB_RGB_COLORS flag is used) or palette index values (the DIB_PAL_COLORS flag is used). The four possible combinations are explained here.

Figure 4. Using StretchDIBits with DIB_RGB_COLORS to draw on a paletized device DC.

When drawing RGB values to a paletized device, GDI translates each RGB value to an index in the current physical palette by calling an internal version of GetNearestPaletteIndex. The set of indices is then passed to the device driver. The driver tests the index set to see if it is an identity palette. An identity palette is one which has an exact 1:1 mapping of logical palette index values to the current physical palette. In other words the palette table contains the values 0, 1, 2, ... 255. If the driver detects an identity palette then the DIB bits can be moved directly to the screen memory without translation. If the palette is not an identity palette then each pixel value (palette index) in the source DIB has to be translated through the lookup table supplied by GDI to the correct physical palette index value.

Figure 5. Using StretchDIBits with DIB_PAL_COLORS to draw on a paletized device DC.

When drawing palette index values to a paletized device, GDI builds a translation table to map the logical index values in the DIB to the current physical palette index values. The set of indices and the translation table are then passed to the device driver. If the palette is found to be an identity palette no translation is performed. (See the previous example).

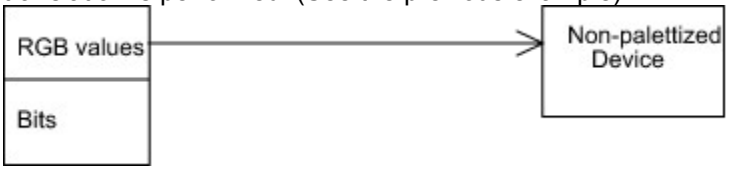

Figure 6. Using StretchDIBits with DIB_RGB_COLORS to draw on a non-paletized device DC.

When drawing RGB values to a non-paletized device, no translation is done by GDI. The driver gets the RGB values directly. In the case of the DIB driver and an 8 bit DIB, these RGB values are converted to 8 bit pixel values by looking them up in the (DIB Driver) DIB color table.

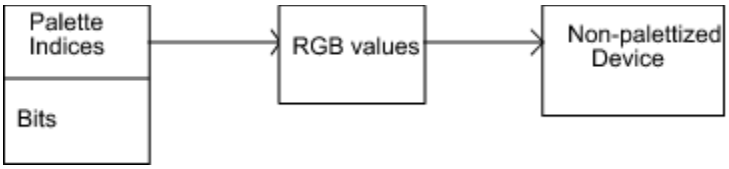

Figure 7. Using StretchDIBits with DIB_PAL_COLORS to draw on a non-paletized device DC.

When drawing palette index values to a non-paletized device, GDI looks up each index in the currently selected logical palette and translates it to an RGB value. The table of RGB values is passed to the device driver.

Palettes

Most of the popular display cards today provide 640 by 480 with 256 colors and Windows provides palette management for the available colors. Any application wishing to use these colors will need to create and use one or more palettes. If you are not familiar with how palettes work in Windows, please refer to Ron Gery's articles: "The Palette Manager: How and Why" and "Using DIBs with Palettes" elsewhere on the Developer Network CD.

In designing an animation application, we must consider how color will be used and create one or more palettes accordingly.

Palette Operations Take Time

Each time an application selects a palette for use by itself, Windows creates a mapping between the colors in the requested (logical) palette and the system (physical) palette. Exactly how it does this and the rules governing it are covered in Ron's articles. The important point is that the process of mapping a logical palette to the system palette takes a finite time and is not something we want to be doing every time we draw an image. In fact, GDI attempts to be helpful here and recognizes when the palette being realized in the current foreground application is the same as the one realized before it, and doesn't repeat the matching operation needlessly. None the less, we need to be careful about our use of palettes or the performance of our application will suffer.

One Palette Fits All

For the SPRITES application I chose to use only one palette. Rather than build the palette into the application I chose to always create the palette from the color table contained in the DIB used for the background scene. This was purely a convenience for me. You could just as easily read a palette in from any DIB file and use that one.

If all the images are to look reasonable when rendered with one common palette then the choice of the colors in that palette is obviously very important. How you should go about choosing those colors is beyond the scope of this article. I chose mine the cowards way, by letting the scanning software chose it for me. It so happens that the scanner I used created a common palette to save all of the images I scanned. It's not a great palette but it's OK for the purpose of demonstrating the principle. The scanner used was a Hewlett Packard ScanJet IIc and the software was Hewlett Packard's DeskScan II version 1.5.

If you have several images and want to play with their palettes, try running the BitEdit and PalEdit tools available in the Video for Windows SDK. If you just want to experiment, try looking at the images in the SPRITES application.

Comment: Add a button to run BitEdit here.

Creating a Palette

Creating a palette from the color table of a DIB is reasonably straight forward once you know how many colors you want to use. A LOGPALETTE structure is created large enough for the number of colors, the color information is copied from the DIB header and a call made to CreatePalette. The LOGPALETTE structure is then freed. Since this is a common requirement when dealing with DIBs, I wrote a function to create a palette directly from the BITMAPINFOHEADER of a DIB. This function is called CreateDIBPalette and is in the palette.c module of the SPRITES sample code. Here it is with the comments and some of the error handling code removed:

```
HPALETTE CreateDIBPalette(LPBITMAPINFO lpBmpInfo)
{
    LPBITMAPINFOHEADER lpBmpInfoHdr;
    HANDLE hPalMem;
    LOGPALETTE *pPal;
    HPALETTE hPal;
    LPRGBQUAD lpRGB;
    int iColors, i;

    lpBmpInfoHdr = (LPBITMAPINFOHEADER) lpBmpInfo;
    if (!IsWinDIB(lpBmpInfoHdr)) return NULL;

    lpRGB = (LPRGBQUAD)((LPSTR)lpBmpInfoHdr + (WORD)lpBmpInfoHdr-
>biSize);
    iColors = NumDIBColorEntries(lpBmpInfo);
    if (!iColors) return NULL;

    hPalMem = LocalAlloc(LMEM_MOVEABLE,
                         sizeof(LOGPALETTE) + iColors *
sizeof(PALETTEENTRY));
    if (!hPalMem) return NULL;
    pPal = (LOGPALETTE *) LocalLock(hPalMem);
    pPal->palVersion = 0x300; // Windows 3.0
    pPal->palNumEntries = iColors; // table size
    for (i=0; i<iColors; i++) {
        pPal->palPalEntry[i].peRed = lpRGB[i].rgbRed;
        pPal->palPalEntry[i].peGreen = lpRGB[i].rgbGreen;
        pPal->palPalEntry[i].peBlue = lpRGB[i].rgbBlue;
        pPal->palPalEntry[i].peFlags = 0;
    }

    hPal = CreatePalette(pPal);
    LocalUnlock(hPalMem);
    LocalFree(hPalMem);

    return hPal;
}
```

The CreateDIBPalette function uses NumDIBColorEntries to get the number of colors in the DIB color table. NumDIBColorEntries copes with the somewhat obscure rules governing exactly how many color entries there are in a DIB color table. The structure contains a **biClrUsed** field which should have the number of colors in it but often is set to zero meaning that the color table is the

maximum size appropriate for the color depth.

All of the DIB management code in the examples assumes that the DIB is in Windows format (having been converted from PM format if required when it was loaded). The helper function **IsWinDIB** is used to make the test. More about the DIB support functions later.

The SPRITES Sample Application

I decided to make my sample some form of sprite (cast based) animation for several reasons. Firstly it's much more challenging than frame animation and I'm a sucker for punishment. Secondly, all the techniques required for frame animation are used in sprite animation so an example of sprite animation code also effectively provides a frame example. Of course, the real reason is that I could scan some images to make the sprites which meant less artwork!

An Overview

The SPRITES application uses a DIB for a background scene and allows the loading of multiple sprites on top of the background scene. Each sprite has x,y and z coordinates and optional x and y velocity. It also has a flag to say if it can be dragged by the mouse or not.

A background and set of sprites can be combined into a scene described in a simple INI file. The entire scene can be loaded using the 'Load Scene' item from the 'File' menu.

The application updates the positions of all sprites which have a non-zero velocity as fast as it can using a PeekMessage loop.

Sprites which have the selectable attribute set can be dragged with the mouse. Double clicking on a sprite brings up a dialog which allows all the sprite attributes to be set.

The Z order value of zero is the front most position and values greater than zero go towards the back. I used a maximum value of 100 but the limit is actually 65535.

A separate debug information window works in conjunction with 'dprintf' statements in the code and the 'Debug' menu to show what's going on.

Figure 8 shows how the background DIB, the DIB Driver DC, common palette and screen DC relate to each other.

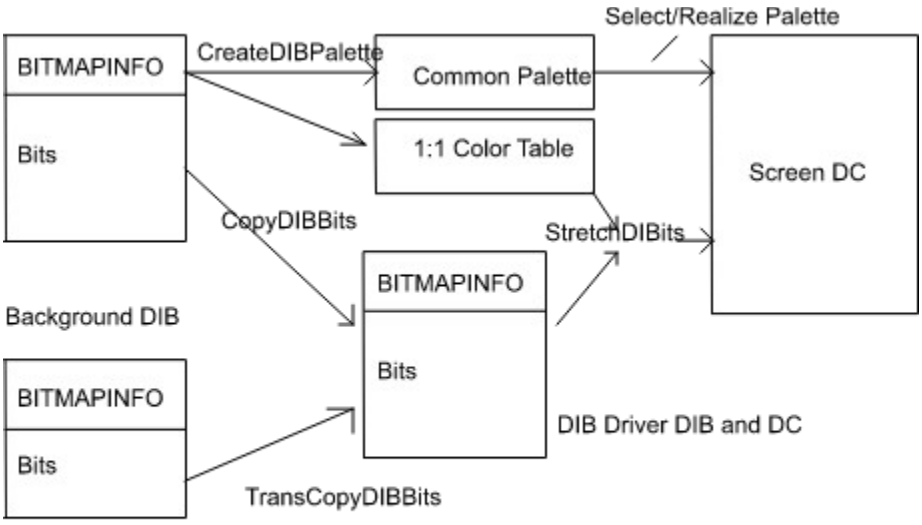

Figure 8. The architecture of the SPRITES application.

Code Notes

The SPRITES sample code has some features which I will mention here to avoid confusion when you read the code.

There are several 'dprintf' statements throughout the code. These are used to print debugging information in the debug window. The amount of information is controlled by the current debug level which can be set from the 'Debug' menu. I try to use level 1 for error messages (most important), level 2 for general procedural steps (like entering a major function), level 3 for increased procedural detail (what's happening in the function) and level 4 for data dumps and things that go on too fast to want to have data about them all the time. The dprintf statements are implemented as macros in GLOBAL.H.

Since the world of Windows programming is moving rapidly towards 32 bits, I try not to include the near or far attribute in pointer names. So instead of npDIB, lpDIB or fpDIB I simply use pDIB. This might seem confusing since it is so common to see lpSomething in Windows code but I believe that it will help in the long run. Almost all pointers in the code are actually FAR pointers. There are also a few HUGE pointers for dealing with big objects and one or two near pointers to data in the local heap where this made a significant difference to the performance.

In many cases when code fragments are included in the text of the article, I have removed comments, debug code and sometimes error reporting statements to help clarify what I'm talking about. Please look at the actual code in the sample before writing your own.

Be aware that last minute changes to the code before publication might mean some slight differences from what's in the article and what's in the sample code. If in doubt, go with the code in the sample.

The Background

The background scene of the animation is a single DIB. The color table found in the background DIB is used to create the palette used to render all the images to the window DC. A background is loaded by using the 'Load Background' item from the 'File' menu or by loading a scene by using the 'Load Scene' item. The LoadBackground function in BKGND.C is responsible for doing the work. We'll look at the code for each step of the function with a brief description of what is happening at each stage.

```
DeleteSpriteList();
```

The current set of sprites is deleted. This isn't really necessary, but it greatly simplified the set of dependencies. The background provides the common palette and each sprite has its color table adjusted to fit the background color set so it was just easier to start again each time a background was loaded.

```
DeleteDIB(pdibBkGnd);
```

The existing background DIB is deleted. The DeleteDIB function does nothing if the DIB does not currently exist.

```
pdibBkGnd = LoadDIB(pszPath);
```

The new background DIB is loaded. If no path was provided for the background DIB, a dialog is presented to choose it.

```
if (hpalCurrent) DeleteObject(hpalCurrent);
hpalCurrent = CreateDIBPalette(pdibBkGnd);
```

Any current palette (from a previous background DIB) is deleted and a new one created from the new background DIB. The CreateDIBPalette function is in the DIB.C module.

It is at this point that the palette could be modified to ensure that the first 10 and last 10 entries exactly match the system color entries. This is important to do so that the palette indicies will not need translation when they are blted to the screen. (This is discussed more elsewhere).

The window rectangle is then adjusted to fit the new background. I'll skip the code for that since it's commonplace.

```
if (hdcOffScreen) {
    DeleteDC(hdcOffScreen);
    hdcOffScreen = NULL;
}
DeleteDIB(pdibOffScreen);
```

Any existing off-screen DC and its associated DIB are deleted. The DeleteDIB function is in DIB.C.

```
pdibOffScreen = CreateCompatibleDIB(pdibBkGnd);
```

A new off-screen DIB is created the same size as the background DIB. The CreateCompatibleDIB function is in DIB.C.

```
hdcOffScreen = CreateDC("DIB", NULL, NULL, (LPSTR)pdibOffScreen);
```

A new off-screen DC is created using the DIB Driver and the new off-screen DIB. Note that the DIB Driver requires that the last argument be a pointer to a packed DIB structure.

```
    if (!pPalClrTable) {
        pPalClrTable = (LPBITMAPINFO) ALLOCATE(sizeof(BITMAPINFOHEADER)
                                               + 256 * sizeof(WORD));
    }

    _fmemcpy(pPalClrTable,
             pdibOffScreen,
             sizeof(BITMAPINFOHEADER));

    pIndex = (LPWORD)((LPSTR)pPalClrTable + sizeof(BITMAPINFOHEADER));
    for (i=0; i<256; i++) {
        *pIndex++ = (WORD) i;
    }
```

A 1:1 color lookup table is required when StretchDIBits is used later to copy image data from the off-screen DC to the Window DC. If the table doesn't already exist, the memory is allocated for it. The table header is copied from the background DIB. This sets the size information to be the same as the background and off-screen DIBs. Lastly, the color table is filled with the values 0 through 255 which gives the 1:1 color index mapping we will need later.

```
    Redraw(NULL, bUpdateScreen);
```

A call is made to render the background image to the off-screen DC and to update the window DC with the new image.

The Sprites

Each sprite consists of a DIB which provides the image and a set of variables which describe its size, position and optionally its velocity. The information about each sprite is contained in a SPRITE structure:

```
typedef struct _SPRITE {
    struct _SPRITE FAR *pNext;  // pointer to the next item
    struct _SPRITE FAR *pPrev;  // pointer to the prev item
    PDIB pDIB;                  // The DIB image of the sprite
    int x;                      // X Coordinate of top-left corner
    int y;                      // Y Coordinate of top-left corner
    int z;                      // Z order for sprite
    int vx;                     // X velocity
    int vy;                     // Y velocity
    int width;                  // width of bounding rectangle
    int height;                 // height of bounding rectangle
    BYTE bTopLeft;              // top left pixel value
    COLORREF rgbTopLeft;        // top left pixel color
    BOOL bSelectable;           // TRUE if sprite can be mouse selected
} SPRITE, FAR *PSPRITE;
```

The transparent regions of the sprite are determined by the color of the top-left pixel of its DIB. When the DIB is authored one color is reserved (it doesn't matter what color it is) for the

transparent regions and they are all filled with that color. The top-left pixel is also set to that color. This is easy to achieve in practice since it is natural that the corners of the sprite rectangle are transparent for most real object shapes.

To see how these parameters are set, let's look at the LoadSprite function in SPRITE.C

```
pSprite = (PSPRITE) ALLOCATE(sizeof(SPRITE));
```

Memory is allocated for the SPRITE structure.

```
pSprite->pDIB = LoadDIB(pszPath);
```

The DIB image of the sprite is loaded. If no path was supplied to LoadSprite a dialog is presented to select the DIB.

```
pSprite->width = (int) DIB_WIDTH(pSprite->pDIB);
pSprite->height = (int) DIB_HEIGHT(pSprite->pDIB);
pSprite->x = 0;
pSprite->y = 0;
pSprite->z = 0;
pSprite->vx = 0;
pSprite->vy = 0;
pSprite->bSelectable = TRUE;
pSprite->pNext = NULL;
pSprite->pPrev = NULL;
```

The defaults are set for the sprite parameters.

```
MapDIBColorTable(pSprite->pDIB, pdibBkGnd);
```

The color table in the DIB is mapped to the color table of the background DIB. This isn't required if all the sprite DIBs are authored with the same color table as the background DIB. If the color tables differ, this at least renders the sprite image in the best way possible rather than as a collection of seemingly random colors. The MapDIBColorTable function is in DIB.C.

```
pSprite->bTopLeft = GetDIBPixelValue(pSprite->pDIB, 0, 0);
pSprite->rgbTopLeft = GetDIBPixelColor(pSprite->pDIB, 0, 0);
```

The index value and color of the top-left pixel are saved for later use in determining the transparent areas of the image.

```
if (pSpriteList) {
    pSpriteList->pPrev = pSprite;
    pSprite->pNext = pSpriteList;
}
pSpriteList = pSprite;
```

The sprite is added to the top of the sprite list. For now the position in the list doesn't matter. It will be adjusted when the Z order is set.

```
SetSpriteZOrder(pSprite, 50, NO_UPDATE);
```

The Z order of the sprite is set to a default value. Z order zero is the front most sprite. Sprites with the same Z order are drawn with the one at the top of the sprite list front most. Painting is

done from the bottom of the list to the top. The list is always maintained in Z order so don't set the Z order value directly, use the SetSpriteZOrder function which correctly manages the list.

```
if (bRedraw != NO_UPDATE) {
    GetSpriteRect(pSprite, &rc);
    Redraw(&rc, UPDATE_SCREEN);
}
```

If redrawing the sprite was requested, the sprite rectangle is added to the redraw list. For a single sprite loaded manually from the menu it needs to be redrawn to be visible, but if the sprite is being loaded along with other sprites to form a scene then the redraw operation only needs to be done when all the sprites are loaded.

The DIB Functions and Macros

Since Windows provides no functions to deal with DIBs the way it does with bitmaps (DDBs) we need to create our own. The DIB.C module contains all the DIB handling functions used in the application except those used for rendering which are in the DRAW.C module. A brief description of each function is given here. See the code for more details. A set of macros are defined in GLOBAL.H for accessing various parameters of a DIB. They are also described briefly here.

The DIB Macros

Macro name	Function
DIB_WIDTH(pDIB)	Image width
DIB_HEIGHT(pDIB)	Image height
DIB_PLANES(pDIB)	Number of color planes
DIB_BITCOUNT(pDIB)	Number of bits per pixel
DIB_CLRUSED(pDIB)	Number of colors used
DIB_COLORS(pDIB)	Number of colors
DIB_PCLRTAB(pDIB)	Pointer to the color table
DIB_BISIZE(pDIB)	Size of the BITMAPINFO struct
DIB_PBITS(pDIB)	Pointer to the bits
DIB_PBI(pDIB)	Pointer to the BITMAPINFO struct
DIB_STORAGEWIDTH(pDIB)	Scan line storage width

Table 2. DIB Macros

The DIB Functions

PDIB LoadDIB(LPSTR pszPath)
Load a DIB. pszPath is the file path or NULL to invoke the file open dialog.

void DeleteDIB(PDIB pDIB)
Delete a DIB. If pDIB is NULL, the request is ignored.

BYTE GetDIBPixelValue(PDIB pDIB, int x, int y)
Get the value (color table index) of a pixel at coordinates x,y of the DIB pointed to by pDIB.

COLORREF GetDIBPixelColor(PDIB pDIB, int x, int y)
Get the color (RGB) of a pixel at coordinates x,y of the DIB pointed to by pDIB.

BOOL IsWinDIB(LPBITMAPINFOHEADER pBI)
Test if a DIB is Windows format (rather than Presentation Manager format).

void ShowInfo(LPBITMAPINFO lpBmpInfo)
A debugging function to display attributes of a DIB in the debug window.

WORD NumDIBColorEntries(LPBITMAPINFO lpBmpInfo)
Get the number of colors in the color table of a DIB. Used in the DIB_COLORS macro.

PDIB CreateCompatibleDIB(PDIB pOld)
Create a new DIB the same size and color format as an existing DIB.

HPSTR GetDIBPixelAddress(PDIB pDIB, int x, int y)
Get a pointer to the pixel at address x,y in the DIB pointed to by pDIB.

void MapDIBColorTable(PDIB pdibObj, PDIB pdibRef)
Map the colors in the color table of the DIB pointed to by pdibObj to the colors in the color table of the DIB pointed to by pdibRef. This is done by creating a temporary DIB Driver DC the same size as the object DIB, with the color table of the reference DIB. The object DIB is then rendered to the DIB Driver DC using StretchDIBBits with the DIB_RGB_COLORS option. The resulting bits (now mapper tot he reference color table) are copied back to the object DIB bits.

The Drawing Functions

The module DRAW.C contains all of the functions to render images to the off-screen DIB Driver DC and to the window DC. The most important functions are RenderDIBBitsOffScreen, Redraw and Paint.

RenderDIBBitsOffScreen

This function is used to render the background DIB and the sprites to the off-screen DC. It uses two functions: CopyDIBBits and TransCopyDIBBits in the FAST32.ASM module to perform the actual bit transfers. Here's a description of the function:

```
rcDraw.top = rcDraw.left = 0;
rcDraw.right = DIB_WIDTH(pdibOffScreen) - 1;
rcDraw.bottom = DIB_HEIGHT(pdibOffScreen) - 1;

if (prcClip) {
    if (!IntersectRect(&rcDraw, &rcDraw, prcClip)) return;
}

rcDIB.left = x;
rcDIB.right = x + DIB_WIDTH(pDIB) - 1;
rcDIB.top = y;
rcDIB.bottom = y + DIB_HEIGHT(pDIB) - 1;

if (!IntersectRect(&rcDraw, &rcDraw, &rcDIB)) return;
```

The function is supplied with a clipping rectangle describing the area to be drawn into. The first step is to intersect that rectangle with the off-screen DIB boundary so we don't try to draw off the DIB. If there is no intersection then there is nothing to do. The resultant rectangle is intersected again, this time with the bounding rectangle of the DIB itself to ensure we aren't going to be doing more work than is really necessary. Again, if there is no intersection, the DIB isn't visible and the function returns.

```
pStartS = GetDIBPixelAddress(pDIB,
                             rcDraw.left - x,
                             rcDraw.bottom - y);

pStartD = GetDIBPixelAddress(pdibOffScreen,
                             rcDraw.left,
                             rcDraw.bottom);

lScanS = DIB_STORAGEWIDTH(pDIB);
lScanD = DIB_STORAGEWIDTH(pdibOffScreen);
```

The address of the bottom-left corner of the draw rectangle is found in both the source DIB and the destination DIB. The length of the physical scan line for each DIB is obtained. The addresses represent the lowest address of the DIB bits we need to copy.

```
if (!bTrans) {
    CopyDIBBits(pStartD,
                pStartS,
                rcDraw.right - rcDraw.left + 1,
                rcDraw.bottom - rcDraw.top + 1,
                lScanD,
                lScanS);
} else {
    TransCopyDIBBits(pStartD,
                     pStartS,
                     rcDraw.right - rcDraw.left + 1,
                     rcDraw.bottom - rcDraw.top + 1,
                     lScanD,
                     lScanS,
                     bTranClr);
}
```

If the DIB is to be treated as non-transparent (as for the background DIB) then the CopyDIBBits function is called. If the DIB has a transparency color associated with it (as for a sprite DIB) then the TransCopyDIBBits function is used. These two copy functions are implemented in 32 bit assembler in FAST32.ASM.

Redraw

This function is used in two ways depending on whether the screen needs to be updated or not. When rendering multiple images, it is important to do the least amount of work so the function is used to add items to the redraw list and optionally to do the actual redraw. Here's what goes on:

```
if (prcClip) {
    AddDrawRectItem(&DrawList, prcClip);
} else {
    if (pdibBkGnd) {
```

```
            rcAll.left = rcAll.top = 0;
            rcAll.right = DIB_WIDTH(pdibBkGnd);
            rcAll.bottom = DIB_HEIGHT(pdibBkGnd);
            AddDrawRectItem(&DrawList, &rcAll);
        }

    }
    if (bUpdate == NO_UPDATE) return;
```

The first stage is to add the supplied clipping rectangle to the redraw list. If NULL was specified, this is interpreted as meaning that the entire window needs to be redrawn and this causes a rectangle the size of the background DIB to be added to the list instead. If no request was made to update the screen, the function exits here.

```
    MergeDrawRectList(&DrawList);
    pLastSprite = pSpriteList;
    if (pLastSprite) {
        while (pLastSprite->pNext) pLastSprite = pLastSprite->pNext;
    }
    if (bUpdate == UPDATE_SCREEN) {
        hDC = GetDC(hwndMain);
    }
    pDrawRect = DrawList.pHead;
    while (pDrawRect) {
        RenderDIBBitsOffScreen(pdibBkGnd,
                               0, 0,
                               &(pDrawRect->rc),
                               0,
                               FALSE);
        pSprite = pLastSprite;
        while (pSprite) {
            RenderSpriteOffScreen(pSprite, &(pDrawRect->rc));
            pSprite = pSprite->pPrev;
        }
        if (bUpdate == UPDATE_SCREEN) {
            Paint(hDC, &(pDrawRect->rc));
        }
        pDrawRect = pDrawRect->pNext;
    }
    if (bUpdate == UPDATE_SCREEN) {
        ReleaseDC(hwndMain, hDC);
    }
    EmptyDrawRectList(&DrawList);
```

The first step in the rendering process is to merge all the overlapping rectangles in the redraw list. This results in a list of non-overlapping rectangles and gives us the least area which will need to be modified. Pointers are obtained to the last sprite in the sprite list and the first rectangle in the redraw list. The sprite list is walked form bottom to top so that high Z order sprites (which are at the bottom of the list) appear at the back of the scene.

Then for each rectangle, the background is replaced in the off-screen DC by calling RenderDIBBitsOffScreen. The sprite list is then walked, rendering each sprite. The clipping of each sprite is handled in part by the RenderSpriteOffScreen function and in part by the RenderDIBBitsOffScreen function.

If Redraw was called with the UPDATE_SCREEN flag, the screen DC is repainted by calling the Paint function for the current draw rectangle.

When all of the rectangles in the list have been redrawn, the list is reset to empty by a call to EmptyDrawRectList.

All of the drawing rectangle functions can be found in DRAW.C.

Paint

The Paint function handles updating the screen DC from the off-screen DIB Driver DC. Here's the code:

```
if (prcClip) {
    w = prcClip->right - prcClip->left;
    h = prcClip->bottom - prcClip->top;
    xs = xd = prcClip->left;
    yd = prcClip->top;
    ys = DIB_HEIGHT(pdibOffScreen) - prcClip->bottom;
} else {

    w = DIB_WIDTH(pdibOffScreen);
    h = DIB_HEIGHT(pdibOffScreen);
    xs = xd = ys = yd = 0;
}
```

The width and height of the rectangle to be copied and the start point in the off-screen DC and the window DC are computed based on either the supplied clipping rectangle or the size of the off-screen image.

```
if (hpalCurrent) {
    hOldPal = SelectPalette(hDC, hpalCurrent, 0);
    RealizePalette(hDC);
}
```

The current palette (obtained originally from the background DIB color table) is selected into the screen DC and realized. This operation only takes any significant amount of time the first time it is called after the application has become the foreground application.

```
StretchDIBits(hDC,              // dest dc
            xd,                 // dest x
            yd,                 // dest y
            w,                  // dest width
            h,                  // dest height
            xs,                 // src x
            ys,                 // src y
            w,                  // src width
            h,                  // src height
            DIB_PBITS(pdibOffScreen), // bits
            pPalClrTable,       // BITMAPINFO
            DIB_PAL_COLORS,     // options
            SRCCOPY);           // rop
```

StretchDIBits is called to copy the bits of the off-screen image to the screen memory. Note the

use of DIB_PAL_COLORS and the 1:1 color lookup table (pPalClrTable). See the section on performance below for more details.

```
    if (hOldPal) SelectPalette(hDC, hOldPal, 0);
```

The palette in the screen DC is restored.

Updating Positions

The UpdatePositions function really has very little to do with creating sprites. I included this to show off the performance by whizzing a few images across the screen. The code walks down the sprite list looking for any sprites which have a non-zero velocity vector. When it finds one it adds the current position to the redraw list so that it will be erased from there. The sprite position is updated and the new position of the sprite added to the redraw list. In both cases the Redraw function is called with the NO_UPDATE option to prevent any actual draw operations until we are done walking the list.

Once all the sprites have been updated a test is made to see if any changes occurred and if so the Redraw function is called one more time, but this time with the UPDATE_SCREEN option which causes the redraw rectangle list to be walked and redrawn with all the updates being reflected to the screen DC.

If you want to do special case sprite movements (funny trajectories, flipping sprite images as they move and so on) this is the place to add the code.

The Scene File Format

To make life easier while debugging the SPRITES application, I created an INI file to describe what background DIB to use and what sprites to load. The format isn't very tidy but it's documented here so that you can create your own files. A nice improvement to the application would be an option to save the current scene in a file.

Here's part of the GARDEN.INI file showing the description of the background and two of the sprites:

```
[Background]
dib=bkgnd.dib
```

This describes the background DIB to use. There is only one entry to specify the name of the DIB file. The default is to load the file from the current directory. If this isn't what you want, put in the full path.

```
[Sprites]
sun=1
cloud2=1
```

This section gives a list of sprites to be loaded. The '=1' bit is bogus. Only the list of names are used by the loader code.

```
[Sun]
x=400
y=0
z=99
dib=sun.dib
selectable=0
vy=1
```

This describes the 'sun' sprite listed in the [sprites] section.

```
[Cloud1]
x=150
y=20
z=80
vx=2
dib=cloud16.dib
```

This describes one of the 'cloud sprites listed in the [sprites] section.

Table 3 shows the list of attributes for a sprite, a description of each one and the default if the entry is omitted.

Key	Description	Default value
dib	The path of the DIB file.	(none) This is a required entry.
x	The X position of the top-left corner of the sprite.	0
y	The Y position of the top-left corner of the sprite.	0
z	The Z order value. O is front most. More than one sprite can have the same Z order value.	50
vx	The X velocity in pixels per redraw cycle.	0
vy	The Y velocity in pixels per redraw cycle.	0
selectable	If the sprite can be selected for dragging by the mouse.	1 (DIB can be selected)

Table 3. The sprite attributes.

Note that you cannot enter negative values directly because the code uses GetPrivateProfileInt to retrieve them. Negative values must be entered as 16 bit two's compliment values. Enter -1 for example) as 0xFFFF or 65535.

Note: This is an example of a stupidly named Windows function. It says 'Int' in the name but always works with unsigned ints. This rather frustrating and brain-dead feature will unfortunately

be with us for life. Such is the mission of backward compatibility in Windows.

Measuring Performance

One of the problems with this sort of work is that analysis of the results can be rather subjective which is something I detest. Often I hear: "Hmm, I'm sure it was faster before you did that". It's very nice to be able to respond with a list of timings and prove the point objectively. To this end I have used two different ways to measure how long various operations in the code took to execute.

Using Software Timing

The first timing technique involves reading the system clock at the start and end of a section of code and reporting the elapsed time. This technique is OK with two provisos: that the time measurements are accurate enough and that displaying the timing information does not contribute significantly to the times being measured.

Timing measurements were done with the MMSYSTEM function timeGetTime rather than the regular Windows function GetTickCount because timeGetTime returns a millisecond count accurate to the nearest millisecond and GetTickCount returns a millisecond count only accurate to the nearest 55 ms (one DOS clock tick - hence the name).

Displaying the results using the debug print statements (dprintf) is however rather invasive and if you use this technique you must be aware that painting the debug information window is a slow process and while this might not contribute to the execution time of the piece of code you are measuring, it certainly does contribute to the overall execution time of the application and hence slows down the animation significantly. Using a small (two or three lines) debug window helps to minimize this.

Despite the invasive nature of showing the results, the timings of small sections of code are quite accurate and very helpful in determining whether optimizing a particular piece of code is worthwhile and when the work has been done, what the improvement was.

Using Hardware Techniques

The second technique I used is covered in another article: "Use Your Printer Port to Measure System Timings" by the same author. This technique sets and clears bits of a printer port at various places in the code. By using an oscilloscope to monitor when the transitions of the bits take place, accurate timing measurements can be made. Please see the article mentioned above for details of how this technique works.

Using the oscilloscope to monitor various operations in the code I discovered several performance problems, the most notable being that allocating small blocks of memory with GlobalAlloc is very slow and inefficient.

Improving Performance

While creating the SPRITES sample I measured the execution times of various bits of my code in order to determine where the bottlenecks were. This section describes what I found out.

Memory Allocation

The first discovery I made when I started looking at performance was that using GlobalAlloc to allocate all of the dynamic memory blocks in the application isn't the best choice. I had defined a macro to allocate and free memory blocks so that the technique could be easily changed if required. I had decided to allocate all memory the same way - which essentially ruled out using local memory for anything. This was a big mistake. I was using my ALLOCATE macro (which calls GlobalAlloc) to allocate the elements of the redraw rectangle list. Using the oscilloscope, I measured the time taken to be almost 3 ms on my 386/33 Compaq (in Enhanced Mode). I was quite disgusted by this and rewrote the rectangle functions to use local memory blocks allocated with LocalAlloc and near pointers to access the structures. The result was that the execution time of AddDrawRectItem fell from about 3 ms to about 50 us. This cut almost 6 ms of the 72 ms cycle time of one of my test scenes which has a single sprite (a dog) moving one pixel at a time from left to right across a plain blue background.

The memory for the SPRITE structure could also be allocated locally and the FAR pointers used within the structure for pPrev and pNext replaced with near pointers. This might give a small improvement in the rendering time since the sprite list is walked quite often.

Rendering DIBs to the Off-Screen DIB

The first try at rendering the sprite DIBs to the off-screen DC was a complete disaster. I tried to use raster ops to create monochrome masks for the transparency regions of each sprite and to insert the non-transparent areas into the off-screen DIB. After many hours of fruitless work and a phone call or two I was reminded that the DIB driver supports a background mode called NEW_TRANSPARENT and if this is set, StretchDIBits will treat the current background color as the transparency color of a bitmap. The resulting code is implemented in the RenderDIBOffScreen function in DRAW.C.

Close examination of the code shows that the call to StretchDIBits is made using the DIB_RGB_COLORS option which causes the RGB values in the color table of the source DIB to be mapped to the RGB values of the color table in the destination DIB (the off-screen DIB in this case). This is vary handy if your two DIBs have different color tables but is rather a waste of time if they are the same. Even assuming that you wanted to use DIBs with different color tables, it seems ridiculous to do the mapping every time the DIB is rendered. Why not do the mapping once and save the results.

To do this I created the MapDIBColorTable function in DIB.C. This function creates a temporary DC using the DIB driver, renders the DIB to the DC using StretchDIBits with the DIB_RGB_COLORS option and then copies the resulting bits in the DIB driver DC DIB back to the original source DIB. Net result is that the pixels in the source DIB are now mapped to the color table supplied by the reference DIB (the background DIB in our case).

So far, so good. I then got way too clever and modified the StretchDIBits call in the RenderDIBOffScreen function to use DIB_PAL_COLORS with a 1:1 color table (as is used in the Paint routine). This didn't work - I got weird colors. After some research I discovered that you can't use DIB_PAL_COLORS when working with a non-palletized device and the DIB Driver is a non-palletized device. What happens is that GDI tries to be helpful. It detects that the device driver isn't palletized, maps the palette indices to RGB values and sends those to the driver. Net result - a waste of processor time, and the wrong colors. There is a way around this problem however. In MMSYSTEM.H there is a macro called DIBINDEX which can be used to define the colors in the DIB color table as palette index values:

```
clr[n] = DIBINDEX[n]
```

This sets a special flag in the COLORREF structure which the DIB driver recognizes and instead of matching the RGB values normally found in the color table it treats the low byte as a palette index. But by this time I'd lost interest in using StretchDIBits and decided to do my own thing.

I decided to implement a simple function to simply copy the bits of the sprite DIB to the off-screen DIB (ignoring the transparency problem for the moment). This worked great. The performance was way better and the colors were right. I added some code to implement the transparency feature (all of this in C) and although I now had transparent sprites again, the performance went way back down. The cause of the loss of performance being the code generated by the C compiler which dealt with all the huge pointers I was using. Still the idea seemed good.

I recoded the function to copy a scan line of the DIB in assembler and found that the speed improved quite a lot. I improved the assembler function to copy a whole block rather than calling it multiple times to copy scan lines and the result (which is in FAST16.ASM) was another small speed improvement.

The final touch was to use a technique developed by Todd Laney to create a 32 bit code segment for the assembler routine to run in. This code can be found in FAST32.ASM and the macros which support it in CMACRO32.INC. The code was taken from FAST16.ASM and optimized by Todd. There are several rules regarding using 32 bit code segments and how to generate them. These are detailed in the section: "Using 32 Bit Code Segments" below.

Table 4 shows some timings measured at various points through the development cycle of this piece of code. The cycle time shown is the time the application takes to complete one move, render, redraw cycle. The background time is the time taken to render the background DIB (which is not transparent) into the area the sprite is being moved from, and the sprite time is the time taken to render the sprite image in the new position. All times are in ms. The sprite was smolivia.dib which is a 256 color image. For all cases the paint time was constant at about 7 ms.

Description	**Cycle**	**Bkgnd**	**Sprite**
StretchDIBits with DIB_RGB_COLORS	300	12	280
StretchDIBits with DIB_PAL_COLORS	75	35	35
Primitive C code rectangle copy	12	-	-
C code with transparency using huge ptrs	40	1.8	31
Using 16 bit assembler to copy lines	17	2	6
Using 16 bit assembler to copy blocks	16	1.	5.

		6	5
Using 32 bit assembler to copy blocks	16	1.6	4.6
Optimized 32 bit assembler	11	0.6	2.6

Table 2. Code timings

There is one further improvement which is as yet untried. I expect that using run length encoded DIBs (RLE) to define a sprite might be more efficient. The RLE code scheme allows for blocks of the DIB to be skipped over using a form of relative addressing. This scheme could be used to encode only the visible areas of a sprite. For images which have a lot of transparency, this could give great improvements in the rendering time.

Updating the Window DC from the Off-Screen DIB

The first and most important thing to know here is that you can't use BitBlt to move bits from the DIB Driver DC to the screen DC. BitBlt is a device driver implemented function and each driver only understands its own DC's. The DIB driver owns the DIB Driver (off-screen) DC and the screen device driver owns the window DC. This rule also prevents you from using BitBlt to copy data from a screen DC to a printer DC. Where I most often get caught out by this is in creating compatible DCs. It's very easy to forget that a particular DC was created (and therefore is owned by) a particular device driver. So if you create a DC compatible with an existing printer DC or DIB Driver DC, you still can't use BitBlt to move data between that DC and any DC owned by the screen device driver. This is where the DIB specific functions come in.

Updating the window from the off-screen DIB is reasonably straightforward:

1) The palette (created from the background DIB when it was loaded) is selected into the window DC and realized.
2) StretchDIBits is called with the DIB_PAL_COLORS option and a 1:1 color table.
3) The original palette is reselected into the DC.

That's all it takes. Using the DIB_PAL_COLORS option prevents any color matching. Actually, the first time the palette is realized, GDI establishes a new system palette and modifies the index values in the logical palette so that they map directly to the system palette. Once this has been done there is no need to map a logical palette index to a physical palette index and consequently the process runs very quickly. This mapping stays valid so long as the system palette doesn't change.

Redraw Rectangle List

Since rendering images to the off-screen DC and copying the changed areas to the main window DC are really what take all the time, an attempt is made to reduce the amount of blitting done to a minimum. To achieve this a list of rectangles is maintained which need to be redrawn on the off-screen DIB and repainted to the window DC. As the first step in performing a redraw, the redraw list is merged so that any overlapping rectangles are combined into a single rectangle. Then,

each rectangle in the list is redrawn. In this way the minimum area is modified each time. Use the "ShowUpdateRects" menu item in the "Debug" menu to look at the redraw areas (in cyan) and the repaint to the screen areas (bordered in magenta).

Using Assembler Code

Writing critical sections of the code in assembler can provide two advantages. First, the code is often smaller and faster than that generated by the compiler and second, because of the way Windows allocates descriptor table entries for large (> 64k) memory blocks, you can access an entire block of memory (a packed DIB for example) with the selector to the base of the block and a 32 bit offset. Simply using a 32 bit register (ESI for example) for the offset forces the assembler to include an addressing modifier into the code which results in a 32 bit offset being used. This is how the FAST16.ASM module works.

Using 32 Bit Code Segments

You can include also 32 bit code segments in your application. This is almost exactly the same as the 16 bit assembler case except that the code is now running with 32 bit offsets by default so no addressing modifier needs to be included before an instruction which uses a 32 bit offset. On a 386 there is almost no improvement in performance from this because the execution time of the modifier is so low, but on a 486 the presemce of a modifier alters the instruction pipelining resulting in a very noticable reduction in performance. So using 32 bit code segments gives a minor improvement on a 386 (a few percent) and usually a one and a half to two times improvemetn on a 486. **Using 32 bit code segments has nothing to do with a flat model address space, you are still working with segment and offset.** Currently, the only way to implement this is in assembly language using a special version of the C macros. There are a some regulations governing how these segments must be created and used:

1) The segment must be separate from any 16 bit code segments. To ensure this, name it with a unique name (such as TEXT32) and link it with the /NOPACKCODE option in the linker to prevent the segment being merged with other segments. Failure to keep the segment isolated will result in a GP fault when one of your 32 bit functions attempts to return to the 16 bit calling code.

2) The 32 bit code segment should not call any 16 bit code. This limits the 32 bit code to doing data manipulation tasks. This is partly to do with the fact that the 32 bit code is running with a 16 bit stack.

3) The assembly code must include the special CMACRO32.INC file. This is a modified version of CMACROS.INC which handles 32 bit code segments.

4) The code must be assembled with MASM 5.1 not MASM 6.0 which is not backward compatible with the macros. A version of the macros compatible with MASM 6.1 will be included with this technote and its sample if they become available.

The way that the 32 bit segments are integrated is quite ingenious. Each procedure has a short piece of code at its start which determines if the segment is running in 32 bit mode. If it is found to be in 16 bit mode, a jump is made to a piece of fix-up code at the start of the segment which modifies the descriptor table entry for the segment to change the mode to 32 bit and then returns to the called function which now executes in 32 bit mode. This fixup only needs to occur once for the segment. Once it has been set to 32 bit mode it stays that way.

There is one minor problem with using 32 bit code segments. The linker sets a bit in the EXE file header in the segment table to mark the 32 bit code segment as being a 32 bit segment - much as you might expect. Windows ignores many of the flags in the segment information including

this one. However the ROM Windows version of Windows makes use of this bit to mark a segment as being in ROM which causes disastrous results if you try to run an application with 32 bit code segments in RAM. This problem can be cured by modifying the flag bits in the segment information table of the EXE file. There is currently no tool available to do this.

Sprite Design

Since the performance is limited largely by the amount of data being moved about, make sure that sprites don't have borders around them. Even though the border is transparent, these areas cause trouble because the entire sprite rectangle needs to be redrawn whenever the sprite is moved.

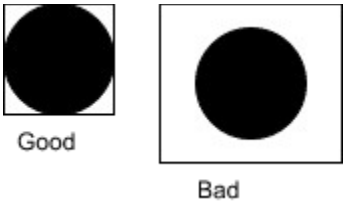

Figure 9. Good and Bad Sprite Designs

Run Time Decisions

Not all device drivers have the StretchDIBits function implemented. Even in some of the drivers which do have it implemented, the implementation may not be all that good. If StretchDIBits is not implemented in the driver then GDI will simulate it by using combinations of SetDIBits and BitBlt. This requires the use of an intermediate DDB which GDI uses to band the image in 64k chunks. This is horrendously slow. The Video for Windows code tests the perfomrance of StretchDIBits when it starts up by timing a few calls to StretchDIBits against calls to SetDIBits and BitBlt for the same image. If StretchDIBits is slower, then the driver is either missing the StretchDIBits function or has a very bad implementation. In any case, the Video for Windows code reports the problem to the user and carries on using the SetDIbits and BitBlt option. Note that although the application still runs without a device driver implementation of StretchDIBits it is much slower than it would be if the function were present.

Notes About BitBlt

During the early stages of developing the SPRITES application I tried creating DDBs from the sprite DIBs and then creating monochrome bitmap masks for their transparent areas. The bitmaps were combined together with the background bitmap using various raster operations. Even though BitBlt isn't used anywhere in the final SPRITES code, I though my notes might still be of some interest:

1) When a DC is first created (for example by calling CreateCompatibleDC) it has a default 1 x 1 monochrome bitmap selected into it. So if you subsequently call CreateCompatibleBitmap you will get a monochrome bitmap. This isn't what you'd expect if you started with a color DC, created a compatible DC from that and then asked for a compatible bitmap. To ensure that your bitmap is the same color format as the original DC get the color information by calling GetDeviceCaps for BITSPIXEL and PLANES and then use that information in a call to CreateBitmap:

```
BitsPixel = GetDeviceCaps(hDC, BITSPIXEL);
Planes = GetDeviceCaps(hDC, PLANES);
```

```
hbm = CreateBitmap(width, height, Planes, BitsPixel, NULL);
```

2) When using BitBlt to convert a color bitmap to a monochrome bitmap, if the pixel color of the source bitmap is the same as the current background color then the monochrome output is white (1) otherwise it is black (0).

3) If you want to be able to use blt operations like XOR and AND to do transparency masking onto a DIB dc, the sprite image and mask must also be in DIB DCs. This is because BitBlt cannot work with device contexts from two different drivers.

Unimplemented Features

The application only supports 8 bit DIBs. It won't let you load a 2 or 4 bit DIB. It also doesn't support RLE DIBs.

It turned out that once the code was complete, the DIB Driver had taken rather a back seat in the design and in fact could be removed all together. The only function it provides is to map RGB color values to index values and this could just as easily be done with a small piece of code.

The background DIB is treated as a special case throughout the code. It might have been better to treat it as just another sprite with a Z order value of 65535. When loaded (as the background DIB) it would still be used to create the common palette but once that was done it would simply be added to the tail end of the sprite list.

Some sprites are simply rectangles and in these cases it would be nice to be able to set a flag to say that transparency does not apply to this sprite. This could easily be done by changing the BYTE value used in the bTransColor field of SPRITE to be a UINT (or WORD) with a special reserved vlaue of 0xFFFF meaning that there is no transparent color.

Another possible improvement to the architecture might be to use two off-screen DIBs in a double buffering arangement. Each time an image was rendered to a buffer, the difference between the new image and the old one in the other buffer would be recorded (in a third DIB) and then converted to RLE before being sent to the screen DC. This might look like a lot of work but has the advantage that it minimizes the amount of data going to the screen DC which is a limiting factor. Doing the render, compare and update to the off-screen DIBs could be done very efficiently with a small piece of assembly code.

Windows NT and Chicago Issues

The DIB Driver is not provided in either Windows NT or Chicago. Instead there will be a new API called CreateDIBSection which will allow you to do the same things but in a slightly different way. If you want to be able to port your application trivially to either Chicago or Windows NT then you should remove the DIB Driver and use your own code to do the RGB to index color conversions and also remove the 32 bit assembler code since this is very platform specific.

The ALLOCATE and FREE macros used for memory management in the sample don't map very well to Windws NT functions. A better approach would be to include the windowsx.h header file and use the GlobalAllocPtr and GlobalFreePtr macros which have direct mapping to NT functions.

Another issue with porting to NT is the use of LPSTR casts when doing address arithmetic such as is used to calculate the offset of DIB bits in a packed DIB. It is better to use LPBYTE since this points to an 8 bit object in both 16 and 32 bit Windows environments. LPSTR points to a 16 bit object in Windows NT.

End.

Waveform Conversion Sample Application
Multimedia Systems Group

If you did not get this from Microsoft Sources, then it may not be the most current version. This sample code in particular will be updated and include more documentation.

Sources are:

The MM Sys BBS:	The phone number is 206 936-4082.
CompuServe:	WINSDK forum, MDK section.
Anonymous FTP from:	ftp.uu.net vendor\microsoft\multimedia\samples
Version 1.0	Released 09/04/92

This sample ONLY works on the following:

- a 386 or better
- Windows 3.1 (or Windows 3.0 + MME 1.0 + COMMDLG)

File	Description
makefile	The makefile
msadpcm.c	Microsoft ADPCM conversion routines.
msadpcm.h	
muldiv32.asm	utility file
pcm.c	PCM conversion routines (WARNING: under construction)
pcm.h	
readme.txt	This file
riffsup.c	Support routines for RIFF files
riffsup.h	
waveconv.c	The main file
waveconv.def	The .DEF file
waveconv.exe	Our application
waveconv.h	
waveconv.ico	Our application's ICON
waveconv.rc	Our application's resource file
waveconv.rcv	Version information for the resource file
waveio.c	Waveform conversion support functions
waveio.h	
wavesup.c	RIFF WAVE support functions
wavesup.h	
wavever.h	Version include file
wavever.ver	The (static) version resource file

Notes about this sample

This application can be used to load, edit and convert waveform data containted in the industry standard RIFF WAV file format. This file format is a standard developed by IBM and Microsoft. Specifications for this format are available by calling 206 936-8644 and are also included in the Windows 3.1 SDK documentation and the Multimedia Programmer's Reference book from Microsoft Press. This format is directly supported by Windows and OS/2.

This program demonstrates conversion to/from the Microsoft ADPCM format (MS ADPCM). DVI ADPCM format is being registered by the IMA and will be added soon.
This sample code does not yet support editting of DISP chunk information.

If you have any comments/questions/requests about this sample, you
can reach me at the addresses below, or by FAX at 206 93MSFAX (7329).
Please let me know if this sample code is usefull to you.

Matt Saettler
CIS: 76150,2523
Internet: matts@microsoft.com
One Microsoft Way
Redmond, WA 98052 USA

End.

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.

Microsoft MS-DOS CD-ROM Extensions

Version 2.21

Hardware-Dependent Device Driver Specification

This document describes the CD-ROM hardware-dependent device driver and its interface with MSCDEX.EXE, the MS-DOS CD-ROM Extensions resident program. Differences between CD-ROM drives and hard- or floppy-disk drives account for the differences in this device driver specification from the normal MS-DOS block and character device driver specification. For more information on device drivers, see the *MS-DOS Programmer's Reference Manual* and *Advanced MS-DOS Programming* by Ray Duncan. (Both books are published by Microsoft Press.) For information on the terms describing for CD-ROM operating characteristics, see the IEC 908 and ISO 10149 specifications.

The MS-DOS operating system reads CONFIG.SYS and installs the device. MSCDEX.EXE performs an open system call on the device driver name in order to communicate with it and uses an IOCTL call to ask the device driver for the address of its device header. From the device header address, MSCDEX.EXE locates the device driver's interrupt and strategy routines. After that, all requests the device driver receives come directly from MSCDEX.EXE, not MS-DOS. To avoid reentrancy problems and allow MSCDEX to monitor all media changes, all other applications that wish to communicate directly with CD-ROM device drivers should do so through the *Send Device Driver Request* INT 2Fh function 1510h. MSCDEX.EXE interfaces with MS-DOS so that normal requests for I/O with files on a CD-ROM drive down to the MS-DOS INT 21h service layer will work just as they would for a normal MS-DOS device.

For maximum compatibility between your CD-ROM device driver, MSCDEX, and other components in the system, observe the following guidelines when developing your device driver:

- Avoid disabling interrupts for extended periods.

- CD-ROM drives and drivers that use DMA must use Windows virtual DMA device services for 386 enhanced mode operation. For information for using Windows virtual DMA device services, see the *Microsoft Windows Device Driver Adaptation Guide* included in the Microsoft Windows Device Development Kit.

Installation

The device driver will be installed in the same way as any other device with an entry in CONFIG.SYS. The syntax is:

DEVICE=<*filename*> /D:<*device_name*> /N:<number of drives>

The following are examples:

DEVICE=HITACHI.SYS /D:MSCD001 /D:MSCD002

DEVICE=SONY.SYS /D:MSCD003 /N:2

The arguments will be the character device names that will be used on the command line when starting MSCDEX.EXE so that it can find and communicate with the device driver.

A device driver may support one or more physical drives or logical disks. This may be done by having multiple device headers in the device driver file (in which case it will be necessary to have more than one *device_name* on the command line - one for each device header; see the HITACHI.SYS example above) or through the use of subunits. Each disk handled by a device driver that supports multiple disks using subunits is addressed by the subunit field of the request header when a request is made for that disk. A device driver that supports more than one disk can share code and data instead of requiring separate device drivers for each disk. A "jukebox" CD-ROM system would be an example of a CD-ROM device that might wish to support more than one drive or a disk pack using a single device driver.

Device drivers that use multiple subunits should use the optional switch /n:<number of drives> to say how many drives are present. If not present, the default number of drives is 1. If the driver can tell how many drives are installed without a command line switch, then this argument is not necessary. Unless there are special considerations, it is better practice to support multiple drives using subunits than to have multiple device headers in the same device driver file.

Device header

The device header is an extension to what is described in the *MS-DOS Programmer's Refrence Manual*.

```
DevHdr    DD    -1          ; Ptr to next driver in file or -1 if last driver
          DW    ?                 ; Device attributes
          DW    ?                 ; Device strategy entry point
          DW    ?                 ; Device interrupt entry point
          DB    8 dup (?)         ; Character device name field
          DW    0                 ; Reserved
          DB    0                 ; Drive letter
          DB    ?                 ; Number of units
```

The following are the device attributes for MSCDEX.EXE device drivers:

```
Bit 15      1     - Character device
Bit 14      1     - IOCTL supported
Bit 13      0     - Output 'till   busy
Bit 12      0     - Reserved
Bit 11      1     - OPEN/CLOSE/RM supported
Bit 10-4 0      - Reserved
Bit 3       0     - Dev is CLOCK
Bit 2       0     - Dev is NUL
Bit 1       0     - Dev is STO
Bit 0       0     - Dev is STI
```

MSCDEX.EXE device drivers will be character devices that understand IOCTL calls and handle OPEN/CLOSE/RM calls.

The drive letter field is a read-only field for the device driver and is initialized to 0. The field is for MSCDEX.EXE to use when it assigns the device driver to a drive letter (A = 1, B = 2...Z = 26). It should never be modified by the device driver. For drivers that support more than one unit, the drive letter will indicate the first unit, and each successive unit is assigned the next higher drive letter. For example, if the device driver has four units defined (0-3), it requires four drive letters.

The position of the driver in the list of all drivers determines which units correspond to which drive letters. If driver ALPHA is the first driver in the device list, and it defines 4 units (0-3), they will be A, B, C, and D. If BETA is the second driver and defines three units (0-2), they will be E, F, and G, and so on. The theoretical limit to the number of drive letters is 63, but it should be noted that the device installation code will not allow the installation of a device if it would result in a drive letter > 'Z' (5Ah). All block device drivers present in the standard resident BIOS will be placed ahead of installable device drivers in the list.

NOTE: It is important that one set *lastdrive=<letter>* in CONFIG.SYS to accommodate the additional drive letters that CD-ROM device drivers will require.

The number-of-units field is set by the device driver to the number of disks that are supported. Normal character devices do not support more than one unit and MS-DOS does not expect a character device to handle more than one unit or have a nonzero subunit value in the request header. Since these device drivers are not called by MS-DOS directly, this is not a problem. Nonetheless, the number of units returned by the device driver in the number-of-units field during the INIT call must be 0, since MS-DOS makes the INIT call and does not expect a nonzero value for a character device. MSCDEX.EXE will never see what is returned anyway, and relies on the number-of-units field in the device header.

The signature field is necessary for MSCDEX confirmation that the device driver is a valid cd-rom device driver and consists of the 4 bytes 'MSCD' followed by two ascii digits for the version which is '00' at present.

Sample device header:

```
HsgDrv    DD    -1              ; Pointer to next device
          DW    0c800h          ; Device attributes
          DW    STRATEGY        ; Pointer to device strategy routine
          DW    DEVINT          ; Pointer to device interrupt routine
          DB    'MSCD003 '      ; 8-byte character device name field
          DW    0               ; Reserved (must be zero)
          DB    0               ; Drive letter (must be zero)
          DB    1               ; Number of units supported
                                ; (one or more)
```

As with other MS-DOS device drivers, the code originates at offset 0, not 100H. The first device header will be at offset 0 of the code segment. The pointer to the next driver is a double word field (offset/segment) that is the address of the next device driver in the list, or -1 if the device header is the only one or the last in the list. The strategy and interrupt entry points are word fields and must be offsets into the same segment as the device header. The device driver is expected to overwrite the name(s) in each of its one or more device headers with the <*device_name*> command line arguments during its initialization.

MSCDEX.EXE will call the device driver in the following manner:

1. MSCDEX.EXE makes a far call to the strategy entry.

2. MSCDEX.EXE passes device driver information in a request header to the strategy routine.

3. MSCDEX.EXE makes a far call to the interrupt entry.

Request header

MSCDEX.EXE will call the device's strategy routine with the address of a request header in ES:BX. The format of the request header is the same as what is described in the *MS-DOS Programmer's Reference Manual.*

```
ReqHdr    DB    ?           ; Length in bytes of request header
          DB    ?           ; Subunit code for minor devices
          DB    ?           ; Command code field
          DW    ?           ; Status
          DB    8 dup (?)   ; Reserved
```

Status

The status word also has the same format as described in the *MS-DOS Programmer's Reference Manual.* It is 0 on entry and is set by the device driver.

```
Bit 15      - Error bit
Bit 14-10   - Reserved
Bit  9      - Busy
Bit  8      - Done
Bit  7-0    - Error code (bit 15 on)
```

Bit 15, the error bit, is set by the device driver if an error is detected or if an invalid request is made to the driver. The low 8 bits indicate the error code.

Bit 9, the busy bit, should be set by the device driver when the drive is in audio play mode. Device drivers should fail all requests to the physical device that require head movement when the device is playing and return the request with this bit and the error bit set and an error code. Requests that would not interrupt audio play may return without error but will also have this bit set when the drive is in audio play mode. Play mode can be terminated prematurely with a reset or STOP AUDIO request and a new request can be made at that point. Monitoring this bit in each successive request, an Audio Q-Channel Info IOCTL for example, will tell when play mode is complete.

Bit 8, the done bit, is set by the device driver when the operation is finished.

Error codes include the following:

```
0    Write-protect violation
1    Unknown unit
2    Drive not ready
3    Unknown command
4    CRC error
5    Bad drive request structure length
6    Seek error
7    Unknown media
8    Sector not found
9    Printer out of paper
```

	A	Write fault
	B	Read fault
	C	General failure
	D	Reserved
	E	Reserved
	F	Invalid disk change

While MS-DOS reserves D as a general error code, the audio PLAY command uses it for a "PARAMETER OUT OF RANGE" error.

Command code field

The following values are valid command codes:

*	0	INIT
	1	MEDIA CHECK (block devices)
	2	BUILD BPB (block devices)
*	3	IOCTL INPUT
	4	INPUT (read)
	5	NONDESTRUCTIVE INPUT NO WAIT
	6	INPUT STATUS
*	7	INPUT FLUSH
	8	OUTPUT (write)
	9	OUTPUT WITH VERIFY
	10	OUTPUT STATUS
#	11	OUTPUT FLUSH
*	12	IOCTL OUTPUT
*	13	DEVICE OPEN
*	14	DEVICE CLOSE
	15	REMOVABLE MEDIA (block devices)
	16	OUTPUT UNTIL BUSY
*	128	READ LONG
	129	Reserved
*	130	READ LONG PREFETCH
*	131	SEEK
+	132	PLAY AUDIO
+	133	STOP AUDIO
#	134	WRITE LONG
#	135	WRITE LONG VERIFY
+	136	RESUME AUDIO

* Supported by a basic CD-ROM device driver (required)
+ Supported by an extended CD-ROM device driver (required for multimedia driver)
See also IOCTL INPUT
Supported by writable CD-ROM device drivers for authoring systems

Unsupported or illegal commands will set the error bit and return the error code for *Unknown Command.* This includes command codes 1, 2, 4, 5, 6, 8, 9, 10, 15, 16, and 129; and 11, 134 and 135 for systems that do not support writing.

If, in the time since the last request to that device driver unit, the media has changed, the device driver will return the error code for invalid disk change and set the error bit. MSCDEX.EXE will then decide whether to retry the request or abort it.

The minimal CD-ROM device driver will read cooked Mode 1 data sectors using HSG addressing mode and return appropriate values for the IOCTL calls. Most other features enhance

performance or add useful capabilities.

INIT

Command code = 0

ES:BX = INIT

```
INIT    DB    13 dup (0)    ; Request header
        DB    0             ; Number of units (must be 0)
        DD    ?             ; End address
        DD    ?             ; Ptr to BPB array
        DB    0             ; Block device number
```

This call is made only once, when the device is installed. INIT and a single IOCTL call for the device header address are the only device driver calls that come directly from MS-DOS. Because the INIT function is called from MS-DOS, the number of units returned is 0, as for normal MS-DOS character devices. MSCDEX.EXE will get the number of units supported from the device header.

The device must return the END ADDRESS, which is a DWORD pointer to the end of the portion of the device driver to remain resident. Code and data following the pointer is used for initialization and then discarded. (The device driver should discard this code to reduce its resident size.) If there are multiple device drivers in a single file, the ending address returned by the last INIT call will be the one that MS-DOS uses, but it is recommended that all the device drivers in the file return the same address. The code to remain resident for all the devices in a single file should be grouped together low in memory with the initialization code for all devices following it in memory.

The pointer to BPB array points to the character after the "=" on the line in CONFIG.SYS that caused this device driver to be loaded. This data is read-only and allows the device driver to scan the invocation line for parameters. This line is terminated by a carriage return or a line feed.

During initialization, the device driver must set the device name field in the device header to the argument provided on the invocation line in CONFIG.SYS. The device driver must also check that the *device_name* command line argument is a legal 8-character filename and pad it out to 8 characters with spaces (20H) when copying it to the device name field.

The block device number and number of units are both 0 for character devices.

The device driver should return the error code for *Unknown Command* for subsequent calls to INIT.

READ (IOCTL Input)

Command code = 3

ES:BX = IOCTLI

```
IOCTLI  DB    13 dup (0)    ; Request header
        DB    0             ; Media descriptor byte from BPB
        DD    ?             ; Transfer address
        DW    ?             ; Number of bytes to transfer
        DW    0             ; Starting sector number
        DD    0             ; DWORD ptr to requested vol
                            ; ID if error 0FH
```

The media descriptor byte, starting sector number, and volume ID fields are all 0.

The transfer address points to a control block that is used to communicate with the device driver. (The types of control blocks are listed in the following table.) The first byte of the control block determines the request that is being made. If the command code is reserved or the function not supported, then the device driver will return the error code for *Unknown Command*. If, for some reason, the device driver is not able to process the request at that time, it will return the error code for *Drive Not Ready*.

	Code	Number of bytes to transfer	Function
	0	5	Return Address of Device Header
*	1	6	Location of Head
	2	?	Reserved
	3	?	Error Statistics
*	4	9	Audio Channel Info
	5	130	Read Drive Bytes
	6	5	Device Status
	7	4	Return Sector Size
	8	5	Return Volume Size
	9	2	Media Changed
*	10	7	Audio Disk Info
*	11	7	Audio Track Info
*	12	11	Audio Q-Channel Info
*	13	13	Audio Sub-Channel Info
*	14	11	UPC Code
*	15	11	Audio Status Info
	16-255	?	Reserved

* Required only for extended/multimedia drivers.

- Return Address of Device Header

```
Raddr   DB    0             ; Control block code
        DD    ?             ; Address of device header
```

The device driver will fill the 4-byte field with the address of its device header. This is used by MSCDEX.EXE to locate the device driver's strategy and interrupt routines.

- Location of Head

```
LocHead   DB    1             ; Control block code
          DB    ?             ; Addressing mode
          DD    ?             ; Location of drive head
```

The device driver will return a 4-byte address that indicates where the head is located. The value will be interpreted based on the addressing mode. (See function READ LONG for more information about addressing modes.)

NOTE: the drive could provide this information by monitoring the Q-channel on the disc.

- Error Statistics

```
ErrStat   DB    3             ; Control block code
          DB    N dup (?)     ; Error statistics
```

The format of the Error Statistics is not defined.

- Audio Channel Info

```
AudInfo   DB    4             ; Control block code
          DB    ?             ; Input channel (0, 1, 2, or 3) for output
                              ; channel 0
          DB    ?             ; Volume control (0 - 0xff) for output channel 0
          DB    ?             ; Input channel (0, 1, 2, or 3) for output
                              ; channel 1
          DB    ?             ; Volume control (0 - 0xff) for output channel 1
          DB    ?             ; Input channel (0, 1, 2, or 3) for output
                              ; channel 2
          DB    ?             ; Volume control (0 - 0xff) for output channel 2
          DB    ?             ; Input channel (0, 1, 2, or 3) for output
                              ; channel 3
          DB    ?             ; Volume control (0 - 0xff) for output channel 3
```

This function returns the present settings of the audio channel control set with the Audio Channel Control Ioctl Write function. The default settings for the audio channel control are for each input channel to be assigned to its corresponding output channel (0 to 0, 1 to 1, etc.) and for the volume control on each channel is set at 0xff.

- Read Drive Bytes

```
DrvBytes  DB    5             ; Control block code
          DB    ?             ; Number bytes read
          DB    128 dup (?)   ; Read buffer
```

Data returned from the CD-ROM drive itself can be read using this function. The number-bytes-read field returns the length of the number of bytes read, which will not exceed 128 per call. If more than this needs to be returned, the application should repeat the call until the number returned is less than 128.

The function and content of these bytes are entirely device and device driver dependent. This function is provided to allow access to device-specific features that are not addressed under any other portion of the device driver spec.

- Device Status

```
DevStat     DB      6               ; Control block code
            DD      ?               ; Device parameters
```

The device driver will return a 32-bit value. Bit 0 is the least significant bit. The bits are interpreted as follows:

Bit 0	0	Door closed
	1	Door open
Bit 1	0	Door locked
	1	Door unlocked
Bit 2	0	Supports only cooked reading
	1	Supports cooked and raw reading
Bit 3	0	Read only
	1	Read/write
Bit 4	0	Data read only
	1	Data read and plays audio/video tracks
Bit 5	0	No interleaving
	1	Supports ISO-9660 interleaving using interleave size and skip factor
Bit 6	0	Reserved
Bit 7	0	No prefetching
	1	Supports prefetching requests
Bit 8	0	No audio channel manipulation
	1	Supports audio channel manipulation
Bit 9	0	Supports HSG addressing mode
	1	Supports HSG and Redbook addressing modes
Bit 10	0	Reserved
Bit 11	0	Disc is present in drive
	1	No disc is present in drive
Bit 12	0	Doesn't support R-W sub-channels
	1	Supports R-W sub-channels
Bit 13-31	0	Reserved (all 0)

- Return Sector Size

```
SectSize    DB      7               ; Control block code
            DB      ?               ; Read mode
```

```
            DW      ?               ; Sector size
```

The device driver will return the sector size of the device given the read mode provided. In the case of CD-ROM, the value returned for cooked is 2048, and the return value for raw is 2352.

- Return Volume Size

```
VolSize     DB      8               ; Control block code
            DD      ?               ; Volume size
```

The device driver will return the number of sectors on the device. The size returned is the address of the lead-out track in the TOC converted to a binary value according to FRAME + (SEC * 75) + (MIN * 60 * 75). A disc with a lead out track starting at 31:14.63 would return a volume size of 140613. The address of the lead-out track is assumed to point to the first sector following the last addressable sector recorded on the disc.

- Media Changed

```
MedChng     DB      9               ; Control block code
            DB      ?               ; Media byte
```

The normal media check function (command code 1) is not performed on character devices and contains additional semantics that are not needed for CD-ROM device drivers. This is why there is an IOCTL request for this function.

When the device driver receives a call to see if the media has changed on that subunit since the last call to this IOCTL, it will return one of the following values:

1	Media not changed
0	Don't know if changed
-1 (0FFh)	Media changed

If the driver can assure that the media has not been changed (through a door-lock or other interlock mechanism), performance is enhanced because MSCDEX.EXE does not need to reread the Volume Descriptor and invalidate in-memory buffers for each directory access. For drives that do not report if the media has changed, CD-ROM device drivers can utilize the same solution that has been applied to floppy disks. In some floppy-disk device drivers, if the MEDIA CHECK occurs within 2 seconds of a floppy-disk access, the driver reports "Media not changed." It is highly recommended though that drives be able to detect and report media changes.

If the drive can enforce a door lock mechanism so that the device driver is notified when the door lock has been unlocked or the device driver is requested to do so by MSCDEX.EXE, then to improve performance, the driver could return that the media has not changed without bothering to communicate with the physical device.

If the media has not been changed, MSCDEX.EXE will proceed with the disk access. If the value returned is "Don't know," or "Media changed," then MSCDEX.EXE will check to see if the disk has changed. It will continue if it has not, and reinitialize what it knows about the disk if it has.

Multimedia drivers assume that after a MEDIA CHECK cached TOC (Table of Contents) information is correct.

- Audio Disk Info

```
DiskInfo    DB    10          ; Control block code
            DB    ?           ; Lowest track number
            DB    ?           ; Highest track number
            DD    ?           ; Starting point of the lead-out track
```

This function returns TOC (Table of Contents) information from the Q-Channel in the lead-in track indicating what the first and last track numbers are and the Redbook address for the lead-out track (PMIN/PSEC/PFRAME when POINT = A2). The first and last track numbers are binary values and not BCD. It is recommended that the information for Audio Disk Info and Audio Track Info should be read by the drive when the disc is initialized and made accessible to the driver so that when these functions are called, the drive or driver do not have to interrupt audio play to read them from the TOC. If the TOC is not made available to the driver and the driver must obtain the information itself from the lead-in track, the driver should read and and attempt to cache the disk and track information during the Audio Disk Info command and invalidate this information only if the media changes. Note that the lowest and highest track numbers do not include the lead-in or lead-out tracks.

- Audio Track Info

```
TnoInfo     DB    11          ; Control block code
            DB    ?           ; Track number
            DD    ?           ; Starting point of the track
            DB    ?           ; Track control information
```

This function takes a binary track number, from within the range specified by the lowest and highest track number given by the Audio Disk Info command, and returns the Redbook address for the starting point of the track and the track control information for that track. The track control information byte corresponds to the byte in the TOC in the lead-in track containing the two 4-bit fields for CONTROL and ADR in the entry for that track. The CONTROL information is in the most significant 4 bits and the ADR information is in the lower 4 bits. The track control information is encoded as follows:

```
00x00000    - 2 audio channels without pre-emphasis
00x10000    - 2 audio channels with pre-emphasis
10x00000    - 4 audio channels without pre-emphasis
10x10000    - 4 audio channels with pre-emphasis
01x00000    - data track
01x10000    - reserved
11xx0000    - reserved
```

xx0x0000 - digital copy prohibited
xx1x0000 - digital copy permitted

- Audio Q-Channel Info

```
QInfo   DB   12      ; Control block code
        DB   ?       ; CONTROL and ADR byte
        DB   ?       ; Track number (TNO)
        DB   ?       ; (POINT) or Index (X)
                     ; Running time within a track
        DB   ?       ; (MIN)
        DB   ?       ; (SEC)
        DB   ?       ; (FRAME)
        DB   ?       ; (ZERO)
                     ; Running time on the disk
        DB   ?       ; (AMIN) or (PMIN)
        DB   ?       ; (ASEC) or (PSEC)
        DB   ?       ; (AFRAME) or (PFRAME)
```

This function reads and returns the most up to date Q-channel address presently available. It must not interrupt the present status of the drive as one of its intended purposes is to monitor the location of the read head while playing audio tracks. This function should return valid information even when no audio tracks are being played and the head is stationary. The fields returned correspond to the data that is stored in the Q-channel as described in the Redbook. The values in MIN-SEC-FRAME, AMIN-ASEC-AFRAME and PMIN-PSEC-PFRAME are converted by the driver from BCD to binary so that minutes range from 0 to 59+, seconds from 0 to 59, and frames from 0 to 74. The Control and ADR byte, TNO, and POINT/Index bytes are always passed through as they appear on the disc and are not converted. If the drive returns Q-channel information when ADR is not equal to 1, then when ADR is not equal to 1 all ten bytes of information are passed through unmodified to the caller.

Audio Sub-Channel Info

```
        DB   13      ; Control block code
        DD   ?       ; Starting sector address
        DD   ?       ; Transfer address
        DD   ?       ; Number of sectors to read
```

This function takes a Redbook address of a particular sector (also known as a block or frame) and copies the sub-channel information for each sector requested to the transfer address. If multiple sectors are requested, they are copied sequentially. Each sector contains 96 bytes of sub-channel information. These bytes do not include the two sync patterns (S0 and S1) that head the subcoding block. They contain only the subcoding symbols--each with one bit of information for the eight different channels (P-W) that follow them. In this byte, the 2 most significant bits that represent channels P and Q are undefined, the next most significant bit (bit 6) represents channel R, and the least significant bit (bit 1) represents channel W.

For the sub-channel data to be generally useful it *must* be provided during an AUDIO PLAY

operation without interrupting it. Therefore, when the requested sectors lie within the current play request they should be provided in real-time. This requires data buffering (in the drive or by the driver) for at least one sector of sub-channel information. This is because an application using sub-channel information will send an AUDIO PLAY command, follow it up with successive and contiguous sub-channel information commands, and expect to get that data starting with the first sector of the play request. It is recommended that the buffer handle up to 32 sectors, to give an application time to process sub-channel data between requests.

Since implementation of this command is optional, bit 12 in the device status long word should indicate whether sub-channel support is available. The driver will return an error code of *Unknown Command* if it is not supported.

The method of data error detection and correction for R-W channels is specified in the Redbook standard, and if not implemented by the drive, must be provided in the driver software.

- UPC Code

```
UPCCode  DB     14              ; Control block code
         DB     ?               ; CONTROL and ADR byte
         DB     7 dup (?)       ; UPC/EAN code
                                ; (last 4 bits are zero; the low-order
                                ; nibble of byte 7)
         DB     ?               ; Zero
         DB     ?               ; Aframe
```

This function returns the UPC/EAN (Universal Product Code - BAR coding) for the disc. This information is stored as a mode-2 (ADR=2) Q-channel entry. The UPC code is 13 successive BCD digits (4 bits each) followed by 12 bits of zero. The last byte is the continuation of FRAME in mode-1 though in the lead-in track (TNO=0) this byte is zero. If the CONTROL/ADR byte is zero or if the 13 digits of UPC code are all zero, then either no catalog number was encoded on the disc or it was missed by the device driver. If the command is supported but the disc does not have a UPC Code recorded, then the driver will return an error code of *Sector not Found*. If the command is not supported, then the driver will return an error code of *Unknown Command.*(ISRC is currently unsupported.)

- Audio Status Info

```
AudStat  DB     15      ; Control block code
         DW     ?       ; Audio status bits
                        ; Bit 0 is Audio Paused bit
                        ; Bits 1-15 are reserved
         DD     ?       ; Starting location of last Play or for next Resume
         DD     ?       ; Ending location for last Play or for next Resume
```

The Audio Paused bit and Starting and Ending locations are those referred to in the RESUME command. These are recorded in Redbook addressing mode.

WRITE (IOCTL OUTPUT)

Command code = 12

ES:BX = IOCTLO

```
IOCTLO   DB    13 dup (0)    ; Request header
         DB    0             ; Media descriptor byte from BPB
         DD    ?             ; Transfer address
         DW    ?             ; Number of bytes to transfer
         DW    0             ; Starting sector number
         DD    0             ; DWORD ptr to requested vol
                             ; ID if error 0FH
```

The media descriptor byte, starting sector number, and volume ID fields are all 0.

The transfer address points to a control block that is used to communicate with the device driver. The first byte of the control block determines the request that is being made. The Length of Block is the number of bytes to transfer. With the exception of Reset Drive and Write Device Control String, the action of these request may be verified with the READ IOCTL Audio Channel Info or Device Status functions. If the command is not supported, then the driver will return an error code of *Unknown Command.*

Code	Length of Block	Function
0	1	Eject Disk
1	2	Lock/Unlock Door
2	1	Reset Drive
3	9	Audio Channel Control
4	?	Write Device Control String
5	1	Close Tray
6-255	?	Reserved

- Eject Disk

```
Eject          DB    0          ; Control block code
```

The device driver will unlock the drive and eject the CD-ROM disk from the drive unit. The door will report as being open until the user has inserted a disk into the drive unit and closed the door. The status bit for door open can be monitored to determine when a disk has been reinserted.

- Lock/Unlock Door

```
LockDoor    DB      1               ; Control block code
            DB      ?               ; Lock function
```

When this function is received, the device driver will ask the CD-ROM drive to unlock or lock the door. If lock function is 0, the device driver will unlock the door. If lock function is 1, it will lock the door.

- Reset Drive

```
ResetDrv    DB      2               ; Control block code
```

This function directs the device driver to reset and reinitialize the drive.

- Audio Channel Control

```
AudInfo     DB      3               ; Control block code
            DB      ?               ; Input channel (0, 1, 2, or 3) for output
                                    ; channel 0
            DB      ?               ; Volume control (0 - 0xff) for output channel 0
            DB      ?               ; Input channel (0, 1, 2, or 3) for output
                                    ; channel 1
            DB      ?               ; Volume control (0 - 0xff) for output channel 1
            DB      ?               ; Input channel (0, 1, 2, or 3) for output
                                    ; channel 2
            DB      ?               ; Volume control (0 - 0xff) for output channel 2
            DB      ?               ; Input channel (0, 1, 2, or 3) for output
                                    ; channel 3
            DB      ?               ; Volume control (0 - 0xff) for output channel 3
```

This function is intended to provide playback control of audio information on the disk. It allows input channels on the CD-ROM to be assigned to specific output speaker connections. The purpose of this function is to allow two independent channels to be recorded - in different languages for example - and to play back only one of them at a time or to be able to manipulate an audio signal so that the source appears to move - to make a sound seem to move from left to right for example.

Output channel 0 is the left channel, 1 is right, 2 is left prime, and 3 is right prime. The Redbook specification allows for 4 audio channels. The two "prime" channels (2 and 3) extend stereo to quadrophonic stereo.

An audio volume setting of 0 means off. Drives that don't support 4 output audio channels may ignore output to channels 2 and 3. Assignment of input channels 2 and 3 to output channels 0 and 1 may be treated as though the volume control for that channel is 0.

Drives that do not support variable audio control will treat a setting of 0 as off and 1-0xff as on. Drives that support less than 256 volume settings will do their best to break up the 256 settings among the settings they can support. For example, if there are 16 settings supported, then the first setting will cover 0x01-0x10, the second 0x11-0x20...the sixteenth 0xf1-0xff. Drives that can't play a single channel in both must play only that one channel and try to suppress the other if possible. Drives that can't swap channels should play the channel that was moved in its normal channel.

- Write Device Control String

```
DrvBytes    DB      4               ; Control block code
            DB      N dup (?)       ; Write buffer
```

This function is provided to allow programs to talk directly to the CD-ROM drive. All remaining bytes are sent uninterpreted to the drive unit.

The function and content of these bytes are entirely device and device driver dependent. This function is provided to allow access to device-specific features that are not addressed under any other portion of the device driver spec.

- Close Tray

```
CloseTray   DB      5               ; Control block code
```

This command is the logical complement to the Eject Disk command. This command will instruct drives that can do so to close the door or tray.

READ LONG

Command code = 128

ES:BX = ReadL

```
ReadL       DB      13 dup (0)      ; Request header
            DB      ?               ; Addressing mode
            DD      ?               ; Transfer address
            DW      ?               ; Number of sectors to read
            DD      ?               ; Starting sector number
            DB      ?               ; Data read mode
            DB      ?               ; Interleave size
            DB      ?               ; Interleave skip factor
```

The request block is different from a normal character device READ to accommodate the larger

size and different characteristics of CD-ROM devices.

The media descriptor byte, which has no meaning for character devices, is now the addressing mode field. The following values are recognized addressing modes:

0	HSG addressing mode
1	Redbook addressing mode
2-255	Reserved

The default addressing mode is the HSG addressing mode. Long (DWORD) address values are treated as logical block numbers, as defined by the High Sierra proposal. When Redbook addressing mode is on, all disk addresses are interpreted as Minute/Second/Frame addresses, according to the Philips/Sony Redbook standard. Each of these fields is 1 byte. The least significant byte of the address field contains data for frames, the second least significant byte contains data for seconds, the third least significant byte contains data for minutes, and the most significant byte of the 4-byte field is unused. These values are represented in binary rather than in BCD format. For example, if we are referencing the sector addressed by minute 36, second 24, frame 12, the hex long value for this would be 0x0024180C. This 4 byte value is recorded in the starting sector number field with the least significant byte first and most significant byte last.

The relationship between High Sierra sectors and Redbook frames is described by the equation:

Sector = Minute * 60 * 75 + Second * 75 + Frame - 150

The byte/sector count field becomes the number of sectors to read and the starting sector number expands from one word to two, which means we can address up to 4 giga-sectors (over 8 terabytes). The DWORD ptr for requested volume ID is eliminated and MSCDEX.EXE will keep track of what volume is needed.

MSCDEX.EXE handles buffering requests, but performance may be improved if the device driver reads ahead or uses a sector caching scheme, given the slow seek times of CD-ROM drives. The operating system will use the prefetch function when it can to give hints to the driver.

The data read mode field will be one of the following:

0	Cooked mode
1	Raw mode
2-255	Reserved

Cooked mode is the default mode in which the hardware typically handles the EDC/ECC and the device driver returns 2048 bytes of data per sector read. When raw mode is set, the driver will return all 2352 bytes of user data, including any EDC/ECC present independent of the actual sector mode (Mode 2 Form 1 vs. Mode 2 Form 2). User programs will have to consider this and allow enough room for buffer space when reading in raw mode as each sector returned will take up 2352 bytes of space. Drives that cannot return all 2352 bytes will return what they can and leave blank what they cannot. For example, drives that can return all 2336 bytes except the 16 byte header will leave a space in the first 16 bytes where the header would go so that the sectors align on 2352 byte boundaries. Drivers should do what they can to return as much of the user data per sector as possible.

The two interleave parameters are for drivers that support interleaved reading. If the driver does not support interleaving, these fields are both ignored. If it does, interleave size is the number of consecutive logical blocks or sectors that are stored sequentially, and the interleave skip factor is the number of consecutive logical blocks or sectors that separate portions of the interleaved file.

READ LONG PREFETCH

Command code = 130

ES:BX = ReadLPre

```
ReadLPre  DB    13 dup (0)      ; Request header
          DB    ?               ; Addressing mode
          DD    0               ; Transfer address
          DW    ?               ; Number of sectors to read
          DD    ?               ; Starting sector number
          DB    ?               ; Read mode
          DB    ?               ; Interleave size
          DB    ?               ; Interleave skip factor
```

This function is similar in form to READ LONG, but control returns immediately to the requesting process. The device driver is not obligated to read in the requested sectors but can instead consider the request for these sectors as hints from the operating system that they are likely to be needed. It is recommended that at a minimum, the driver seek to the location provided. The attribute in the device status for prefetching is used to distinguish drivers that do more than just seek to the given location. The requests are low priority and preemptible by other requests for service. A READ LONG PREFETCH with 0 number of sectors to read should be treated as an advisory seek, and the driver can, if it is not busy, move the head to the starting sector. Since prefetching requests are advisory, there will be no functional difference between a device driver that supports prefetching from one that does not, except in terms of performance. The transfer address is not applicable for this call as the driver is not meant to transfer any data into the user address space.

SEEK

Command code = 131

ES:BX = SeekReq

```
SeekReq   DB    13 dup (0)      ; Request header
          DB    ?               ; Addressing mode
          DD    0               ; Transfer address
          DW    0               ; Number of sectors to read
          DD    ?               ; Starting sector number
```

Control returns immediately to the caller without blocking and waiting for the seek to be completed. The number of sectors to be read and the transfer address are ignored. SEEK is used to relocate the head in order to begin playing audio or video tracks, or in anticipation of reading in a particular region on the disk. Further requests for disk activity will wait until the given SEEK is completed. This seek is not advisory and the head must move to the desired location. If the address is not within the range of the disc, return a "SEEK ERROR".

PLAY AUDIO

Command code = 132

ES:BX = PlayReq

```
PlayReq   DB    13 dup (0)      ; Request header
          DB    ?               ; Addressing mode
          DD    ?               ; Starting sector number
          DD    ?               ; Number of sectors to read
```

This function will cause the driver to play the selected audio tracks until the requested sectors have been exhausted or until play is interrupted with an AUDIO STOP request. Control returns immediately to the caller. The busy bit in the status word will indicate if the drive is presently playing audio and when the play request is completed. The busy bit in the status word of the request header, as well as the paused bit and starting and ending locations returned by the Audio Status Info IOCTL, will be updated. If the drive does not support playing audio this request is ignored and it should return the error code for *Unknown Command.* If a data address is supplied, a "READ FAULT" error should be returned and the command should fail. If the sum of the starting sector and number of sectors to read exceeds the range of the disc, or there are intermediate data tracks, return the error code for *PARAMETER OUT OF RANGE.*

STOP AUDIO

Command code = 133

ES:BX = StopPlayReq

```
StopPlayReq     DB      13 dup (0)      ; Request header
```

This function is included to pause the drive unit when it is currently in play mode, or to reset the starting and ending locations when in paused mode. If applicable, at the next stopping point it reaches the drive will discontinue playing, and process the next request. The busy bit in the status word of the request header, as well as the paused bit and starting location returned by the Audio Status Info IOCTL, will be updated. If the drive does not support playing audio, it should ignore this request and return the error code for *Unknown Command.*

RESUME AUDIO

Command code = 136

ES:BX = ResumeReq

```
ResumeReq       DB      13 dup (0)      ; Request header
```

This function is used to resume playing audio tracks when a previous PLAY AUDIO call has been paused with a STOP AUDIO command. It will resume playing from the starting location indicated by the Audio Status Info IOCTL. It will modify the Audio Paused bit returned by the Audio Status Info IOCTL, and the busy bit in the status word of the request header. If the drive does not support playing audio, it should ignore this request and return the error code for *Unknown Command.*

In the following example the playing flag corresponds to the state reported by the busy bit in the status word of the request header when the drive is in audio play mode. The paused flag corresponds to the Audio Paused bit and last_startloc and last_endloc correspond to the starting and ending location reported in the Audio Status Info IOCTL.

```
// upon RESET,  NEW DISC,  or PLAY/RESUME COMPLETED the state
//should be updated:

playing              = FALSE;
paused               = FALSE;
last_startloc        = 0;
last_endloc          = 0;
PLAY_AUDIO( startloc, endloc )
 {if (playing) return error;

// Play overrides pause else if ( play( startloc, endloc ) != SUCCESSFUL ) return error;
      else
{playing = TRUE;
paused = FALSE;
last_startloc = startloc
last_endloc = endloc
return no error;}}

STOP_AUDIO( void )
{if ( playing )
{last_startloc = present q-channel location
playing = FALSE;
paused = TRUE;
stop();
return no error;}
      else


{playing = FALSE;
paused = FALSE;
last_startloc = 0;
last_endloc = 0;
return no error;}}

RESUME_AUDIO()
{if ( paused )
{if ( play( last_startloc, last_endloc ) != SUCCESSFUL )
return error;
      else
{playing = TRUE;
paused = FALSE;
return no error;}}
```

```
    else
return error;}
```

WRITE LONG

Command code = 134

ES:BX = WriteL

```
WriteL    DB    (dup 13 0)    ; Request header
          DB    ?             ; Addressing mode
          DD    ?             ; Transfer address
          DW    ?             ; Number of sectors to write
          DD    ?             ; Starting sector number
          DB    ?             ; Write mode
          DB    ?             ; Interleave size
          DB    ?             ; Interleave skip factor
```

The device will copy the data at the transfer address to the CD RAM device at the sector indicated. The media must be writable for this function to work. Data is written sector by sector, depending on the current write mode and the interleave parameters. The following values are recognized as valid write modes:

0	Mode 0
1	Mode 1
2	Mode 2 Form 1
3	Mode 2 Form 2
4-255	Reserved

Writing in Mode 1 is the default and must be supported. If the device driver supports the other modes, then they can be used. If Mode 0 is used, the transfer address is ignored and all sectors are written with zeroes. If the current write mode is Mode 1 or Mode 2 Form 1, each sector will consist of 2048 bytes of data located sequentially at the transfer address. If the write mode is Mode 2 Form 2, the device driver will expect 2336 bytes of data per sector at the transfer address.

WRITE LONG VERIFY

Command code = 135

ES:BX = WriteLV

```
WriteLV   DB    (dup 13 0)    ; Request header
          DB    ?             ; Addressing mode
          DD    ?             ; Transfer address
          DW    ?             ; Number of sectors to write
          DD    ?             ; Starting sector number
          DB    ?             ; Write mode
          DB    ?             ; Interleave size
```

```
                DB      ?               ; Interleave skip factor
```

This function is identical to WRITE LONG, with the addition that the device driver is responsible for verifying the data written to the device.

INPUT FLUSH

Command code = 7

ES:BX = FlushI

```
FlushI          DB      13 dup (0)      ; Request header
```

Requests that the device driver free all input buffers and clear any pending requests.

OUTPUT FLUSH

Command code = 11

ES:BX = FlushO

```
FlushO          DB      (dup 13 0)      ; Request header
```

Requests that the device driver write all unwritten buffers to the disk.

DEVICE OPEN
DEVICE CLOSE

Command code = 13,14

ES:BX = DevOpen, DevClose

```
DevOpen   DB      13 dup (0)      ; Request header
```

Used by the device driver to monitor how many different callers are currently using the CD-ROM device driver. All new device drivers should support these calls even if nothing is done with the information.

End.

CD-ROMifying Your Software

CD-ROM is the first of what will probably be several alien file structures that will start appearing in the MS-DOS world primarily with the introduction of installable file systems under newer versions of MS-DOS. The following will attempt to outline some guidelines for writing software that will help in porting your software to these new file systems and for CD-ROM specifically.

Choice of Filename Characters

On the first Microsoft Test CD-ROM disc, the Codeview demo failed because certain filename characters that were legal on MS-DOS were not allowed according to the High Sierra file format. When the software looked for file 'S1.@@@', it wasn't found because the character '@' is illegal for High Sierra filenames and during High Sierra premastering, the file was renamed 'S1'.

Valid High Sierra filename characters are the letters 'A' through 'Z', the digits '0' through '9', and the underscore character '_'. All other characters are invalid. Note that the letters 'a' through 'z' are not included so that High Sierra file names are not case sensitive. Under MS-DOS, filenames are mapped to upper case before they are looked up so this is typically not a problem. When choosing file name characters, keep in mind the restrictions of the file structure format and the operating systems your media may be targeted towards.

Depth of Path

The High Sierra format allows for pathnames to be up to 8 levels deep. It's possible to create a path on MS-DOS that is deeper than that but you won't be able to transfer it to a CD-ROM. The following example shows valid and invalid pathnames:

```
\one\two\three\four\five\six\seven\eight\file.txt          /* Valid  */
\one\two\three\four\five\six\seven\eight\nine\file.txt     /* Invalid */
```

Length of Path

The High Sierra format allows for the entire pathname to be a maximum of 255 characters. Since MS-DOS imposes a limit far lower than this, it should not present a problem. The MS-DOS call to connect to a sub-directory is limited to a directory string of 64 characters. The length of path restriction is more a concern for Xenix/Unix than MS-DOS.

Amusingly enough, for MS-DOS versions 2.X and 3.X, the MS-DOS call to create a sub-directory allows a directory string greater than 64 characters which allows you to create sub-directories that you cannot connect to.

Unfortunately, a CD-ROM may potentially contain a pathname that is much larger than 64 characters long. This is not a concern here but is discussed in a related sectionMS-DOSifying your CD-ROM. As a rule, try to keep the length of your longest path less than 64 characters and you should be pretty safe.

Read-only Attributes

Even though most people understand that CD-ROM discs are read-only, there's still a lot of software written that assumes the current disk is always writable. For example, the Microsoft Multiplan Demo assumes that it can create and write temporary files to the presently connected drive.

To avoid this problem, try to provide another means of letting the user specify where temporary files can be created. Many applications check the environment for the variables TMP or TEMP which contain the pathname to use when creating temporary files. Most people understand this convention now. (If the TEMP directory is located on a ram-drive, the user gains an added benefit of improving the speed of accessing the information in the temporary file.) If the environment

variable is not set, then the application can fall back on the assumption that the media is writable or ask where temporary files should be kept.

As a rule, for both temporary and permanent files, if a file creation error occurs, allow the user to re-specify the pathname used so that he can work around the error. The last thing that should happen is to lose work because the user was not allowed to store his work data in a valid place.

Data and Disk Format

Don't depend on the format of data on the disk. For example, CD-ROM's do not have a FAT so an application should not rely on finding one. Do not talk to any media at a physical level (reading/writing sectors) unless you expect to be media dependent (such as CHKDSK or FORMAT). MS-DOS INT 21h calls should provide everything you need to access the file contents and attributes.

Small Directories

For performance, try to keep directory sizes smaller than about 40 files. Much beyond this the directory files use more than one 2048 byte sector. Typically this is not a problem unless the number of sector buffers specified when MSCDEX is started is small and the directory uses multiple sectors. In this case, there is a possibility of software thrashing badly when it searches a directory if it must reload sectors read previously to find an entry.

For certain pathological programs, such as certain implementations of the Xenix utility FIND, the penalty is about 1 second per directory sector that you have to scan to get to the next entry. If the directory is large, say 8 sectors, the time for FIND to scan that one directory could potentially take a half hour for something that would take less than a second if all the entries fit in the cache.

The solution for this problem is to make sure that MSCDEX never throws out of the cache what it will need next. This is accomplished by growing the cache (very easysimply change the parameter to MSCDEX) and to make sure that the largest object that goes through the cache will not clear it out. There is a balance between having too many directories and too many files in a few directories, but the balance is heavily weighted towards many small to medium sized directories. Keep this in mind when laying out your files.

Since the penalty for using a file in the lowest subdirectory instead of the root directory is virtually nil and as more directories don't cost much, it's a good idea to break up large directories into several smaller ones. This will help avoid problems of flushing the disc sector cache. Try to keep related files close togetherboth in location on the CD-ROM and in the same directories. Keeping related files close together will reduce seek time when accessing them concurrently. Keeping them in the same directory will help prevent swapping out directory sectors.

Updating CD-ROM Databases and Software

Many people are interested in providing updates to files contained on a CD-ROM disc. They would like to create a directory on their hard disk will all updated files in them and have the CD-ROM Extensions look there first before searching the CD-ROM. Unfortunately, by the time the Extensions gets the request, it is very difficult for it to look for updates on the hard disk. So whatever alternative searching that is necessary will have to be done in the application software.

For this reason, it's a good idea to set your path so that it lists the directories on the hard disk first. Another good strategy is to copy executables to a directory on your hard disk. This lets you update them, and it lets them start faster. Also, have the application software search alternative hard disk directories for updates before it searches the CD-ROM. Storing both software updates and updated or commonly used database files on a hard disk will both speed performance and allow incremental updating.

Search Strategies

Try to avoid relying on the operating system in your search strategy. If your database is broken up

into a hierarchy and your order is imposed through the file structure by breaking up the database into many files in a tree, then accessing data in the database is typically going to require a lot of directory reading and searching.

On a hard disk, the time involved to access this type database is not large. However, on a CD-ROM the search times can add up. Opening a file can be an expensive operation simply because the system must read the directory before it can open the file. With a fast drive, seeking to a location on a CD-ROM can take about 10 milliseconds. At worst, a seek can exceed a second on some older CD-ROM drives. Some newer drives have a worst case seek time of about half a second. If you can avoid this, your application will respond faster. MSCDEX caches as many directory sectors as it can so that searching the most active directories is very quick. However, any operations that search multiple directories once through continually clears out the cache and renders its caching ineffective.

The strategy used by *Microsoft Bookshelf* was to lump the entire database into a single file and structure the indexing so that searching used a minimum of seeks. Bookshelf uses packed b-trees with each node holding as many entries as will fit into a single sector and also cache in memory as much of the root of the tree as it can.

Combining databases avoids the overhead of repeatedly opening and closing database files. Caching as much of the indexes in memory as possible allows searching of keywords to be completed typically with a single seek.

In general, identify your software bottlenecks and through judicious use of faster storage media (either memory or hard disk) you can both have large storage and respectable performance.

Portability

One advantage of the High Sierra format is data interchangeability with other operating systems. For portability, chose a set of the High Sierra features that are supported across different operating systems to be sure you can read the disc in each of them. The lowest common denominator then (this list is not complete - see also what must be done to target MS-DOS) would need a logical block size of 512 bytes, both type L and M path tables and for all fields, single volume sets, at least one Primary Volume Descriptor and terminator. Be aware that if one of your goals is data portability, you must determine what restrictions the other operating systems might impose on the High Sierra format.

End.

Documentation Contents

Function Requests Specification — Section 11

This section describes the programming interface for applications using MSCDEX.

Questions and Answers — Section 12

This section answers common question asked about device drivers and MSCDEX.

Disk Contents

MS-DOS CD-ROM Extensions Version 2.21

The following directories and files are included on this disk:

README.TXT	Release information
\BIN\MSCDEX.EXE	MSCDEX Extensions program
\BIN\HITACHIA.SYS	Hitachi MSCDEX CD-ROM device driver
\HITACHI	Sources to build Hitachi driver
\BIN\CDSPEED.EXE	Device driver test program for data transfer rate under MS-DOS
\CDSPEED	CDSPEED batch files and Excel Macros to chart CDSPEED results
\BIN\TESTDRV.EXE	Device driver test program for MSCDEX specifications
\TESTDRV	Sources to build TESTDRV
\BIN\AUDIO.EXE	Sample audio program
\AUDIO	Sample audio program
\BIN\SVD.EXE	Utility for Kanji support
\SVD	Sources for Kanji support utility

MS-DOS CD-ROM

The following directories and files are included on this disk:

README.TXT	Release information
\TECH_ART\CDROM	Source files for MSCDEX documents
\TECH_ART\CDROM\CDROMIFY.*	Source files for CD-ROMifying Your Software
\TECH_ART\CDROM\CONTENTS.*	Source files for Contents
\TECH_ART\CDROM\COVER.*	Source files for MSCDEX title pages
\TECH_ART\CDROM\DEVICE.*	Source files for Hardware-Dependent Device Driver Specification
\TECH_ART\CDROM\EXAMPLE.*	Source files for Example Device Driver
\TECH_ART\CDROM\INSTALL.*	Source files for Installation Instructions
\TECH_ART\CDROM\KANJI.*	Source files for Kanji Support.
\TECH_ART\CDROM\MCTRL.*	Source files for Function Requests Specification
\TECH_ART\CDROM\MSDOSIFY.*	Source files for MS-DOSifying Your CD-ROM
\TECH_ART\CDROM\NETINFO.*	Source files for Networking CD-ROMS
\TECH_ART\CDROM\OVERVIEW.*	Source files for Product Overview
\TECH_ART\CDROM\QNA.*	Source files for Questions and Answers
\TECH_ART\CDROM\SPEED*	Source files for Speed Tests
\TECH_ART\CDROM\TESTDRV.*	Source files for TESTDRV Test Utility

End.

Microsoft MS-DOS CD-ROM Extensions
Version 2.21

Hardware-Dependent Device Driver Specification

This document describes the CD-ROM hardware-dependent device driver and its interface with MSCDEX.EXE, the MS-DOS CD-ROM Extensions resident program. Differences between CD-ROM drives and hard- or floppy-disk drives account for the differences in this device driver specification from the normal MS-DOS block and character device driver specification. For more information on device drivers,see the MS-DOS Programmer's Reference Manual and Advanced MS-DOS Programming by Ray Duncan. (Both books are published by Microsoft Press.) For information on the terms describing for CD-ROM operating characteristics, see the IEC 908 and ISO 10149 specifications.

The MS-DOS operating system reads CONFIG.SYS and installs the device. MSCDEX.EXE performs an open system call on the device driver name in order to communicate with it and uses an IOCTL call to ask the device driver for the address of its device header. From the device header address, MSCDEX.EXE locates the device driver's interrupt and strategy routines. After that, all requests the device driver receives come directly from MSCDEX.EXE, not MS-DOS. To avoid reentrancy problems and allow MSCDEX to monitor all media changes, all other applications that wish to communicate directly with CD-ROM device drivers should do so through the Send Device Driver Request INT 2Fh function 1510h. MSCDEX.EXE interfaces with MS-DOS so that normal requests for I/O with files on a CD-ROM drive down to the MS-DOS INT 21h service layer will work just as they would for a normal MS-DOS device. For maximum compatibility between your CD-ROM device driver, MSCDEX, and other components in the system, observe the following guidelines when developing your device driver:

Avoid disabling interrupts for extended periods
CD-ROM drives and drivers that use DMA must use Windows virtual DMA device services for 386 enhanced mode operation. For information for using Windows virtual DMA device services, see the Microsoft Windows Device Driver Adaptation Guide included in the Microsoft Windows Device Development Kit.

Installation
The device driver will be installed in the same way as any other device with an entry in CONFIG.SYS. The syntax is:
DEVICE=<filename> /D:<device_name> /N:<number of drives>
The following are examples:
DEVICE=HITACHI.SYS /D:MSCD001 /D:MSCD002 DEVICE=SONY.SYS /D:MSCD003 /N:2
The arguments will be the character device names that will be used on the command line when starting MSCDEX.EXE so that it can find and communicate with the
device driver.

A device driver may support one or more physical drives or logical disks. This may be done by having multiple device headers in the device driver file (in which case it will be necessary to have more than one device_name on the command line - one for each device header; see the HITACHI.SYS example above) or through the use of subunits. Each disk handled by a device driver that supports multiple disks using subunits is addressed by the subunit field of the request header when a request is made for that disk. A device driver that supports more than one disk can share code and data instead of requiring separate device drivers for each disk. A "jukebox" CD-

ROM system would be an example of a CD-ROM device that might wish to support more than one drive or a disk pack using a single device driver.

Device drivers that use multiple subunits should use the optional switch /n:<number of drives> to say how many drives are present. If not present, the default number of drives is 1. If the driver can tell how many drives are installed without a command line switch, then this argument is not necessary. Unless there are special considerations, it is better practice to support multiple drives using subunits than to have multiple device headers in the same device driver file.

Device header
The device header is an extension to what is described in the MS-DOS Programmer's Refrence Manual.

```
DevHdr    DD     -1              ; Ptr to next driver in
                                 ; file or -1 if last driver
          DW     ?               ; Device attributes
          DW     ?               ; Device strategy entry point
          DW     ?               ; Device interrupt entry point
          DB     8 dup (?)       ; Character device name field
          DW     0               ; Reserved
          DB     0               ; Drive letter
          DB     ?               ; Number of units
```

The following are the device attributes for MSCDEX.EXE device drivers:

Bit 15	1	- Character device
Bit 14	1	- IOCTL supported
Bit 13	0	- Output 'till busy
Bit 12	0	- Reserved
Bit 11	1	- OPEN/CLOSE/RM supported
Bit 10-4	0	- Reserved
Bit 3	0	- Dev is CLOCK
Bit 2	0	- Dev is NUL
Bit 1	0	- Dev is STO
Bit 0	0	- Dev is STI

MSCDEX.EXE device drivers will be character devices that understand IOCTL calls and handle OPEN/CLOSE/RM calls. The drive letter field is a read-only field for the device driver and is initialized to 0. The field is for MSCDEX.EXE to use when it assigns the device driver to a drive letter (A = 1, B = 2...Z = 26). It should never be modified by the device driver. For drivers that support more than one unit, the drive letter will indicate the first unit, and each successive unit is assigned the next higher drive letter. For example, if the device driver has four units defined (0-3), it requires four drive letters. The position of the driver in the list of all drivers determines which units correspond to which drive letters. If driver ALPHA is the first driver in the device list, and it defines 4 units (0-3), they will be A, B, C, and D. If BETA is the second driver and defines three units (0-2), they will be E, F, and G, and so on. The theoretical limit to the number of drive letters is 63, but it should be noted that the device installation code will not allow the installation of a device if it would result in a drive letter > 'Z' (5Ah). All block device drivers present in the standard resident BIOS will be placed ahead of installable device drivers in the list.

NOTE:
It is important that one set lastdrive=<letter> in CONFIG.SYS to accommodate the additional drive letters that CD-ROM device drivers will require. The number-of-units field is set by the device driver to the number of disks that are supported. Normal character devices do not support more than one unit and MS-DOS does not expect a character device to handle more than one unit or have a nonzero subunit value in the request header. Since these device drivers are not called by MS-DOS directly, this is not a problem. Nonetheless, the number of units returned by

the device driver in the number-of-units field during the INIT call must be 0, since MS-DOS makes the INIT call and does not expect a nonzero value for a character device. MSCDEX.EXE will never see what is returned anyway, and relies on the number-of-units field in the device header.

The signature field is necessary for MSCDEX confirmation that the device driver is a valid cd-rom device driver and consists of the 4 bytes 'MSCD' followed by two ascii digits for the version which is '00' at present.
Sample device header:

```
HsgDrv
    DD      -1              ; Pointer to next device
    DW      0c800h          ; Device attributes
    DW      STRATEGY        ; Pointer to device
                            ; strategy routine
    DW      DEVINT          ; Pointer to device
                            ; interrupt routine
    DB      'MSCD003 '      ; 8-byte character
                            ; device name field
    DW      0               ; Reserved (must be zero)
    DB      0               ; Drive letter (must be zero)
    DB      1               ; Number of units supported
                            ; (one or more)
```

As with other MS-DOS device drivers, the code originates at offset 0, not 100H. The first device header will be at offset 0 of the code segment. The pointer to the next driver is a double word field (offset/segment) that is the address of the next device driver in the list, or -1 if the device header is the only one or the last in the list. The strategy and interrupt entry points are word fields and must be offsets into the same segment as the device header. The device driver is expected to overwrite the name(s) in each of its one or more device headers with the <device_name> command line arguments during its initialization.

MSCDEX.EXE will call the device driver in the following manner:

1. MSCDEX.EXE makes a far call to the strategy entry.

2. MSCDEX.EXE passes device driver information in a request header to the strategy routine.

3. MSCDEX.EXE makes a far call to the interrupt entry.

Request header
MSCDEX.EXE will call the device's strategy routine with the address of a request header in ES:BX. The format of the request header is the same as what is described in the MS-DOS Programmer's Reference Manual.

```
ReqHdr
            DB    ?             ; Length in bytes of request header
            DB    ?         ; Subunit code for minor devices
            DB    ?             ; Command code field
            DW    ?             ; Status
            DB    8 dup (?) ; Reserved
```

Status
The status word also has the same format as described in the MS-DOS Programmer's Reference Manual. It is 0 on entry and is set by the device driver.
Bit 15 - Error bit

Bit 14-10 - Reserved
Bit 9 - Busy
Bit 8 - Done
Bit 7-0 - Error code (bit 15 on)

Bit 15, the error bit, is set by the device driver if an error is detected or if an invalid request is made to the driver. The low 8 bits indicate the error code.

Bit 9, the busy bit, should be set by the device driver when the drive is in audio play mode. Device drivers should fail all requests to the physical device that require head movement when the device is playing and return the request with this bit and the error bit set and an error code. Requests that would not interrupt audio play may return without error but will also have this bit set when the drive is in audio play mode. Play mode can be terminated prematurely with a reset or STOP AUDIO request and a new request can be made at that point. Monitoring this bit in each successive request, an Audio Q-Channel Info IOCTL for example, will tell when play mode is complete.

Bit 8, the done bit, is set by the device driver when the operation is finished.

Error codes include the following:

	0	Write-protect violation
	1	Unknown unit
	2	Drive not ready
	3	Unknown command
	4	CRC error
	5	Bad drive request structure length
	6	Seek error
	7	Unknown media
	8	Sector not found
	9	Printer out of paper
	A	Write fault
	B	Read fault
	C	General failure
	D	Reserved
	E	Reserved
	F	Invalid disk change

While MS-DOS reserves D as a general error code, the audio PLAY command uses it for a "PARAMETER OUT OF RANGE" error.

Command code field

The following values are valid command codes:

*	0	INIT
	1	MEDIA CHECK (block devices)
	2	BUILD BPB (block devices)
*	3	IOCTL INPUT
	4	INPUT (read)
	5	NONDESTRUCTIVE INPUT NO WAIT
	6	INPUT STATUS
*	7	INPUT FLUSH
	8	OUTPUT (write)
	9	OUTPUT WITH VERIFY
	10	OUTPUT STATUS
#	11	OUTPUT FLUSH
*	12	IOCTL OUTPUT
*	13	DEVICE OPEN

*	14	DEVICE CLOSE
	15	REMOVABLE MEDIA (block devices)
	16	OUTPUT UNTIL BUSY
*	128	READ LONG
	129	Reserved
*	130	READ LONG PREFETCH
*	131	SEEK
+	132	PLAY AUDIO
+	133	STOP AUDIO
#	134	WRITE LONG
#	135	WRITE LONG VERIFY
+	136	RESUME AUDIO

***** Supported by a basic CD-ROM device driver (required)
+ Supported by an extended CD-ROM device driver (required for multimedia driver) See also IOCTL INPUT
# Supported by writable CD-ROM device drivers for authoring systems

Unsupported or illegal commands will set the error bit and return the error code for Unknown Command. This includes command codes 1, 2, 4, 5, 6, 8, 9, 10, 15, 16, and 129; and 11, 134 and 135 for systems that do not support writing.

If, in the time since the last request to that device driver unit, the media has changed, the device driver will return the error code for invalid disk change and set the error bit. MSCDEX.EXE will then decide whether to retry the request or abort it. The minimal CD-ROM device driver will read cooked Mode 1 data sectors using HSG addressing mode and return appropriate values for the IOCTL calls. Most other features enhance performance or add useful capabilities.

INIT
Command code = 0
ES:BX = INIT

```
INIT
    DB    13 dup (0)        ; Request header
    DB    0                 ; Number of units (must be 0)
    DD    ?                 ; End address
    DD    ?                 ; Ptr to BPB array
    DB    0                 ; Block device number
```

This call is made only once, when the device is installed. INIT and a single IOCTL call for the device header address are the only device driver calls that come directly from MS-DOS. Because the INIT function is called from MS-DOS, the number of units returned is 0, as for normal MS-DOS character devices. MSCDEX.EXE will get the number of units supported from the device header.

The device must return the END ADDRESS, which is a DWORD pointer to the end of the portion of the device driver to remain resident. Code and data following the pointer is used for initialization and then discarded. (The device driver should discard this code to reduce its resident size.) If there are multiple device drivers in a single file, the ending address returned by the last INIT call will be the one that MS-DOS uses, but it is recommended that all the device drivers in the file return the same address. The code to remain resident for all the devices in a single file should be grouped together low in memory with the initialization code for all devices following it in memory.

The pointer to BPB array points to the character after the "=" on the line in CONFIG.SYS that

caused this device driver to be loaded. This data is read-only and allows the device driver to scan the invocation line for parameters. This line is terminated by a carriage return or a line feed.

During initialization, the device driver must set the device name field in the device header to the argument provided on the invocation line in CONFIG.SYS. The device driver must also check that the device_name command line argument is a legal 8-character filename and pad it out to 8 characters with spaces (20H) when copying it to the device name field.

The block device number and number of units are both 0 for character devices.
The device driver should return the error code for Unknown Command for subsequent calls to INIT.

READ (IOCTL Input)
Command code = 3
ES:BX = IOCTLI

```
IOCTLI      DB      13 dup (0) ; Request header
            DB      0       ; Media descriptor byte from BPB
            DD      ?       ; Transfer address
            DW      ?       ; Number of bytes to transfer
            DW      0       ; Starting sector number
            DD      0       ; DWORD ptr to requested vol
                            ; ID if error 0FH
```

The media descriptor byte, starting sector number, and volume ID fields are all 0.
The transfer address points to a control block that is used to communicate with the device driver. (The control blocks are defined in the following table.) The first byte of the control block determines the request that is being made. If the command code is reserved or the function not supported, then the device driver will return the error code for Unknown Command. If, for some reason, the device driver is not able to process the request at that time, it will return the error code for Drive Not Ready.

	Code	Number of bytes to transfer	Function
	0	5	Return Address of Device Header
*	1	6	Location of Head
	2	?	Reserved
	3	?	Error Statistics
*	4	9	Audio Channel Info
	5	130	Read Drive Bytes
	6	5	Device Status
	7	4	Return Sector Size
	8	5	Return Volume Size
	9	2	Media Changed
*	10	7	Audio Disk Info
*	11	7	Audio Track Info
*	12	11	Audio Q-Channel Info
*	13	13	Audio Sub-Channel Info
*	14	11	UPC Code
*	15	11	Audio Status Info
	16-255	?	Reserved

* Required only for extended/multimedia drivers.
- Return Address of Device Header

```
Raddr       DB      0       ; Control block code
            DD      ?       ; Address of device header
```

_____

The device driver will fill the 4-byte field with the address of its device header. This is used by MSCDEX.EXE to locate the device driver's strategy and interrupt routines.

- Location of Head

_____

```
LocHead   DB    1          ; Control block code
          DB    ?          ; Addressing mode
          DD    ?          ; Location of drive head
```

_____

The device driver will return a 4-byte address that indicates where the head is located. The value will be interpreted based on the addressing mode. (See function READ LONG for more information about addressing modes.)

NOTE: the drive could provide this information by monitoring the Q-channel on the disc.

- Error Statistics

_____

```
ErrStat   DB    3          ; Control block code
          DB    N dup (?) ; Error statistics
```

_____

The format of the Error Statistics is not defined. - Audio Channel Info

_____

```
AudInfo   DB    4          ; Control block code
          DB    ?          ; Input channel (0, 1, 2, or
                           ; 3) for output
                           ; channel 0
          DB    ?          ; Volume control (0 - 0xff)
                           ; for output channel 0
          DB    ?          ; Input channel (0, 1, 2, or
                           ; 3) for output
                           ; channel 1
          DB    ?          ; Volume control (0 - 0xff)
                           ; for output channel 1
          DB    ?          ; Input channel (0, 1, 2, or
                           ; 3) for output
                           ; channel 2
          DB    ?          ; Volume control (0 - 0xff)
                           ; for output channel 2
          DB    ?          ; Input channel (0, 1, 2, or
                           ; 3) for output
                           ; channel 3
          DB    ?          ; Volume control (0 - 0xff)
                           ; for output channel 3
```

_____

This function returns the present settings of the audio channel control set with the Audio Channel Control Ioctl Write function. The default settings for the audio channel control are for each input channel to be assigned to its corresponding output channel (0 to 0, 1 to 1, etc.) and for the volume control on each channel is set at 0xff.

- Read Drive Bytes

_____

```
DrvBytes  DB    5              ; Control block code
          DB    ?              ; Number bytes read
          DB    128 dup (?)    ; Read buffer
```

_____

Data returned from the CD-ROM drive itself can be read using this function. The number-bytes-read field returns the length of the number of bytes read, which will not exceed 128 per call. If more than this needs to be returned, the application should repeat the call until the number returned is less than 128.

The function and content of these bytes are entirely device and device driver dependent. This function is provided to allow access to device-specific features that are not addressed under any other portion of the device driver spec.

- Device Status

```
DevStat     DB      6               ; Control block code
            DD      ?               ; Device parameters
```

The device driver will return a 32-bit value. Bit 0 is the least significant bit. The bits are interpreted as follows:

```
Bit 0               0       Door closed
            1       Door open
Bit 1               0       Door locked
            1       Door unlocked
Bit 2               0       Supports only cooked reading
            1       Supports cooked and raw reading
Bit 3               0       Read only
            1       Read/write
Bit 4               0       Data read only
            1       Data read and plays audio/video tracks
Bit 5               0       No interleaving
            1       Supports ISO-9660 interleaving using
                    interleave size and skip factor
Bit 6               0       Reserved
Bit 7               0       No prefetching
            1       Supports prefetching requests
Bit 8               0       No audio channel manipulation
            1       Supports audio channel manipulation
Bit 9               0       Supports HSG addressing mode
            1       Supports HSG and Redbook addressing modes
Bit 10              0       Reserved
Bit 11              0       Disc is present in drive
            1       No disc is present in drive
Bit 12              0       Doesn't support R-W sub-channels
            1       Supports R-W sub-channels
Bit 13-31 0 Reserved (all 0)
```

- Return Sector Size

```
SectSize    DB      7               ; Control block code
            DB      ?               ; Read mode
            DW      ?               ; Sector size
```

The device driver will return the sector size of the device given the read mode provided. In the case of CDROM, the value returned for cooked is 2048, and the return value for raw is 2352.

- Return Volume Size

```
VolSize     DB      8               ; Control block code
            DD      ?               ; Volume size
```

The device driver will return the number of sectors on the device. The size returned is the address of the lead-out track in the TOC converted to a binary value according to FRAME + (SEC * 75) + (MIN * 60 * 75). A disc with a lead out track starting at 31:14.63 would return a volume size of 140613. The address of the leadout track is assumed to point to the first sector following the last addressable sector recorded on the disc.

- Media Changed

```
MedChng   DB      9               ; Control block code
          DB      ?               ; Media byte
```

The normal media check function (command code 1) is not performed on character devices and contains additional semantics that are not needed for CD-ROM device drivers. This is why there is an IOCTL request for this function.

When the device driver receives a call to see if the media has changed on that subunit since the last call to this IOCTL, it will return one of the following values:

1	Media not changed
0	Don't know if changed
-1 (0FFh)	Media changed

If the driver can assure that the media has not been changed (through a door-lock or other interlock mechanism), performance is enhanced because MSCDEX.EXE does not need to reread the Volume Descriptor and invalidate in-memory buffers for each directory access. For drives that do not report if the media has changed, CD-ROM device drivers can utilize the same solution that has been applied to floppy disks. In some floppydisk device drivers, if the MEDIA CHECK occurs within 2 seconds of a floppy-disk access, the driver reports "Media not changed." It is highly recommended though that drives be able to detect and report media changes.

If the drive can enforce a door lock mechanism so that the device driver is notified when the door lock has been unlocked or the device driver is requested to do so by MSCDEX.EXE, then to improve performance, the driver could return that the media has not changed without bothering to communicate with the physical device.
If the media has not been changed, MSCDEX.EXE will proceed with the disk access. If the value returned is "Don't know," or "Media changed," then MSCDEX.EXE will check to see if the disk has changed. It will continue if it has not, and reinitialize what it knows about the disk if it has.

Multimedia drivers assume that after a MEDIA CHECK cached TOC (Table of Contents) information is correct.

- Audio Disk Info

```
DiskInfo  DB      10        ; Control block code
          DB      ?         ; Lowest track number
          DB      ?         ; Highest track number
          DD      ?         ; Starting point of the lead-out track
```

This function returns TOC (Table of Contents) information from the Q-Channel in the lead-in track indicating what the first and last track numbers are and the Redbook address for the lead-out track (PMIN/PSEC/PFRAME when POINT = A2). The first and last track numbers are binary values and not BCD. It is recommended that the information for Audio Disk Info and Audio Track Info should be read by the drive when the disc is initialized and made accessible to the driver so that when these functions are called, the drive or driver do not have to interrupt audio play to read them from the TOC. If the TOC is not made available to the driver and the driver must obtain the information itself from the lead-in track, the driver should read and and attempt to cache the disk and track information during the Audio Disk Info command and invalidate this information only if the media changes. Note that the lowest and highest track numbers do not include the lead-in or lead-out tracks.

- Audio Track Info

```
TnoInfo   DB      11        ; Control block code
          DB      ?         ; Track number
          DD      ?         ; Starting point of the track
          DB      ?         ; Track control information
```

This function takes a binary track number, from within the range specified by the lowest and highest track number given by the Audio Disk Info command, and returns the Redbook address for the starting point of the track and the track control information for that track. The track control information byte corresponds to the byte in the TOC in the lead-in track containing the two 4-bit fields for CONTROL and ADR in the entry for that track. The CONTROL information is in the most significant 4 bits and the ADR information is in the lower 4 bits. The track control information is encoded as follows:
00x00000 - 2 audio channels without pre-emphasis 00x10000 - 2 audio channels with pre-emphasis 10x00000 - 4 audio channels without pre-emphasis 10x10000 - 4 audio channels with pre-emphasis 01x00000 - data track
01x10000 - reserved
11xx0000 - reserved
xx0x0000 - digital copy prohibited xx1x0000 - digital copy permitted - Audio Q-Channel Info

```
QInfo       DB      12      ; Control block code
            DB      ?       ; CONTROL and ADR byte
            DB      ?       ; Track number (TNO)
            DB      ?       ; (POINT) or Index (X)
                            ; Running time within a track
            DB      ?       ; (MIN)
            DB      ?       ; (SEC)
            DB      ?       ; (FRAME)
            DB      ?       ; (ZERO)
                            ; Running time on the disk
            DB      ?       ; (AMIN) or (PMIN)
            DB      ?       ; (ASEC) or (PSEC)
            DB      ?       ; (AFRAME) or (PFRAME)
```

This function reads and returns the most up to date Qchannel address presently available. It must not interrupt the present status of the drive as one of its intended purposes is to monitor the location of the read head while playing audio tracks. This function should return valid information even when no audio tracks are being played and the head is stationary. The fields returned correspond to the data that is stored in the Q-channel as described in the Redbook. The values in MIN-SEC-FRAME, AMIN-ASEC-AFRAME and PMIN-PSECPFRAME are converted by the driver from BCD to binary so that minutes range from 0 to 59+, seconds from 0 to 59, and frames from 0 to 74. The Control and ADR byte, TNO, and POINT/Index bytes are always passed through as they appear on the disc and are not converted. If the drive returns Q-channel information when ADR is not equal to 1, then when ADR is not equal to 1 all ten bytes of information are passed through unmodified to the caller.

Audio Sub-Channel Info

```
SubChanInfo     DB      13      ; Control block code
                DD      ?       ; Starting sector address
                DD      ?       ; Transfer address
                DD      ?       ; Number of sectors to read
```

This function takes a Redbook address of a particular sector (also known as a block or frame) and copies the sub-channel information for each sector requested to the transfer address. If multiple sectors are requested, they are copied sequentially. Each sector contains 96 bytes of sub-channel information. These bytes do not include the two sync patterns (S0 and S1) that head the subcoding block. They contain only the subcoding symbols--each with one bit of information for the eight different channels (P-W) that follow them. In this byte, the 2 most significant bits that

represent channels P and Q are undefined, the next most significant bit (bit 6) represents channel R, and the least significant bit (bit 1) represents channel W.

For the sub-channel data to be generally useful it must be provided during an AUDIO PLAY operation without interrupting it. Therefore, when the requested sectors lie within the current play request they should be provided in real-time. This requires data buffering (in the drive or by the driver) for at least one sector of sub-channel information. This is because an application using sub-channel information will send an AUDIO PLAY command, follow it up with successive and contiguous sub-channel information commands, and expect to get that data starting with the first sector of the play request. It is recommended that the buffer handle up to 32 sectors, to give an application time to process sub-channel data between requests.

Since implementation of this command is optional, bit 12 in the device status long word should indicate whether sub-channel support is available. The driver will return an error code of Unknown Command if it is not supported.

The method of data error detection and correction for RW channels is specified in the Redbook standard, and if not implemented by the drive, must be provided in the driver software.
- UPC Code

```
_____________
UPCCode  DB     14          ; Control block code
         DB     ?                   ; CONTROL and ADR byte
         DB     7 dup (?) ; UPC/EAN code
                          ; (last 4 bits are zero; the ; low-order nibble of byte 7)
         DB     ?         ; Zero
         DB     ?         ; Aframe
_____________
```

This function returns the UPC/EAN (Universal Product Code - BAR coding) for the disc. This information is stored as a mode-2 (ADR=2) Q-channel entry. The UPC code is 13 successive BCD digits (4 bits each) followed by 12 bits of zero. The last byte is the continuation of FRAME in mode-1 though in the lead-in track (TNO=0) this byte is zero. If the CONTROL/ADR byte is zero or if the 13 digits of UPC code are all zero, then either no catalog number was encoded on the disc or it was missed by the device driver. If the command is supported but the disc does not have a UPC Code recorded, then the driver will return an error code of Sector not Found. If the command is not supported, then the driver will return an error code of Unknown Command.(ISRC is currently unsupported.)

Audio Status Info

```
_____________
AudStat  DB     15        ; Control block code
         DW     ?         ; Audio status bits
                          ; Bit 0 is Audio Paused bit
                          ; Bits 1-15 are reserved
         DD     ?         ; Starting location of last Play or
                          ; for next Resume
         DD     ?         ; Ending location for last Play or
                          ; for next Resume
_____________
```

The Audio Paused bit and Starting and Ending locations are those referred to in the RESUME command. These are recorded in Redbook addressing mode.
WRITE (IOCTL OUTPUT)
Command code = 12
ES:BX = IOCTLO

____

```
IOCTLO   DB    13 dup (0)     ; Request header
         DB    0              ; Media descriptor byte from BPB
         DD    ?              ; Transfer address
         DW    ?              ; Number of bytes to transfer
         DW    0              ; Starting sector number
         DD    0              ; DWORD ptr to requested vol
                              ; ID if error 0FH
```

The media descriptor byte, starting sector number, and volume ID fields are all 0.
The transfer address points to a control block that is used to communicate with the device driver. The first byte of the control block determines the request that is being made. The Length of Block is the number of bytes to transfer. With the exception of Reset Drive and Write Device Control String, the action of these request may be verified with the READ IOCTL Audio Channel Info or Device Status functions. If the command is not supported, then the driver will return an error code of Unknown Command.

Code	Length of Block	Function
0	1	Eject Disk
1	2	Lock/Unlock Door
2	1	Reset Drive
3	9	Audio Channel Control
4	?	Write Device Control String
5	1	Close Tray
6-255 ?		Reserved

Eject Disk

```
Eject    DB    0          ; Control block code
```

The device driver will unlock the drive and eject the CD-ROM disk from the drive unit. The door will report as being open until the user has inserted a disk into the drive unit and closed the door. The status bit for door open can be monitored to determine when a disk has been reinserted.

Lock/Unlock Door

```
LockDoor DB    1          ; Control block code
         DB    ?          ; Lock function
```

When this function is received, the device driver will ask the CD-ROM drive to unlock or lock the door. If lock function is 0, the device driver will unlock the door. If lock function is 1, it will lock the door. - Reset Drive

```
ResetDrv DB    2          ; Control block code
```

This function directs the device driver to reset and reinitialize the drive.

Audio Channel Control

```
AudInfo  DB    3          ; Control block code
         DB    ?          ; Input channel (0, 1, 2, or
                          ; 3) for output channel 0
         DB    ?          ; Volume control (0 - 0xff)
                          ; for output channel 0
         DB    ?          ; Input channel (0, 1, 2, or
                          ; 3) for output channel 1
         DB    ?          ; Volume control (0 - 0xff)
```

```
                    ; for output channel 1
        DB      ?   ; Input channel (0, 1, 2, or
                    ; 3) for output channel 2
        DB      ?   ; Volume control (0 - 0xff)
                    ; for output channel 2
        DB      ?   ; Input channel (0, 1, 2, or
                    ; 3) for output channel 3
        DB      ?   ; Volume control (0 - 0xff)
                    ; for output channel 3
```

This function is intended to provide playback control of audio information on the disk. It allows input channels on the CD-ROM to be assigned to specific output speaker connections. The purpose of this function is to allow two independent channels to be recorded - in different languages for example - and to play back only one of them at a time or to be able to manipulate an audio signal so that the source appears to move - to make a sound seem to move from left to right for example.

Output channel 0 is the left channel, 1 is right, 2 is left prime, and 3 is right prime. The Redbook specification allows for 4 audio channels. The two "prime" channels (2 and 3) extend stereo to quadrophonic stereo. An audio volume setting of 0 means off. Drives that don't support 4 output audio channels may ignore output to channels 2 and 3. Assignment of input channels 2 and 3 to output channels 0 and 1 may be treated as though the volume control for that channel is 0.

Drives that do not support variable audio control will treat a setting of 0 as off and 1-0xff as on. Drives that support less than 256 volume settings will do their best to break up the 256 settings among the settings they can support. For example, if there are 16 settings supported, then the first setting will cover 0x01-0x10, the second 0x11-0x20...the sixteenth 0xf10xff. Drives that can't play a single channel in both must play only that one channel and try to suppress the other if possible. Drives that can't swap channels should play the channel that was moved in its normal channel.
- Write Device Control String

```
DrvBytes  DB      4         ; Control block code
          DB      N dup (?) ; Write buffer
```

This function is provided to allow programs to talk directly to the CD-ROM drive. All remaining bytes are sent uninterpreted to the drive unit.
The function and content of these bytes are entirely device and device driver dependent. This function is provided to allow access to device-specific features that are not addressed under any other portion of the device driver spec.

Close Tray

```
CloseTray DB      5           ; Control block code
```

This command is the logical complement to the Eject Disk command. This command will instruct drives that can do so to close the door or tray.

READ LONG
Command code = 128
ES:BX = ReadL

```
ReadL     DB      13 dup (0) ; Request header
          DB      ?          ; Addressing mode
          DD      ?          ; Transfer address
          DW      ?          ; Number of sectors to read
```

```
DD      ?       ; Starting sector number
DB      ?       ; Data read mode
DB      ?       ; Interleave size
DB      ?       ; Interleave skip factor
```

The request block is different from a normal character device READ to accommodate the larger size and different characteristics of CD-ROM devices.
The media descriptor byte, which has no meaning for character devices, is now the addressing mode field. The following values are recognized addressing modes:

0	HSG addressing mode
1	Redbook addressing mode
2-255	Reserved

The default addressing mode is the HSG addressing mode. Long (DWORD) address values are treated as logical block numbers, as defined by the High Sierra proposal. When Redbook addressing mode is on, all disk addresses are interpreted as Minute/Second/Frame addresses, according to the Philips/Sony Redbook standard. Each of these fields is 1 byte. The least significant byte of the address field contains data for frames, the second least significant byte contains data for seconds, the third least significant byte contains data for minutes, and the most significant byte of the 4-byte field is unused. These values are represented in binary rather than in BCD format. For example, if we are referencing the sector addressed by minute 36, second 24, frame 12, the hex long value for this would be 0x0024180C. This 4 byte value is recorded in the starting sector number field with the least significant byte first and most
significant byte last.

The relationship between High Sierra sectors and Redbook frames is described by the equation:

Sector = Minute * 60 * 75 + Second * 75 + Frame - 150

The byte/sector count field becomes the number of sectors to read and the starting sector number expands from one word to two, which means we can address up to 4 giga-sectors (over 8 terabytes). The DWORD ptr for requested volume ID is eliminated and MSCDEX.EXE will keep track of what volume is needed. MSCDEX.EXE handles buffering requests, but performance may be improved if the device driver reads ahead or uses a sector caching scheme, given the slow seek times of CD-ROM drives. The operating system will use the prefetch function when it can to give hints to the driver.

The data read mode field will be one of the following:

0	Cooked mode
1	Raw mode
2-255	Reserved

Cooked mode is the default mode in which the hardware typically handles the EDC/ECC and the device driver returns 2048 bytes of data per sector read. When raw mode is set, the driver will return all 2352 bytes of user data, including any EDC/ECC present independent of the actual sector mode (Mode 2 Form 1 vs. Mode 2 Form 2). User programs will have to consider this and allow enough room for buffer space when reading in raw mode as each sector returned will take up 2352 bytes of space. Drives that cannot return all 2352 bytes will return what they can and leave blank what they cannot. For example, drives that can return all 2336 bytes except the 16 byte header will leave a space in the first 16 bytes where the header would go so that the sectors align on 2352 byte boundaries. Drivers should do what they can to return as much of the user data per sector as possible.

The two interleave parameters are for drivers that support interleaved reading. If the driver does not support interleaving, these fields are both ignored. If it does, interleave size is the number of consecutive logical blocks or sectors that are stored sequentially, and the interleave skip factor is the number of consecutive logical blocks or sectors that separate portions of the interleaved file.

READ LONG PREFETCH
Command code = 130
ES:BX = ReadLPre

```
ReadLPre  DB    13 dup (0); Request header
          DB    ?               ; Addressing mode
          DD    0               ; Transfer address
          DW    ?               ; Number of sectors to read
          DD    ?       ; Starting sector number
          DB    ?       ; Read mode
          DB    ?       ; Interleave size
          DB    ?       ; Interleave skip factor
```

This function is similar in form to READ LONG, but control returns immediately to the requesting process. The device driver is not obligated to read in the requested sectors but can instead consider the request for these sectors as hints from the operating system that they are likely to be needed. It is recommended that at a minimum, the driver seek to the location provided. The attribute in the device status for prefetching is used to distinguish drivers that do more than just seek to the given location. The requests are low priority and preemptible by other requests for service. A READ LONG PREFETCH with 0 number of sectors to read should be treated as an advisory seek, and the driver can, if it is not busy, move the head to the starting sector. Since prefetching requests are advisory, there will be no functional difference between a device driver that supports prefetching from one that does not, except in terms of performance. The transfer address is not applicable for this call as the driver is not meant to transfer any data into the user address space.

SEEK
Command code = 131
ES:BX = SeekReq

```
SeekReq   DB    13 dup (0) ; Request header
          DB    ?       ; Addressing mode
          DD    0       ; Transfer address
          DW    0       ; Number of sectors to read
          DD    ?       ; Starting sector number
```

Control returns immediately to the caller without blocking and waiting for the seek to be completed. The number of sectors to be read and the transfer address are ignored. SEEK is used to relocate the head in order to begin playing audio or video tracks, or in anticipation of reading in a particular region on the disk. Further requests for disk activity will wait until the given SEEK is completed. This seek is not advisory and the head must move to the desired location. If the address is not within the range of the disc, return a "SEEK ERROR".

PLAY AUDIO
Command code = 132
ES:BX = PlayReq

```
PlayReq   DB    13 dup (0) ; Request header
          DB    ?               ; Addressing mode
          DD    ?               ; Starting sector number
          DD    ?               ; Number of sectors to read
```

This function will cause the driver to play the selected audio tracks until the requested sectors have been exhausted or until play is interrupted with an AUDIO STOP request. Control returns immediately to the caller. The busy bit in the status word will indicate if the drive is presently playing audio and when the play request is completed. The busy bit in the status word of the request header, as well as the paused bit and starting and ending locations returned by the Audio Status Info IOCTL, will be updated. If the drive does not support playing audio this request is ignored and it should return the error code for Unknown Command. If a data address is supplied, a "READ FAULT" error should be returned and the command should fail. If the sum of the starting sector and number of sectors to read exceeds the range of the disc, or there are intermediate data tracks, return the error code for PARAMETER OUT OF RANGE.

STOP AUDIO
Command code = 133
ES:BX = StopPlayReq

_____
StopPlayReq DB 13 dup (0) ; Request header _____

This function is included to pause the drive unit when it is currently in play mode, or to reset the starting and ending locations when in paused mode. If applicable, at the next stopping point it reaches the drive will discontinue playing, and process the next request. The busy bit in the status word of the request header, as well as the paused bit and starting location returned by the Audio Status Info IOCTL, will be updated. If the drive does not support playing audio, it should ignore this request and return the error code for Unknown Command.

RESUME AUDIO
Command code = 136
ES:BX = ResumeReq

_____
ResumeReq DB 13 dup (0) ; Request header _____

This function is used to resume playing audio tracks when a previous PLAY AUDIO call has been paused with a STOP AUDIO command. It will resume playing from the starting location indicated by the Audio Status Info IOCTL. It will modify the Audio Paused bit returned by the Audio Status Info IOCTL, and the busy bit in the status word of the request header. If the drive does not support playing audio, it should ignore this request and return the error code for Unknown Command.

In the following example the playing flag corresponds to the state reported by the busy bit in the status word of the request header when the drive is in audio play mode. The paused flag corresponds to the Audio Paused bit and last_startloc and last_endloc correspond to the starting and ending location reported in the Audio Status Info IOCTL.

```
// upon RESET,   NEW DISC,   or PLAY/RESUME COMPLETED the state //should be updated:
playing = FALSE;
paused = FALSE;
last_startloc = 0;
last_endloc = 0;
PLAY_AUDIO( startloc, endloc )
    {
if (playing) return error;      // Play overrides pause else if ( play( startloc, endloc ) !=
SUCCESSFUL )
return error;
            else
            {
                    playing = TRUE;
                    paused = FALSE;
                    last_startloc = startloc
```

```
                        last_endloc = endloc
                        return no error;
                }
}
STOP_AUDIO( void )
{
                if ( playing )
                {
last_startloc = present q-channel location playing = FALSE;
                        paused = TRUE;
                        stop();
                        return no error;
                }
                else
                {
                        playing = FALSE;
                        paused = FALSE;
                        last_startloc = 0;
                        last_endloc = 0;
                        return no error;
                }
}
RESUME_AUDIO()
{
                if ( paused )
                {
if ( play( last_startloc, last_endloc ) != SUCCESSFUL ) return error;
                        else
                        {
                                playing = TRUE;
                                paused = FALSE;
                        return no error;
                }
        }
        else
                return error;
}
WRITE LONG
Command code = 134
ES:BX = WriteL
_________________
WriteL      DB      (dup 13 0) ; Request header
            DB      ?       ; Addressing mode
            DD      ?       ; Transfer address
            DW      ?       ; Number of sectors to write
            DD      ?       ; Starting sector number
            DB      ?       ; Write mode
            DB      ?       ; Interleave size
            DB      ?       ; Interleave skip factor
_________________
```

The device will copy the data at the transfer address to the CD RAM device at the sector indicated. The media must be writable for this function to work. Data is written sector by sector, depending on the current write mode and the interleave parameters. The following values are recognized as valid write modes:

0 Mode 0

1 Mode 1
2 Mode 2 Form 1
3 Mode 2 Form 2
4-255 Reserved

Writing in Mode 1 is the default and must be supported. If the device driver supports the other modes, then they can be used. If Mode 0 is used, the transfer address is ignored and all sectors are written with zeroes. If the current write mode is Mode 1 or Mode 2 Form 1, each sector will consist of 2048 bytes of data located sequentially at the transfer address. If the write mode is Mode 2 Form 2, the device driver will expect 2336 bytes of data per sector at the transfer address.

WRITE LONG VERIFY
Command code = 135
ES:BX = WriteLV

```
_________________
WriteLV     DB      (dup 13 0) ; Request header
            DB      ?          ; Addressing mode
            DD      ?          ; Transfer address
            DW      ?          ; Number of sectors to write
            DD      ?          ; Starting sector number
            DB      ?          ; Write mode
            DB      ?          ; Interleave size
            DB      ?          ; Interleave skip factor
```

_____
This function is identical to WRITE LONG, with the addition that the device driver is responsible for verifying the data written to the device.

INPUT FLUSH
Command code = 7
ES:BX = FlushI

```
_________________
FlushI          DB    13 dup (0)       ; Request header _________________
```

Requests that the device driver free all input buffers and clear any pending requests.
OUTPUT FLUSH
Command code = 11
ES:BX = FlushO

```
_________________
FlushO          DB    (dup 13 0)       ; Request header _________________
```

Requests that the device driver write all unwritten buffers to the disk.

DEVICE OPEN
DEVICE CLOSE
Command code = 13,14
ES:BX = DevOpen, DevClose

```
_________________
DevOpen    DB    13 dup (0)       ; Request header _________________
```

Used by the device driver to monitor how many different callers are currently using the CD-ROM device driver. All new device drivers should support these calls even if nothing is done with the information.

End.

Example CD-ROM

The purpose of the example CD-ROM device driver is to demonstrate an implementation of a MSCDEX CD-ROM device driver conforming to the specification in the document "Microsoft MS-DOS CD-ROM Extensions Hardware-Dependent Device Driver Specification". Using this example as a starting point, together with an understanding of the device driver specification, greatly simplifies the task of creating a MSCDEX CD-ROM device driver.

Installation

This example driver is for use on an AT style bus with most of Hitachi bus drives and controllers (with the exception of their SCSI devices). The example driver is not currently shipped with the Hitachi drives. The device driver is installed in the same way as any other device, with an entry in CONFIG.SYS. The syntax of this entry is:

DEVICE=<*filename*> /D:<*device_names*> /N:<*drives*> /P:<*port address*> C:<*minutes*>
where the arguments are:

<*filename*>	The MS-DOS path to, and name of, the device driver file.
/D:<*device_names*>	The character device names that will be used by MSCDEX.EXE to find and communicate with the device driver. Although more than one device name on the command line provides names for multiple device headers to control more than one physical device, this example supports multiple devices as sub-units of a single device header (the recommended approach). Choose unique device names that you do not expect to use as file names. MS-DOS's file open code looks at the device list before opening files, so a device named 'CDROM' would prevent opening a file or directory named 'CDROM'.
/N:<*drives*>	The number of sub-units (physical drives) attached to the interface card (defaults to one sub-unit). This example supports a maximum of 8 drives, however when more than 4 drives are controlled by one device driver, you need to use an interface card and drives that support 8-drive-daisy-chaining (for example, CDR-IFI8-A or CDR-3600)
/P:<*port address*>	The port address to be used to communicate with the card (defaults to 300H).
/C:<*minutes*>	The number of minutes of driver inactivity before the power save feature (which will increase the life span of the drive unit) should turn off the drive motor. This is a value from 0 to 14H, where 0 disables the spin down feature.

Building the Driver

The source files and the makefile you need to build the driver are in the HITACHIA directory on the SUP: disk. The tools (MASM 5.1, etc.) are in BIN directory of the TOOLS: disk. The files needed to build the driver include:

MAKEFILE	The makefile to build the driver.
MSCDEX.ASM	This file is device independent and can be used by a device driver with no changes. It contains all the entry points for the commands that a MSCDEX CD-ROM device driver is expected to handle. Some commands, and certain device independent data, are handled without

	actually calling the device dependent code in CD.ASM.
CD.ASM	This file contains device dependent code for all the entry points defined in MSCDEX.ASM. These routines start with the letters "cdrom_", and have explicitly defined entry and exit conditions. A device driver should use this file as a starting point by replacing the Hitachi implementation with the necessary device dependent code to perform the equivalent functions, and carefully adhering to the indicated entry and exit conditions.
MSCDEX.INC	This is an include file for MSCDEX.ASM and CD.ASM with useful definitions for command structures, error codes, and the status word.
MACROS.MAC	This is an include file for MSCDEX.ASM and CD.ASM with useful macro definitions to make the code easier to read.
CD.INC	This is an include file for CD.ASM defining names for the low-level driver commands provided in CDREAD.ASM.
CDREAD.ASM	This file contains the function CDREAD() which has about 26 sub-functions to perform various low level driver commands.
CDREAD.AT	This is an include file for CDREAD.ASM defining various port addresses, control codes, and other defines.
CDREAD.DOC	This file documents the entry and exit conditions for the sub-functions provided by CDREAD().
ECC.ASM	This file contains the routine to do the last level of error correction in the event the Cross Interleave Reed-Solomon Code implemented in the drive is not able to correct a read error. For performance considerations it is recommended that this level of error correction also be done on the drive.
TRACER.ASM	This file is linked if the DEBUG label is defined in the MAKEFILE. It defines WriteAux which directs messages to the serial port.
CMACROS.INC	This is an include file used by TRACER.ASM, and defines various constructs to improve readability and ease writing of assembly code.

End.

Installing MSCDEX.EXE and Associated CD-ROM Device Drivers Without Using the SETUP program

Copying the MSCDEX Files

If you have a floppy-disk based system, create a bootable MS-DOS floppy disk for the MSCDEX files. You can do this by connecting to drive A, inserting your normal bootable floppy (the disk you normally use to start your computer) into drive A and a blank unformatted disk in drive B, and running diskcopy.

Next, copy the contents of the MSCDEX directory onto your bootable floppy. Change the directory on the CDROM to the MSCDEX directory and leave the bootable floppy that was just created in drive A. Type the following to copy the files:

COPY *.* A:

If there is not enough space to copy the files, you will have to make room on your disk by deleting unnecessary files and repeating the copy procedure.

If you have a hard-disk based system (usually drive C:), change directory to the MSCDEX directory and type:

COPY *.* C:

Once this is finished, you need to modify the CONFIG.SYS and AUTOEXEC.BAT files to complete the installation of MSCDEX.

Changing the CONFIG.SYS File

After copying the system files, you need to edit the CONFIG.SYS file used to boot the computer. You will need to update the file to set the last drive designator, and to add the command line to load the CD-ROM device driver. (If the CONFIG.SYS file does not exist, you will have to create it.)

Scan the contents of CONFIG.SYS to see if it contains a LASTDRIVE command line that specifies the last drive designator. You will have to add or modify this line to read:

LASTDRIVE=Z

This lets MS-DOS know that it will have to allocate enough drive records to account for the additional CD-ROM drive letters. It's a good idea to have this line as the first entry in the CONFIG.SYS file. The last drive only needs to be the largest drive letter that will be used on the system. For example, a PC with 2 floppy drives (drives A and B), a hard disk (drive C), and two CD-ROM drives (drives D and E) would only need LASTDRIVE=E. A computer that is on a network though will typically allocate more drive records than are used so that new network drives can be added after booting.

Next, add entries for each device driver that is going to be installed. The entry will have the following format:

DEVICE=<*driver.sys*> /D:<*device_name*> <*driver specific switches*>

Replace the *driver.sys* parameter with the full pathname of the device driver being installed. This will include the drive letter of the bootable drive, the directory path of the device driver, and the filename of the device driver. For example, if the device driver is located in the root directory on a hard disk system that boots off drive C:, a sample entry might read:

DEVICE=C:\DRIVER.SYS /D:MSCD001

For the Hitachi driver, this should read:

DEVICE=C:\HITACHI.SYS /N:1 /D:MSCD001

The /D:*device_name* parameter is the name MSCDEX will use to find the device driver. After the device driver is installed, the system will use this name to identify the device driver. Every installed device driver must have a unique name. In addition, because of the way MS-DOS handles opening files, if a file has the same name as a device driver, then a file open call with that name will open a handle to the device driver rather than to the file. For this reason, you will want to chose a *device_name* that will not likely be used as a file name. We recommend using names of "MSCDXXX" where XXX is three digits. This will also let the setup program locate CD-ROM device driver entries in the CONFIG.SYS file at a later date.

The remaining fields for *driver specific switches* are device driver dependent. The applicable switches are described in the documentation accompanying individual device drivers.

Changing the AUTOEXEC.BAT File

You need to edit the AUTOEXEC.BAT file to include an entry to invoke MSCDEX each time the system is booted. If your computer is on a network, the network software must be installed before MSCDEX. Otherwise the network will not install, since MSCDEX will not allow the network to be installed after it.

If your AUTOEXEC.BAT file starts up a shell program such as DOSSHELL or Windows, or runs another BAT file, make sure the line that starts MSCDEX is ahead of the line that starts your shell or BAT file. Otherwise MSCDEX will not have a chance to start the CD-ROM drives before your shell or BAT file begins.

A sample entry in AUTOEXEC.BAT to install MSCDEX would be:

C:\MSCDEX.EXE /D:MSCD000 /D:MSCD001 /M:20 /V /E

Include the drive letter of the drive containing MSCDEX and its full pathname for MSCDEX. If the previous example, MSCDEX will be in the root directory. Each device driver will have a *device_name* listed on the command line to MSCDEX following the **/D:** switch. MSCDEX uses this parameter to locate each device driver. Names used must match those used for the **/D:***device_name* parameters for each device entry in CONFIG.SYS.

The **/M:<value>** switch determines how many sector buffers MSCDEX allocates when it installs itself. The larger this value is, the more sector cache entries are available and the less MSCDEX will have to read directly from the CD-ROM drive. It is especially important that there be enough entries to cache the path table, and the more entries available for directory sectors, the less MSCDEX will have to reread the directory files. Typically, each drive should have at a minimum about 4-5 buffers per drive but the larger this value is, the better the performance will be.

Both the CD-ROM device driver and MSCDEX require small amounts of additional memory to run

which should not cause any problems. But if you find that some applications no longer run because of insufficient memory after you have installed MSCDEX, you might review your CONFIG.SYS and AUTOEXEC.BAT files to locate other entries that use memory that you might not require and can remove. For example, you might be able to free memory by eliminating entries that start the GRAPHICS or FASTOPEN programs.

There is an additional switch **/E** which tells MSCDEX to use expanded memory if it is available. The verbose switch **/V** asks MSCDEX to print additional information about memory usage during initialization.

For software that requires that the CD-ROM drive be identified by particular drive letter, you can use the **/L:**<*drive letter*> switch. The SETUP program does not set this switch so you will have to edit your AUTOEXEC.BAT file if this switch is needed. For example, the following assigns CD-ROM drives starting at drive letter L:

MSCDEX /D:MSCD001 /L:L

There are also two other switches: **/K** and **/S.** The **/K** switch causes MSCDEX to look for a Supplementary Volume Descriptor that identifies a shift-JIS Kanji volume for Japanese. The **/S** switch tells MSCDEX to patch MS-DOS to allow sharing of CD-ROM drives on MS-NET based servers.

End.

Kanji Support in CD-ROM Extensions

The Kanji support in MSCDEX presently recognizes High Sierra CD-ROM discs with a coded character set that has bit 0 set to 1 in the volume flags. The set bit specifies that at least one escape sequence is not registered according to ISO 2375, and has an escape sequence of three bytes in the coded character set for descriptor identifiers field of "$+:". This indicates that the character set is a private multi-byte G3 coded character set and identifies the disc as having shift-Kanji.

Scanning for the Supplementary Volume Descriptor

The command line switch **/K** makes MSCDEX scan for the SVD (Supplementary Volume Descriptor) instead of the PVD (Primary Volume Descriptor). When you specify this switch, MSCDEX uses the shift-Kanji SVD if it is present, otherwise it uses the PVD. All discs are required by ISO-9660 to have a PVD even if there is an SVD.
In addition, the accompanying program SVD can change the default preference each CD-ROM drive has for scanning for a SVD or PVD. The syntax is:

SVD [<drive letter>: <std | svd>]

Running SVD with no arguments will report the current settings. Including a drive letter and either STD or SVD will change the preference for that drive from one to the other. Alternately, you can use INT 2fh function 150Eh. For more information on this function, see the Function Requests Specification section.

End.

MSCDEX Command Services.

There is a need for access to features from the MSCDEX redirector that transend DOS capabilities. This proposal documents a means the application can use to talk directly to MSCDEX to request information or set parameters that only MSCDEX can provide. This document outlines the services MSCDEX provides at present. Comments and suggestions are welcome.

Access to these functions is provided through an INT 2Fh interface. AH contains 15h which is what MSCDEX will use to tell its requests from those of other INT 2Fh handlers. AL will contain the code of the function to be performed. When writing applications for Windows, all pointers must be allocated in lower memory and properly translated to real mode addresses through the DOS Protected Mode Interface (DPMI). Information on DPMI is available from the Intel Corporation.

Function Request Command Codes:

Contents of AL	Function
00h	Get Number of CD-ROM drive letters
01h	Get CD-ROM drive device list
02h	Get copyright file name
03h	Get abstract file name
04h	Get bibliographic doc file name
05h	Read VTOC
06h	Reserved
07h	Reserved
08h	Absolute Disk Read
09h	Absolute Disk Write
0Ah	Reserved
0Bh*	CD-ROM Drive Check
0Ch*	MSCDEX Version
0Dh*	Get CD-ROM drive letters
0Eh*	Get/Set Volume Descriptor Preference
0Fh*	Get Directory Entry
10h**	Send Device Request
11h-0FFh	Reserved

* - New functions added in version 2.00
** - New functions added in version 2.10

- Get Number of CD-ROM drive letters

AX	1500h
BX	Number of CD-ROM drive letters used
CX	Starting drive letter of CD-ROM drive letters (A=0, B=1, ... Z=25)

MSCDEX will return the number of CD-ROM drive letters in BX and the starting drive letter in CX. The first CD-ROM device will be installed at the starting drive letter and subsequent drives will be assigned the next greater drive letter. A single device driver may be assigned to more than one drive letter, such as the case of a device driver that supports multiple units. MSCDEX keeps track

of which sub-unit a particular drive letter is assigned to.

NOTE: This function can be used to determine if MSCDEX is installed by setting BX to zero before executing INT 2Fh. MSCDEX is not installed if BX is still zero on return.

Also, in a networking environment, one cannot assume that drive letters will always be assigned contiguously beginning with the starting drive letter. Use function Get CD-ROM drive letters instead.

- Get CD-ROM drive device list

_____

AX 1501h
ES:BX Transfer address; pointer to buffer to copy drive letter device list

_____

The buffer must be large enough to hold the device list. By calling function *Get Number of CD-ROM Drive Letters*, one can find out the number of CD-ROM drive letters and the buffer size will be a multiple of that. This will be an absolute maximum of 26. Each drive letter device entry will consist of one byte for the sub-unit followed by 4 bytes for the address of the device header assigned to that drive letter. This byte for the sub-unit takes care of the problem of distinguishing which unit is assigned to which drive letter for device drivers that handle sub-units.

For example: Suppose there are two installed CD-ROM device drivers, CD_ONE, which supports 1 sub-unit, and CD_TWO, which supports two sub-units, on a system with 2 floppy drives (A=0 and B=1) and a hard disk (C=2). Then asking for the number of CD-ROM drive letters will report that there are 3 drive letters used starting at drive letter D=3. ES:BX must point to a buffer that is at least 3*5 = 15 bytes long. The buffer will be filled as follows:

```
ES:BX  = Buffer
Buffer DB     0                              ; sub-unit of CD_ONE on drive letter D:
DD     <far addr of CD_ONE device header>
DB     0                                     ; sub-unit of CD_TWO
                                             ;on drive letter E:
DD     <far addr of CD_TWO device header>
DB     1                                     ; sub-unit of CD_TWO
                                             ; on drive letter F:
DD     <far addr of CD_TWO device header>
```

- Get copyright file name

_____

AX 1502h
ES:BX Transfer address; pointer to a 38 byte buffer
CX CD-ROM drive letter (A=0, B=1, ... Z=25)

_____

MSCDEX will copy the name of the copyright file in the VTOC for that drive letter into the buffer space provided. The copyright filename is presently restricted in the High Sierra proposal to 8.3 but we require 38 bytes here for the possibility at a later date of handling 31 character file names plus 6 bytes for a ';' and 5 digit version number and 1 byte for a NULL at the end. Carry will be set if the drive letter is not a CD-ROM drive and error_invalid_drive (15)

will be returned in AX.

- Get abstract file name

_____

AX 1503h
ES:BX Transfer address; pointer to a 38 byte buffer
CX CD-ROM drive letter (A=0, B=1, ... Z=25)

_____

MSCDEX will copy the name of the abstract file in the VTOC for that drive letter into the buffer space provided. The abstract filename is presently restricted in the High Sierra proposal to 8.3 but we require 38 bytes here for the possibility at a later date of handling 31 character file names plus 6 bytes for a ';' and 5 digit version number and 1 byte for a NULL at the end. Carry will be set if the drive letter is not a CD-ROM drive and error_invalid_drive (15) will be returned in AX.

- Get bibliographic documentation file name

_____

AX 1504h
ES:BX Transfer address; pointer to a 38 byte buffer
CX CD-ROM drive letter (A=0, B=1, ... Z=25)

_____

NOTE: This function is provided in advance of the ISO standard. For discs complying with the May 28th draft from the High Sierra Group, this function will return a null string as though the field is blank on the disc.

MSCDEX will copy the name of the bibliographic documentation file in the VTOC for that drive letter into the buffer space provided. The bibliographic documentation filename is presently restricted in the High Sierra proposal to 8.3 but we require 38 bytes here for the possibility at a later date of handling 31 character file names plus 6 bytes for a ';' and 5 digit version number and 1 byte for a NULL at the end. Carry will be set if the drive letter is not a CD-ROM drive and error_invalid_drive (15) will be returned in AX.

- Read VTOC

_____

AX 1505h
ES:BX Transfer address; pointer to a 2048 byte buffer
CX CD-ROM Drive letter
DX Sector index

_____

This function is provided to scan the Volume Descriptors on a disc. A sector index of 0 will read the first volume descriptor, 1 reads the second, etc. If there is no error, then AX will return 1 if the volume descriptor read was the standard volume descriptor, 0FFh if it was the volume descriptor terminator and there are no more volume descriptors to be read, and 0 for all other types.

If there is an error in processing the request, the Carry Flag will be set and AL will contain the MS-DOS error code. These will be either error_invalid_drive (15) or error_not_ready (21).

- Absolute Disk Read

_____

AX 1508h
ES:BX Disk Transfer Address; pointer to a buffer to copy data to
CX CD-ROM Drive letter (A=0, B=1, ... Z=25)
DX Number of sectors to read
SI:DI Starting sector

_____

This function corresponds to INT 25h. It will be converted directly into a READ_LONG device driver request and sent to the correct device driver. There are no requirements for this call to pop flags as there are with INT 25h. SI holds the high word and DI the low word for the starting sector to begin reading from.

If there is an error in processing the request, the Carry Flag will be set and AL will contain the MS-DOS error code. These will be either error_invalid_drive (15) or error_not_ready (21).

- Absolute Disk Write

_____

AX 1509h
ES:BX Disk Transfer Address; pointer to buffer to copy data from
CX CD-ROM Drive letter
DX Number of sectors to write
SI:DI Starting sector

_____

This function corresponds to INT 26h. It is not supported at this time and is reserved. It is intended to be used by authoring systems.

- CD-ROM Drive Check

_____

AX 150Bh
BX Signature word
CX CD-ROM Drive letter (A=0, B=1,...Z=25)

_____

This function returns whether or not a drive letter is a CD-ROM drive supported by MSCDEX. If the extensions are installed, BX will be set to ADADh. If the drive letter is supported by MSCDEX, then AX is set to a non-zero value. AX is set to zero if the drive is not supported. One must be sure to check the signature word to know that MSCDEX is installed and that AX has not been modified by another INT 2Fh handler.

- MSCDEX Version

AX 150Ch
BX MSCDEX Version

This function returns the version number of the CD-ROM Extensions installed on the system. BH contains the major version number and BL contains the minor version. Values returned are binary. For example, BX would contain 0x020a for version 2.10. This function does not work on versions earlier than 2.00 so if BX is zero before and after this function is called, an earlier version of MSCDEX is installed.

- Get CD-ROM Drive Letters

AX 150Dh
ES:BX Transfer address; pointer to buffer to copy drive letter device list

---

The buffer must be large enough to hold a list of drive letters. The buffer size will be a multiple of the number of drives returned by the *Get Number of CD-ROM drive letters* function. There are a maximum of 26 drive letters. Each drive letter entry is a single byte (0=A:, 1=B: .. 25=Z:) that exactly corresponds each respective entry returned by the command *Get CD-ROM drive device list.* This command is included to allow applications to locate CD-ROM drives supported by MSCDEX. CD-ROM drive letters may sometimes be noncontiguous so this command is necessary.

For example: Suppose there is an installed CD-ROM device driver FOO supporting 3 sub-units on a system with 2 floppy drives (A=0 and B=1), a hard disk (C=2) and a network drive (E=4). Note the network drive occupies one of the drive letters normally taken by a CD-ROM drive. MSCDEX assigns that CD-ROM drive to the next available drive letter. Asking for the number of CD-ROM drive letters reports there are 3 drive letters used starting at drive letter D=3. ES:BX must point to a buffer that is at least 3 bytes long and will be filled as follows:

```
ES:BX  = Buffer
Buffer DB     3                          ; drive letter for CD-ROM (D=3)
DB     5                          ; drive letter for CD-ROM (F=5)
DB     6                          ; drive letter for CD-ROM (G=6)
```

- Get/Set Volume Descriptor Preference

```
AX     150Eh
BX     0 - Get Preference. 1 - Set Preference
CX     CD-ROM Drive letter (A=0, B=1,...Z=25)
DX     if BX = Get Preference
              DX = 0
              MSCDEX will return preference settings in DX
       if BX = Set Preference
              DH = volume descriptor preference
                     1 - PVD - Primary Volume   Descriptor
                     2 - SVD - Supplementary Volume Descriptor
```

DL = Supplementary Volume Descriptor Preference
if DH = PVD
DL = 0
if DH = SVD
1 - shift-Kanji (an unregistered ISO coded character set)

Normally, MSCDEX will scan for the PVD (Primary Volume Descriptor) when initializing a CD-ROM. This behavior can be altered for each individual drive to scan for a SVD (Supplementary Volume Descriptor) instead. A CD-ROM drive set to scan for an SVD will use the PVD if there is no SVD present. There can be more than one SVD on a CD-ROM but at present, MSCDEX will only recognize SVDs for shift-Kanji CD-ROMs. Carry will be set, AX will be set to error_invalid_function (1) and DX will be set to 0 if the coded character set is not recognized.

If BX contains Get_Preference, MSCDEX will report the present setting for that drive. If DX is still zero on return, that version of MSCDEX does not support this function or reading SVDs. Otherwise DX will contain the setting.

If the drive letter is not a CD-ROM drive, carry will be set and error_invalid_drive (15) will be returned in AX. If BX is anything other than Get/Set_Preference, AX will be set to error_invalid_function (1) and carry will be set.

- Get Directory Entry

AX	150Fh
CL	CD-ROM Drive letter (A=0, B=1,...Z=25)
CH	Copy flags (bit 0: 0 - direct copy, 1 - copied to structure)
ES:BX	Pointer to buffer with null-terminated path name
SI:DI	Pointer to buffer to copy directory record information
AX	0 is returned if the disc is High Sierra, 1 is returned if the disc is ISO-9660

The pathname expected is a null-terminated string e.g. char far *path = "\\a\\b\\c.txt"; (note: the "\\" characters map to a single '\' character in C so this would be '\a\b\c.txt' if printed). The path must consist only of valid High Sierra or ISO-9660 filename characters and must not contain any wildcards nor may it include entries for '.' or '..'.

The copy flags indicate how the data should be copied. If bit 0 (0x01) is 0, then the directory entry is copied as it appears on the disc. The directory record is a direct copy from the directory file and it is up to the application to choose what fields to use. If bit 0 is 1, then the directory entry is copied into a structure that removes any differences between High Sierra and ISO-9660 directory entries and makes some fields more accessible or easily interpreted.

If copying the directory entry as it appears on the disc, the buffer to copy the directory record should be at least 255 bytes long to include all system use information and if copying into the common structure at least 280 bytes long otherwise MSCDEX may overrun the end of the buffer.

Carry will be set and an error code returned if there were problems with the request. The error codes will be error_invalid_drive (15) if the drive letter is incorrect, error_not_ready (21) if the

disc didn't initialize correctly, error_file_not_found (2) if the file was not found and error_no_more_files (18) if the pattern fails to find a match or if MSCDEX failed to allocate buffers.

The format of the directory record for High Sierra discs is:

```
/* High Sierra directory entry structure */
typedef struct hsg_dir_entry
{
        uchar   len_dr;             /* length of this directory entry                */
        uchar   XAR_len;            /* XAR length in LBN's (logical                  */
                                                                                      /* blocks
numbers)        */
        ulong   loc_extentI;        /* LBN of data Intel format                      */
        ulong   loc_extentM;        /* LBN of data Molorola format                   */
        ulong   data_lenI;          /* length of file Intel format                   */
        ulong   data_lenM;          /* length of file Motorola format                */
        uchar   record_time[6];     /* date and time                                 */
        uchar   file_flags_hsg;     /* 8 flags                                       */
        uchar   reserved;           /* reserved field                                */
        uchar   il_size;            /* interleave size                               */
        uchar   il_skip;            /* interleave skip factor                        */
        ushort  VSSNI;              /* volume set sequence number Intel              */
        ushort  VSSNM;              /* volume set sequence number                    */
                                                                                      /*
Motorola        */
        uchar   len_fi;             /* length of name                                */
        uchar   file_id[...];       /* variable length name upto 32                  */
                                                                                      /* chars
        */
        uchar   padding;            /* optional padding if file_id is                */
                                                                                      /* odd
length  */
        uchar   sys_data[...]       /* variable length system data                   */
        } hsg_dir_entry;
```

The format of the directory record for ISO-9660 discs is:

```
/* ISO-9660 directory entry structure */
typedef struct iso_dir_entry
{
        uchar   len_dr;             /* length of this directory entry                */
        uchar   XAR_len;            /* length of XAR in LBN's                        */
        ulong   loc_extentI;        /* LBN of data Intel format                      */
        ulong   loc_extentM;        /* LBN of data Molorola format                   */
        ulong   data_lenI;          /* length of file Intel format                   */
        ulong   data_lenM;          /* length of file Motorola format                */
        uchar   record_time[7];     /* date and time                                 */
        uchar   file_flags_iso;     /* 8 flags                                       */
        uchar   il_size;            /* interleave size                               */
        uchar   il_skip;            /* interleave skip factor                        */
        ushort  VSSNI;              /* volume set sequence num Intel                 */
        ushort  VSSNM;              /* volume set sequence num Motorola              */
        uchar   len_fi;             /* length of name                                */
        uchar   file_id[...];       /* variable length name upto 32                  */
                                                                                      /*chars
```

```
        */
        uchar     padding;          /* optional padding if file_id is          */
                                                                    /*odd
length  */
        uchar     sys_data[...]     /* variable length system data             */
                                                                    }
iso_dir_entry;
```

Note that the difference between the two forms is the file flag byte moved to account for an additional byte of date and time used for a Greenwich mean time offset. Also, the C structs above are not syntactically correct as C does not allow variable length arrays as struct elements. They are meant to illustrate the differences between the directory entries. See the May 28th draft of the High Sierra proposal or ISO-9660 for a more complete explanation of the fields.

If bit 0 is set to one in the Copy Flags, then the format of the directory entry structure that is returned is the following:

```
typedef struct dir_entry
 {
        uchar     XAR_len;          /* length of XAR in LBN's                  */
        ulong     loc_extent;       /* logical block number of file start      */
        ushort    lb_size;          /* logical block size of disc              */
        ulong     data_len;         /* length of file                          */
        uchar     record_time[7];   /* date and time                           */
        uchar     file_flags;       /* 8 flags                                 */
        uchar     il_size;          /* interleave size                         */
        uchar     il_skip;          /* interleave skip factor                  */
        ushort    VSSN;                                                       /*
volume set sequence number          */
        uchar     len_fi;           /* length of the filename                  */
        uchar     file_id[38];      /* filename, null terminated               */
        ushort    file_version;     /* version number of file                  */
        uchar     len_su;           /* length of valid system use bytes        */
        uchar     su_data[220]      /* up to 220 bytes of system use data      */
        } dir_entry;
```

The location of the extent is reported in logical block numbers and the caller can use the logical block size on the disc (logical block sizes of 512, 1024, and 2048 bytes per sector are valid for CD-ROM) to determine the exact sector and offset in that sector that the file extent begins in. If the logical block size is 2048 then logical block numbers are equivilent to logical sector numbers.

The Greenwich mean time offset byte in this structure for May 28 High Sierra discs is always 0. The file_id field contains the file identifier less the semicolon and version number if present and is null terminated. The version number is already parsed and is present in binary form in file_version. All bytes beyond what are indicated by len_fi and len_su in the file_id and su_data fields are undefined except the null byte immediately following the file identifier which is not counted as part of the filename length.

- Send Device Driver Request

_____

AX 1510h
CX CD-ROM drive letter (A=0, B=1, ... Z=25)

ES:BX Address of CD-ROM device driver request header

This function has been added to simplify communication with CD-ROM drivers and help prevent contention between applications that wish to communicate with the device driver. It is highly recommended that all applications communicate with device drivers through this function request. Applications using this function will not have to locate the device driver. The format of the request header is specified by the Microsoft MS-DOS CD-ROM Extensions Hardware-dependent Device Driver Specification. MSCDEX will supply the sub-unit field.

End.

MS-Dosifying Your CD-ROM

Most of the following guidelines apply to the present version of the Microsoft CD-ROM Extensions. Future versions of the Extensions are expected to support many of the items listed below that are presently best avoided. The behavior of the Extensions with fields and records that are currently ignored might change at any time.

Correctness

Make sure that your disc is in valid High Sierra format. *Nothing* is guaranteed if your disc does not have a valid format. Surprisingly enough, we have received several discs that have one or more invalid formatted data areas. Some of the formatting errors included directories being sorted incorrectly, incorrect path table sizes, incorrect directory file sizes, directories missing from the path table, and invalid directory names. In almost every case, the Extensions will behave incorrectly, and at worst, the system might crash.

In addition to running validation software to verify the High Sierra image, you should also verify that the Extensions work with your CD-ROM disc and application software before distributing it. Unfortunately, it may not matter if your disc is correct and the Extensions are wrong if they don't work together. Please report any and all problems you think are in the Extensions to Microsoft so that they can be fixed.

Path Table and Directory Sizes

Using invalid sizes for the path table and directory are a common error when creating CD-ROMS for MS-DOS. The following guidelines apply to path tables and directories:

- MS-DOS directory file sizes are always a multiple of the logical sector size2 kilobytes.
- Path table sizes are always the exact number of bytes contained in the path table (which is typically not a multiple of 2 kilobytes).
- You must not have any blank directory sectors.
- The directory length must reflect the correct length of the directory file.
- Directory sectors always begin on a logical sector boundary.

8.3 Filenames

MS-DOS cannot handle longer than 8.3 filenames (8 characters for the filename and 3 characters for the filename extension). If the CD-ROM filename is longer than 8.3, then the filename will be truncated. If this happens, two files that are not unique within 8.3 characters map to the same filename. For example, both FILENAME1.TXT and FILENAME2.TXT will appear as FILENAME.TXT.

Kanji filenames are also limited to 8.3 or 4.1 Kanji characters. Only shift-Kanji filenames are recognized at present. To get Kanji, you must specify a supplementary volume descriptor indicating you have Kanji filenames. Contact Microsoft to find out how this is done.

Record Formats

The Extensions do not support any record formats so the Extensions will ignore the RECORD bit in the file flags byte in the directory entry.

Interleaving

In the present version, the Extensions do not support interleaving so if the interleave size and interleave factor are non-zero, the file will ignore these fields and return erroneous data.

Multi-Extent Files

Multi-extent files are not supported in the present version. Each extent of a multi-extent file will appear as a separate file with the same name.

Multi-volume Disc Sets

Multi-volume disc sets are not supported in the present version. Directories that are located on another volume could potentially cause the Extensions to crash if searched and erroneous data will be returned for files that are located on another volume.

Coded Character Sets

Only one coded character set or supplementary volume descriptor is recognized in the latest version. This is for shift-Kanji.

Version Numbers and Associated Files

Version numbers and associated files are not supported by the Extensions. The Extensions will strip the version string off the end of the filename so that two identical filenames with different versions will appear to have the same name. There is no way to specifically ask for any but the first instance of that filename. Two files with the same name and different version numbers have the same accessing problem as two files with longer than 8.3 filenames that have been truncated to the same filename. In the case of associated files, they are not accessible by any MS-DOS function.

Protection

Protection bits are not used on MS-DOS. If the protection bit is set in the file flags byte in the directory entry for a file, it is ignored and normal access is allowed.

No XAR Support

At present, the Extensions ignore the contents of any XAR record.

Motorola Format Tables

The additional copies of the path table and any values in Motorola format (most significant bytes using the lowest address values) are ignored at present. MSCDEX only pays attention to Intel formatted values. They should be included though for portability sake.

Multiple Copies of the Path Table

The Extensions presently only read and use the first copy of the path table. Later versions may check to see that copies of the path table agree.

Additional Volume Descriptor Records

Boot records and unspecified volume descriptors are ignored. The first standard volume descriptor found is the one that is used. Additional copies are ignored at present.

File Flags

The existence bit is treated the same as the hidden bit on MS-DOS. Some other operating systems may not handle the existence bit so you might not want to use it if you are also developing for these systems.
The directory bit for High Sierra is treated the same as the directory bit in MS-DOS.
Files with the protection bit set are not found when searched for or opened.
None of the remaining bits (Associated/Record/Multi-extent/Reserved) are handled at present. Using files with these bits set will have undefined behavior.

Unique Volume Identifiers

It is highly recommended that the volume identifier be unique. The Extensions use the volume identifier to do volume tracking and to double-check if the disc has changed. Using unique volume identifiers for each CD-ROM product and version minimizes the chances of a user having two discs with the same volume identifier. This will increase the effectiveness of the checks the Extensions perform to determine if the user has changed the CD-ROM.

Your application should also use the volume label to tell if the CD-ROM disc has changed. Your application can obtain the volume label for a CD-ROM on MS-DOS from the volume identifier field

in the primary volume descriptor. The call to get the volume label is very inexpensive to make once the CD-ROM has been initialized and will cause no disc I/O to be done unless the media has changed. This is the best way for your application to tell if the disc it wants is in the drive. Your application should not communicate with the driver or drive to determine if the media has changed. Doing this can prevent the Extensions from learning that the disc has changed and from performing the initialization it needs for a new disc.

Many Small Directories or A Few Large Directories

As a rule, it is better to have many small directories that contain fewer files than one very large directory. However, the exact answer depends on your application's behavior because if you try very hard, you can thrash almost as badly with many small directories as you can with one large directory.

What makes the difference? For example, suppose you have 1000 directories with 40 files. On average, you'll read about one sector per file open and scan one-half of it. On the other hand, you could have 1 directory with 4000 files. On average, each file open in this large directory (about 100 sectors) will involve scanning about 50 sectors to open that one file. As long as it is very inexpensive to get to each directory through the path table, it is much better to have many small directories.

You can get further improvements by grouping files that are related and are opened together in each of these subdirectories. As you open each successive file, the directory sector is very likely in the disc cache. This helps minimize the reads from the CD-ROM disc. Putting each file in a separate subdirectory is extreme and might increase the access time. With this strategy you never gain the benefits of locating the next file in a directory sector that has already been cached and you needlessly enlarge the path table.

The size of the path table also limits how many subdirectories you might want. If there are too many you might thrash reading the path table sectors. Each path table sector holds pointers to 100 to 200 directory files. (The number of pointers depends on the length of the directory names.) If you have a path table that is 10 sectors long, you will want at least 10 sectors of memory buffers to hold the path table. If you use less than 10 sectors you risk re-reading sections of the path table on every file open which is very costly.

The most important point from this discussion is that you can vastly improve your file open speed by making sure you have enough memory buffers. If you are repeatedly trying to scan a 10 sector directory file (approximately 400 entries) and you only have 4 sectors in the sector cache, the cache is going to work against you because you will end up churning it excessively. If you allocate 14 sectors for this example (/M:14), then the whole directory file will find its way into the cache and you will stop hitting the disc.

The difference in speed may be several orders of magnitude. A safe recommendation is to reserve as many sectors as there are in the path table plus the number of sectors for the largest directory plus two. The last two sectors are reserved for data sectors and internal dynamic data. Multiple drives complicate this formula because the buffers are not tied to specific drives and are shared, and because not all drives are active at the same time.

Also, do not rely on the file system to do your searching for you. If you are performance conscious, finding a chunk of data by looking for it with a file name through the file system is expensive. Most of the time, locating data through the file system is fine because the cost is a single one-time operation. If this is repeated often enough, it may pay to do some of the work yourself. A better method is to place everything into one big file and cache your own hierarchy, indexing, binary trees, or whatever searching scheme you choose rather than asking for the file system to locate it.

MSCDEX Extended Errors

MSCDEX issues the following extended errors that your application can trap if it chains into the INT 24h vector:

Value	Message
100	CDR100: Unknown error

101	CDR101: Not ready
102	CDR102: EMS memory no longer valid
103	CDR103: CD-ROM not High Sierra or ISO-9660 format
104	CDR104: Door open

Special Considerations When Using a Read-Only Device

When developing software that is used with both read/write and read-only devices, your application should be prepared to handle the MS-DOS function responses from a read-only device if it attempts to write to it. This section summarizes how MSCDEX responds to the common functions used to update files or directories.

INT 21h Function 3Ch - Create File
This function always sets the carry flag to indicate failure.

INT 21h Function 3Dh - Open File
This function fails with write access and when attempting an open with a non-existent file. To prevent compatibility problems, it does not fail when opening for read or for read/write access. Note that even when the file is opened for read/write access, any attempts to write to the file (INT 21h Function 40h) will fail.

INT 21h Function 4301h - Setting File Attributes
This function always sets the carry flag to indicate failure.

INT 21h Function 56h - Rename File
This function always sets the carry flag to indicate failure.

INT 21h Function 40h - Write File or Device
This function always fails. This is true even if the file was opened for read/write access.

INT 21h Function 39h - Create Directory
This function always sets the carry flag to indicate failure.

INT 21h Function 3Ah - Delete Directory
This function always sets the carry flag to indicate failure.

INT 21h Function 6Ch - Extended Open File
This function fails with write access and when attempting an open with a non-existent file. To prevent compatibility problems, it does not fail when opening for read or for read/write access. Note that even when the file is has been opened for read/write access, any attempts to write to the file (INT 21h Function 40h) will fail.

INT 21h Function 13h - Delete File (FCB and Extended FCB)
This function always sets AL = FFh to indicate failure.

INT 21h Function 16h - Create File (FCB and Extended FCB)
This function always sets AL = FFh to indicate failure.

INT 21h Function 17h - Rename File (FCB and Extended FCB)
This function always sets AL = FFh to indicate failure.

Special Considerations For Other MS-DOS INT 21h Functions

In most instances, file-based MS-DOS INT 21h functions behave identically to those used with any other media. Drive- or device-based MS-DOS INT 21h functions are a different matter. Since MSCDEX relies upon the MS-NET interface to DOS, many of these MS-DOS functions will

behave as if the CD-ROM were a remote drive. With other MS-DOS functions, the MSCDEX redirected drives will not respond as a network drive would in order to preserve some characteristics of removable local media.

INT 21h Function 42h - Set File Pointer
As a note, repositioning the file pointer (seeking) does not translate directly to moving the actual CD-ROM device head location. The CD-ROM drive might not reposition the heads until data is read from the disc.

INT 21h Function 44h (IOCTL) Subfunction 08h - Check Block Device Removable
This function fails when checking for a drive supported by MSCDEX, but since it checks as being remote this is correct.

INT 21h Function 44h (IOCTL) Subfunction 09h - Check Block Device Remote
Checking for a CD-ROM drive returns as a remote drive and fails when checking for removability. This is similar to a network drive; however, note that the CD-ROM drive does not appear on the network redirection list.

INT 21h Function 47h - Get Current Directory
After a disc is ejected, this function will return the directory selected for the last current directory. Thus, this function is not a reliable indicator to determine if the media is present in the drive or if the directory is still valid. For a reliable indication, execute the Volume Label function (INT 21h Function 440Dh Minor Code 66h) prior to executing this function.

INT 21h Function 5Fh Subfunction 02h - Get Redirection List Entry
The logical drive supported by MSCDEX does not show up in this list under any of the indexes. To this respect the CD-ROM drive behaves like a local drive; however, the drive checks as remote making it behave somewhere between a network drive and a local drive.

End.

Networking CD-ROMS

Although it is possible to share CD-ROM drives on a local area network or LAN, it is not as easy as it should be. While MS-DOS provides a single, stable platform to develop a file system driver like the Microsoft CD-ROM Extensions, there are a wide variety of LANs and LAN server implementations that do not. Because each LAN implementation takes a different approach for server support, the approach for CD-ROM support on a server depends on what LAN implementation you are using. This document should help clarify the present situation and help get you started.

At present, there are several CD-ROM products that allow sharing of CD-ROM drives on a LAN. These are:

- MicrosoftMSCDEX (The Microsoft CD-ROM Extensions)
- Meridian DataCD-NET
- OnlineOpti-Net

Choosing which product depends on your LAN and your needs.

Using MSCDEX With an MS-DOS Based LAN

There are some LANs, such as Lantastic by Artisoft, that can share CD-ROM drives using MSCDEX on a server. This is possible because these servers run as an MS-DOS application and use standard MS-DOS INT 21 services for I/O. LAN servers that use standard MS-DOS INT 21 services to access the drive letter, and do not make assumptions about the underlying media or try to bypass MS-DOS, will likely work with all versions of MSCDEX.

Using MSCDEX With an MS-NET Type LAN

There are several LAN products based on MS-NET or a similar LAN server model (for example, Ungermann-Bass or 3COM) that can use MSCDEX Version 2.10 or later to share CD-ROM drives. The MS-NET based products do not access files on the server using standard INT 21 calls and, because of some assumptions MS-DOS makes about non-standard calls from the server, you cannot share CD-ROM drives on MS-NET based servers using earlier versions of MSCDEX. Although the server seems to allow sharing of the CD-ROM drive letter, requests to the server from workstations do not work correctly.

Starting with MSCDEX Version 2.10, the /S command line switch instructs MSCDEX to patch the in-memory image of MS-DOS so that it will work with MS-NET based server software. With this switch on the MSCDEX command line, load MSCDEX after the network redirector but before the network server software. The CD-ROM drive letters can then be shared by MS-NET based server software, and workstations will see the correct behavior. For this solution, only the server needs to run MSCDEX or load any CD-ROM related device drivers. You do not need to make any software or hardware changes to the workstations. To a workstation, the CD-ROM drivers connected to the server are indistinguishable from other server drives.

Using MSCDEX on LANs With NETBIOS Support

For LAN products that are not MS-NET based and have NETBIOS support (for example, Novell and IBM PC-NET), both Online and Meridian Data have adapted the MSCDEX and CD-ROM Device Driver model to provide LAN CD-ROM support. Each workstation runs MSCDEX and a

special CD-ROM device driver. The special device driver accepts normal CD-ROM driver requests from MSCDEX and uses the NETBIOS to transmit the command to a network driver on a server. The network driver submits the request to a true CD-ROM device driver on the server and transmits the results back to the workstation pseudo CD-ROM driver. The psuedo driver in turn responds to MSCDEX. So long as the workstation CD-ROM device driver responds appropriately, MSCDEX is unaware that the command has passed through the network to a server. Contact Meridian Data and Online for information for these networks as they can both describe their products and features best.

Online offers one potential configuration for computer systems that do not wish to dedicate a machine as a server. The workstation operates as above, but instead of communicating the workstations driver request to a dedicated server process, another user's workstation running a special TSR version of their system can field the driver request, submit it to the CD-ROM driver, and respond to the requesting workstation. This allows a network of workstations to share the CD-ROM drives that each computer has connected to it at the same time all workstations are available to the users. This option may work for many users although it does slow performance of the workstation when outside requests come in and uses memory for the TSR system code.

Support for OS/2 and Some Novell LAN Implementations

OS/2 servers (and some Novell LAN implementations) can integrate CD-ROM media into the networks in the same way it does other media. However, access to the device characteristics is abstracted and the INT 2Fh function 15h services are not supported.

Sources For CD-ROM LAN Products Not Based on MS-NET

For more information on CD-ROM LAN products for LANs other than MS-NET, contact:

Meridian Data Inc.
Ms. Candace Brown
5615 Scotts Valley Drive
Scotts Valley, CA 95066
(408) 438-3100
(408) 438-6816 (Fax)

Online Computer Systems
Mr. Mike Romanies
20251 Century Blvd
Germantown, MD 20874
(301) 428-3700

End.

DDK Product Overview

The Microsoft MS-DOS CD-ROM Extensions are an extension to the MS-DOS operating system which permit reading CD-ROM discs which conform to both the High Sierra format and the ISO-9660 version of the High Sierra format. The CD-ROM disc appears just like a read-only magnetic disk to the user and to applications software, ensuring compatibility with current software.

Product Components

This product consists of a program supplied by Microsoft to assist programmers in developing MSCDEX compatible CD-ROM device drivers. The program supplied by Microsoft is named MSCDEX.EXE. Technical documentation as well as a sample hardware-dependent device driver is also supplied by Microsoft.

Product Highlights of Version 2.21

The Microsoft MS-DOS CD-ROM Extensions Version 2.21 include these updated features:

- The MS-DOS version variables have been updated and MS-DOS 5.0 no longer needs SETVER to run MSCDEX.

- The program TESTDRV has been updated with a new release. This program provides a general test that very rigorously tests your driver against the MSCDEX specifications.

- The programs DOSSPEED and WINSPEED have been replaced with CDSPEED. CDSPEED checks the transfer rate of your driver under the MS-DOS environment. CDSPEED can be used to verify the the transfer rate of the Multimedia PC Specification.

- The section "MS-DOSifying Your CD-ROM" has been expanded to include information on writing applications for read-only media, trapping extended INT 24h errors, and a description of the behavior of INT 21h functions when used with CD-ROM.

Technical Overview

This product uses the Microsoft Networks interface to MS-DOS so it requires MS-DOS version 3.1 or higher. MS-DOS 3.1 virtualizes the interface to drives. The entire CD-ROM (potentially all 660 megabytes) will appear to applications as a single MS-DOS drive letter. The Microsoft MS-DOS CD-ROM Extensions provide a high degree of compatibility with applications that depend on MS-DOS standard interfaces. See the section "MSDOSIFYing Your CD-ROM" for more information on accessing files on the

CD-ROM from MS-DOS. The following list summarizes the characteristics of the MS-DOS CD-ROM Extensions:

Requires MS-DOS 3.1 or higher (or PC-DOS 3.1 or higher)

- Implements the High Sierra format and ISO-9660 format

- Requires a hardware-dependent device driver

The program MSCDEX.EXE is an installable file system driver implemented as a terminate and stay resident module. The user will usually load this program using AUTOEXEC.BAT when the computer is booted. (The user might also load MSCDEX with the INSTALL command in CONFIG.SYS or alternately load it directly at the command prompt.) The hardware-dependent device driver implements basic functions to read the CD-ROM disc and is loaded with the MS-DOS CONFIG.SYS file.

The Microsoft MS-DOS CD-ROM Extensions implement both the High Sierra file format and the ISO-9660 version of that standard. All features defined in May 28th proposal for Level 1 are implemented. In addition the following features beyond the High Sierra Level 1 standard are implemented:

- Support for CD-ROM XA Interleaved Files
- Support for 31 Character File Names when possible through truncation
- Support for Hidden Files
- Support for Access to VTOC
- Ignores Higher Level Files and Functions when present on the disk:

  - Associated Files
  - Protection Bits
  - Record Bits
  - File Version Numbers

- Support for shift-JIS Kanji (Japanese character) filenames

Hardware-Dependent Device Driver

This product requires a hardware-dependent device driver that interfaces to a specific OEM drive or drives. A detailed specification for the device driver as well as a sample driver are included. The driver implements the basic functions of reading the CD-ROM and is installed using the MS-DOS CONFIG.SYS conventions. A minimum set of functions for reading the CD-ROM disc are required to be in the device driver. Optional additional functions for increased performance when supported by the CD-ROM drive and controller may also be implemented in the driver. For more information on these functions, see the Hardware-Dependent Device Driver Specification section of this documentation.

Licensing the Microsoft MS-DOS CD-ROM Extensions

Microsoft will license the MS-DOS CD-ROM Extensions to manufacturers and marketers of CD-ROM disc drives. The license agreement allows the use of the product on a personal computer to which a licensed disc drive is attached. Developers of CD-ROM discs will not need to acquire any license or pay any royalty in order to develop or sell CD-ROM discs, and will not be entitled to distribute the MS-DOS CD-ROM Extensions. The end user will purchase the driver from drive manufacturer or marketer, not the CD-ROM disc developer.

The Microsoft MS-DOS CD-ROM Extensions will be delivered to licensees on a 3-1/2" MS-DOS diskette. Licensees are expected to distribute the Extensions to their customers on a floppy diskette containing both MSCDEX.EXE and the hardware-dependent device driver written by the licensee. The floppy would be included in the package containing the CD-ROM drive.

Creating CD-ROM Discs in the High Sierra Format

The Microsoft MS-DOS CD-ROM Extensions provide for reading CD-ROM discs in the High Sierra/ISO-9660 format on MS-DOS computers. They do not create CD-ROM discs in the High Sierra/ISO-9660 format. Microsoft does not manufacture CD-ROM discs, nor provide pre-mastering services. Third party companies can create CD-ROM discs in the High Sierra/ISO-9660 format and provide other pre-mastering services. Microsoft can supply a list of companies providing or planning to provide these services upon request.

Software developers do not need the MS-DOS CD-ROM Extensions to create either applications software that reads CD-ROM discs, or to create CD-ROM discs. Once the software is ready and a disc has been pressed, developers will want a copy of the Extensions for testing; however, they are not needed to start development.

Software developers need do nothing special for accessing CD-ROM discs; they issue the same MS-DOS OPEN and READ calls as for opening any magnetic disks. Programmers can develop CD-ROM applications using standard MS-DOS tools. They need to be aware that they cannot create any temporary files or write any files in either the directory or on the entire CD-ROM disc. Software developers will want to minimize the number of seeks to the CD-ROM because of the comparatively long seek times of CD-ROM drives.

End.

Questions and Answers

Q: Can you provide more info on how the READ/WRITE device control string should work. Does the read device bytes command get information that was written by the write device control string?

A: At present, there are very few people that use this command. There are a couple other features that are either used only in special instances or not at all at present. This is not to say that they won't be used at some later time.

The purpose of these commands was to allow a standard way of delivering commands that were not specified in the CD-ROM device driver spec to the drive. For example, sending SCSI command strings and reading the responses from the drive. This function is deliberately open-ended and vague because it was intended to provide a catch-all mechanism for application programs to communicate requests or request data in ways that were not specified by the device driver spec. For application programs to use these functions they have to know the driver supports these functions and also how to communicate with that specific drive. The mechanism would let the driver do what it does best and worry about which ports and interrupts to use. This relieves the application program from these details and allow it to deal with controlling the device at a higher level.

Right now, if the driver does not support these functions, it should return an error for Unknown Command. One could test whether these two function were supported this way. Note: if there are commands which you feel should be supported by the device driver specification, please communicate them to us and we will consider adding them if they are of sufficient general interest.

Q: How can an application access CD-ROM drives that are subunits of one driver? The IOCTL calls do not take an argument for subunit. MSCDEX seems to handle this OK since when I do a directory of each CD-ROM in turn it accesses the correct drive. I do not see any clean way for an application to, for example lock the door on CD-ROM drive G: which is the third drive handled by the driver.

A: Requests all have a sub-unit field in the request header. Commands that one would expect to be directed to a specific drive, such as open door, are targeted at a particular drive through the use of the sub-unit field.

Q: Why doesn't MSCDEX allow IOCTL access via the drive letter (that is, MS-DOS func. 44h subfunc. #4,5), as if the CD-ROM were a normal drive. I understand that the driver is not a block device, but this is being handled already in some way since you allow a user to perform file I/O to a CD-ROM making it appear to be a block device. It would seem that all that would be necessary to accomplish this is to intercept IOCTL calls in the same way that file access calls are being intercepted.

A: MSCDEX doesn't presently hook INT 21h, which is what this would involve. It's doubtful that this will change. It's not that much more difficult to open the file and send an IOCTL to the handle. File access calls are not caught at an INT 21h level but are caught from within MS-DOS at another interface. CD-ROM drives are far more like network drives than traditional MS-DOS FAT file structure block drives and their drivers. For example, try to FORMAT a CD-ROM drive and you'll see. Part of all this prevents IOCTL's to the drive letter from being directed to the appropriate driver. For more information, see the section MS-DOSifying Your CD-ROM.

Q: Why not allow access to the PLAY, STOP and SEEK functions via the INT 2Fh entry point as is allowed for READ LONG. This would be much simpler than requiring the application to locate the driver header and then find the STRATEGY entry point and create request control blocks etc. This is a lot of code to start the music playing!

A: The reason we haven't included play and other commands of this type in the INT 2Fh interface is to avoid loading down MSCDEX with additional functionality that most people don't use. Your suggestion would only move that code from the CD-playing program into MSCDEX. It makes your program smaller at the expense of making MSCDEX larger.

Q: Shouldn't the specification eliminate the need for the application to OPEN the driver by name? This is especially important in systems where the driver creates a new driver header for each CD-ROM drive. As it is, MS-DOS allows so few file handles to be open simultaneously that requiring applications to open even more is very bad.

A: Simply close the driver handle after you have located the device header. You no longer need to communicate through MS-DOS to control it, so free the handle and make it available for other programs to use. With version 2.10, it is no longer necessary to OPEN the device driver in order to communicate with it. Applications can communicate with the device driver using *Send Device Driver Request.*

Q: It seems that there should be bits in the Device Status word to indicate whether a driver supports Reading/Writing device control strings.

A: Reading and writing device control strings was put in as a catch-all for anything that was missed so that application programs could send specific commands through the device driver to the device if they understood the device and knew how to communicate to it. A manufacturers CD-ROM diagnostic program would be an example of a program that might choose to communicate with the drive in this way. If the driver does not support these functions, it should return an error. One can test whether these two function are supported by testing if the error returned is for *Unknown Command.*

Q: In the spec, treatment of the BUSY bit in the status word with regard to the PLAY AUDIO function seems to assume only one CD-ROM drive. What happens when the user has two or more drives each of which want to be playing music? How does the application tell whether the desired drive is busy? It would seem better to use some of the bits in the upper word of Device Status to indicate BUSY for each drive, perhaps allowing 8 or 16 drives.

A: Requests all have sub-unit numbers associated with them. A request for service from one sub-unit may report that the drive is busy at the same time another sub-unit was not busy. The sub-unit field is used to distinguish requests between the drives supported by the driver. The busy bit serves as an indication of drive status for the sub-unit the request is for.

Q: In the device driver spec, it says that if more than one unit is supported by the driver that this field should be set to the number of units. I suspect that this is wrong since this is not a block device. As far as I can see, this field should only ever be set to one since each unit will actually have its own header with its own unique name.

A: CD-ROM device drivers are a hybrid of block and char device drivers and are not technically legal as one or the other. Block drivers make some assumptions about the media format that aren't meaningful for CD-ROM and don't have a read call that can deal with CD-ROM's large data space. They were made as char devices with some additional calls and rules. One of the changes that was made for CD-ROM device drivers was to allow multiple sub-units for the device so the treatment of this field is correct as specified even though CD-ROM device drivers are character device drivers.

If one has more than one CD-ROM drive, one can approach supporting them from several ways. One could have separate device drivers for each drive and load one per drive, have a single driver with multiple device headers, or have a single driver with one device header that supports sub-units. This last method is borrowed from block drivers. For the case that the drives and drive commands are all the same, using sub-units will allow you to distinguish between which drive receives which command. The alternatives clutter MS-DOS up with drivers or device headers. Sub-units may not be legal character device driver fields but conceptually, they're the right thing. Since CD-ROM device drivers could not be block drivers and had to be char device drivers, some liberties were taken with the specification to merge the best of both specifications.

Q: Is there any support through MSCDEX for WRITE LONG? I have a need for this to support a CD mastering system. I would like to be able to treat a WORM drive as a CD-ROM and allow writing to the drive once to create a master and then be able to test it out by using it as CD-ROM to verify that our data has been correctly stored in High Sierra format.

A: Such a call exists. It only serves to define a standard interface for CD-ROM device drivers that are running over re-writable media - such as a CD mastering system. It is in the driver specificiation starting with version 2.00 of the CD-ROM Extensions.

Q: How important is it that I should support RAW mode access in my driver? What would this typically be used for?

A: This is something that is gaining in importance. Several drives support reading in raw mode. Since drives and their command capabilities are hardware dependent, you would know based on the capabilities of your hardware if you were capable of supporting it. This function was added for completeness. A standard way was needed to define how to get at the 288 bytes of EDC/ECC if drives allowed access and to have an avenue prepared if people found useful applications that would not use EDC/ECC where they wanted the additional space (such as gracefully degrading low-fidelity audio or graphics). It will be useful for copying audio information or for audio systems that will want to be able to manipulate audio tracks. Many people have expressed interest in having this capability.

Q: In the structure for the UPC Code function, "The UPC/EAN code (last 4 bits are zero)". Does this mean the low order or high order 4 bits?

A: This is less ambiguous if you read Redbook under mode-2 of the Q-channel info. This is now clarified in the UPC Code call. It should be the low nibble of byte 7. Redbook specifies that MSB comes out first so the high nibble will contain the 13th nibble of the UPC code and the 14th nibble will be zero.

Unfortunately, scanning for the UPC code is time consuming especially if it is done by the software. This is due to the design of the q-channel in Redbook. It's a pity because this could be a useful number to verify the correct disc has been inserted. Most CD-ROMs do not have a UPC code or have it zeroed out. The same seems to be true for CD-audio as well. We believe that CD-ROM drives should scan for the UPC code as they read the Table of Contents when initializing from power up or a new disc. If the hardware does not do this, the UPC code has to be scanned by polling the q-channel which may occasionally miss the UPC code.

Q: It would be nice if the device driver spec included a list of what types of disk access functions would and would not work so that users could get an idea of what utilities and applications will and will not work with the extensions.

A: The device driver specification describes just what is necessary for writing a CD-ROM device driver. The information you would like concerning things such as INT 25h/26h not supported

as well as the behavior CHKDSK/FORMAT/etc belongs and is mentioned in the MSCDEX overview.

Q: If I have a low priority request and if the system has time, can the prefetch request read into and transfer the data into a transfer address?

A: We have looked at this for some time but the bottom line is that asynchronous I/O under MS-DOS is very difficult to support in all cases. It is difficult for MSCDEX or the CD-ROM device driver to know that the transfer address is still valid because MS-DOS never notifies MSCDEX or the device driver if the requesting process was been terminated. The request runs the risk of writing over another program. The best approach now is if the driver wants to, it can reserve internal buffer space for data from the disc and put prefetched data there. Then it can copy the data to the read transfer address once the read request finally arrives. Alternately, some of the caching or prefetching can reside in the CD-ROM controller or in the drive itself.

Q: Is there any status indication that a prefetch transfer has occurred or some interaction with the READ LONG command?

A: There is no way to tell if a prefetch request was successful or the state of it. The prefetch simply provides a hint to the driver and the read request later is the request that finally takes delivery of the data.

Q: My driver seems to work ok except that whenever I connect to a subdirectory and do a directory, I am suddenly back in the root directory again. What's going wrong?

A: What is most likely happening is the driver is returning an incorrect value for MEDIA CHECK and MSCDEX thinks that the disc is changing all the time. When this happens, MSCDEX rereads the volume descriptors and pathtable and reinitializes what it knows about the disc and changes the current working directory back to root as if the drawer had been opened, the disc removed, and then reinserted. This will be accompanied with a larger amount of disc activity than one would expect for a simple directory scan. Fixing the driver to return the correct value when asked for a media check will correct this behavior.

Q: What is the best way for my application to know if the disc has changed since it was last accessed?

A: Use the MS-DOS function find first and look for the volume id. When the disc has been read and MSCDEX has already initialized the internal information it keeps for each disc, this is a relatively inexpensive operation. The information is in memory and the disc does not have to be touched, so checking the volume id is very quick. Only if the disc has been changed does the disc have to be touched. This operation takes considerably longer than if the disc was not changed but even so, this has to be done anyway because MSCDEX has to read and initialize what it knows about the new disc so it can report the volume id correctly so the application can know if the disc in the drive is the one that it is looking for.

Q: When I do a directory, the first couple filenames are either duplicated or they are random characters. What might cause this?

A: The problem comes from having the incorrect bytes in the file identifier field for the first two directory entries. The first directory entry in each directory file is supposed begin with a copy of the directory record for that directory file followed by a copy of the directory record for the parent directory (also known as '.' and '..' on Unix or MS-DOS). The filename or directory identifier is supposed to be 1 byte long and the contents are supposed to be 0 for the first directory entry and 1 for the second directory entry. This is discussed in clause 6.8.2.2 of the ECMA standard or the ISO-9660 proposal. Several places on the disc in question, you have a

1 where there should be a 0 and in one directory, the file identifier consists of 0x8A which is why DIR in that directory begins with an "e". Incorrectly formatted discs will not be handled by the extensions correctly. This is why it is a good idea to test your disc image using MSCDEX before you press a disc to make sure your data is formatted correctly and as MSCDEX expects it.

Q: I have a directory file that is 4 kilobytes long but when I do a DIR in that directory, it is slower than usual and random filenames are printed out. I can tell by watching the device driver commands that MSCDEX is asking for sectors far beyond the end of the directory. I can see how this might account for the random filenames but why is it scanning so far?

A: Problems such as this result from having with multi-sector directory files that include empty sectors in the directory file. The High Sierra specification does not allow you to have empty directory sectors at the end or to have gaps in the middle. However, ISO9660 does not have this constraint. (This will be addressed in a later version.) The problem stems from the fact that your directory length is too long. For example, for the disc in question, the root directory begins at sector 28 and its length is 4096 bytes but the second sector is completely blank (all 0's). This confuses MSCDEX because it does not expect to see empty sectors.

Q: I noticed that Function Request 0 Get Number of CD-ROM drive letters may not always return unambiguous results. Suppose I start the network first and use one of the drive letters for a network drive (F:). When I start the Extensions, it will begin assigning drive letters after the last used drive letter (C: on my machine). If I have 4 CD-ROM drives on my system, they will be assigned drive letters D:, E:, G:, and H:. Function 0 returns 4 in BX for the number of CD-ROM drives and 3 in CX for drive letter D: correctly. But as you can see, the CD-ROM drives do not use contiguous drive letters so I cannot deduce from what this function returns that drive F: is not a CD-ROM drive.

A: That is correct. This is why function 0Dh Get CD-ROM drive letters was added. To get an unambiguous list of CD-ROM drives, use this function or use function 0Bh CD-ROM Drive Check to tell if a drive letter is for a CD-ROM drive.

Q: Is it possible to do an absolute read using the Extensions. I am trying to read mode 2 (uncooked) data using Function Request 8 Absolute Read. I use a normal device I/O to turn off error correction and perform a read but all I get back is 2048 bytes of data instead of the full 2356 bytes. Is there another way in INT 2Fh to get the data uncooked?

A: Not at present. If you want to get at the data including error correction code, you will have to communicate directly with the device driver. The Extensions provides the *Send Device Driver Request* mechanism for communicating with the device driver.

Q: Is it possible to access a non-High Sierra disc with the Extensions using an absolute disc read?

A: One can use either the extensions to read a non-High Sierra disc using INT 2Fh or one can communicate directly with the device driver to do this. The device driver itself makes no distinction between High Sierra and non-High Sierra discs so it can be used to read them although the burden of file system translation and reading then falls on the application talking to the driver. The INT 2Fh Absolute Read function simply packages the request to read and sends it directly to the driver and returns the result.

Q: What we have done is, in AUTOEXEC.BAT, first loaded the MS-DOS CD-ROM Extensions and then the MS-NET software. The error message is "Redirector already installed". The network software is then not loaded. We are using MS-NET 1.1 in an HP product called ThinLAN. Any hints as to what they should try next?

A: MSCDEX is a CD-ROM "redirector". It hooks into MS-DOS the same way the network redirector does to get requests for file access to files that are not on local hard/floppy disks. As far as MS-DOS is concerned, CD-ROM drives look just like network drives. MS-DOS passes all file accesses through the redirector interface to the network redirector which in turn sends file access requests out over the net. MSCDEX splices itself in front of the network redirector and takes requests belonging to CD-ROM drives and passes the rest to the network redirector.

The problem is that the network redirector code assumes that there will only be one redirector installed (itself) whereas MSCDEX does not make this assumption. If the network redirector is installed after MSCDEX (before it in the interrupt chain), it will process all requests from MS-DOS and never pass any CD-ROM requests through to MSCDEX. For this reason, MSCDEX has to be installed after the network redirector (before it in the interrupt chain) and so MSCDEX prevents the network redirector from installing afterwards to ensure this. Since you installed MSCDEX first, the network believes a redirector is already installed so it does not install itself which is what you are seeing. In order to install both, simply install your network software first and MSCDEX second and you're set.

Q: CHKDSK, ASSIGN, and SUBST report that the CD-ROM is a network disc. Why is this?

A: From the above explanation, you understand that to MS-DOS, the CD-ROM drives look like network drives. The programs CHKDSK, ASSIGN, and SUBST check the same fields MS-DOS does and think the same thing. There is no way to get around this.

Q: RENAME gives error message "Duplicate file name or File not found" instead of something that makes sense such as "Access denied" or "Can't rename CD-ROM files".

A: The error message is coming from the code for RENAME and not MSCDEX. The error condition is being returned correctly but the error code returned by version 1.01 is correct according to MS-DOS documentation. The problem seems to be that there are two error codes for access denied - 5 and a special one 65 which is error_net_access_denied which is returned by the network redirector when it has a problem. MSCDEX version 2.00 and up returns error code error_net_access_denied and so RENAME now reports "Access denied".

Q: Why does the SETVER command have to be used when MSCDEX 2.20 is used with MS-DOS 5.0?

A: MSCDEX 2.20 was completed in the early phases of MS-DOS 5.0 development, thus, the capabilities of MS-DOS 5.0 were not available for MSCDEX development. While MSCDEX 2.20 was developed in advance of MS-DOS 5.0, its release however,was delayed until a few months before that of MS-DOS 5.0.

Q: Why can't I load MSCDEX 2.21 in high memory using MS-DOS 5.0?

A: MSCDEX Version 2.21 cannot load high because it is essentially an upgrade to improve the interface between it and MS-DOS 5.0. A future release of MSCDEX will take advantage of the features available in MS-DOS 5.0.

While you cannot load MSCDEX 2.21 in high memory, you can still use the /E command line switch to have MSCDEX use the expanded memory installed in your system. This moves a portion of MSCDEX into expanded memory which frees some of the lower memory normally used by MSCDEX.

Q: Where can I get more information on CD-AUDIO, CD-ROM devices, and MS-DOS device drivers?

A: ISO 9660 specifies the standards of the volume and file structure used by CD-ROM discs. ISO 10149 specifies the data characteristics (such as the format of the tracks, error-detecting and error-correcting characters, and coding of information) of CD-ROM discs. IEC 908 specifies the standards used by CD digital audio system. These specifications are available from ISO (International Organization for Standardization) and IEC (International Electrotechnical Commission).

For information about MS-DOS drivers, you will find the following books helpful:

Duncan, Ray. *Advanced MS-DOS Programming.* Redmond: Microsoft Press, 1988

Microsoft MS-DOS Programmer's Reference, Redmond: Microsoft Press, 1991.

Also, the MS KnowledgeBase topics of The Microsoft Programmer's Library CD-ROM contain answers to many of the common questions asked about programming under MS-DOS.

Q: What is the current support for CDROM XA in MSCDEX?

A: CDROM XA is a novel definition of the use of Mode 2 sectors on a CD. MSCDEX supports plain ISO 9660 file access to files with XA system use fields. It is the responsibility of the driver and hardware to process or control the XA subsystems. A driver that has such specialized support will have bit 10 set in the DEVICE_STATUS (IOCTL Read (03h) subfunction 06h) return value.

End.

Measuring Your CD-ROM Speed

This document describes the file transfer test program CDSPEED. This program characterizes the performance of a CD-ROM drive, controller, and device driver.

About CDSPEED

CDSPEED, a performance monitoring and evaluation tool, measures CPU utilization when transferring data from a CD-ROM drive at sustained data transfer rates into main memory. CPU utilization is an important measurement that indicates:

- the CD-ROM drive's ability to transfer data at sustained data transfer rates
- the efficiency of the CD-ROM device driver
- the CPU bandwidth available to process the data after a read request and before the next required read operation

CDSPEED is a DOS application. It will not run from inside a DOS window when Windows runs in the enhanced mode. CDSPEED is intended to be an indicator to assist you in evaluating overall system performance. It can help identify key areas in the device driver that might need to be tuned to optimize performance. By specifying the desired sustained transfer rate and data block sizes, you can characterize the CD-ROM drive over a range of operating conditions. You can have CDSPEED display the results in either detailed or terse formats.

The Need for Speed Tests

Advantages of CD-ROM drives include not only the ability to store large volumes of information but also the ability to transfer data off the CD at a sustained data transfer rate. Transferring data at a sustained rate is called *data streaming* and is measured in the number of kilobytes per second transferred. For CD-ROM drives, this value is typically 150 kilobytes per second.

Many applications, such as those for multimedia, rely upon the data stream coming off the disc at a guaranteed rate of 150 kilobytes per second. While the transfer rate on any one drive is fairly constant, it is often less than 150 kilobytes per second; sometimes it is much less. If the designers of an application decide to support a drive which cannot maintain 150 kilobytes per second, then they have to decide the minimum transfer rate they will support. To intelligently select the minimum transfer rate, they need information about sustainable transfer rates. Therefore the sustainable transfer rate of your drive is important information which you should provide to your customers.

Multimedia PC (MPC) Speed Requirements

You can use the CDSPEED test utility to test your CD-ROM drive against the speed requirements for the MPC specification. The MPC requirements for CD-ROM drives are:

- sustained 150 kilobyte/second transfer time
- 1 second maximum seek time

- 10,000 hours MTBF
- mode 1 capability (mode 2 and form 1 and 2 optional)
- subchannel Q (subchannels P and R-W optional)
- MSCDEX 2.2 (or greater) driver validated by Microsoft
- physical capability of vertical or horizontal orientation

The drive must be capable of maintaining a sustained transfer rate of 150 kilobytes per second without consuming more than 40% of the CPU bandwidth in the process. This requirement is for read block sizes of between 8 kilobytes and 24 kilobytes and lead time of no more than is required to load the CD-ROM buffer with 1 read block of data. We recommend that the drive have on-board buffers of 64 kilobytes and implement read-ahead buffering.

Operating Environment for CDSPEED

Your computer will require the following software to run CDSPEED:

- MS-DOS Version 3.31 or higher
- MSCDEX Version 2.20 or higher
- OEM Device Driver for the CD-ROM drive

Installing CDSPEED

Install CDSPEED and other related files onto your system by copying the distribution disk to your hard disk. The following procedure places these files in a directory called CDSPEED:

1. From the DOS prompt, enter the following command and press ENTER to create the CDSPEED directory on your hard disk:

   MD \CDSPEED

2. Copy the files from the distribution disk to CDSPEED. The following command assumes the distribution disk is in drive A and the current drive is a hard disk:

   COPY A:*.* \CDSPEED

The following files are included in this package:

BLKTEST.BAT	Batch file that calls CDSPEED with varying equested block size
CDTEST.BAT	Batch file that calls BLKTEST.BAT with varying requested transfer rates
CHARTCD1.XLM	Excel 3.0 macro to chart actual transfer rate versus block size and overall CPU utilization versus block size
CHARTCD2.XLM	Excel 3.0 macro to chart background CPU utilization versus block size and CPU blocked by readings versus block size
CDSPEED.EXE	Test program

README.TXT Text file containing the latest information on CDSPEED

3. Change the current working directory to CDSPEED with the following command:

   CD \CDSPEED

Running CDSPEED

You can run CDSPEED directly from the DOS prompt or from the batch files included with CDSPEED. These batch files characterize the CD-ROM by running a standard series of tests that vary the requested data transfer rates and data block sizes for each test. After obtaining the initial information from the batch files, you might run CDSPEED directly with your selection of parameters to further characterize the CD-ROM drive in the areas that you are most interested.

Running CDSPEED From Batch Files

The batch files CDTEST.BAT and BLKTEST.BAT test a wide range of CD ROM performance parameters by successively calling CDSPEED.EXE. Test results from the batch files are stored in a text file that you specify.

CDTEST.BAT calls the second batch file BLKTEST.BAT passing it the requested data transfer rate in increments of 30 kilobytes per second. (CDTEST starts at 30 kilobytes per second and ends with 150 kilobytes per second.) CDTEST also passes the name of the test file located on the CD-ROM to BLKTEST.

During execution, BLKTEST.BAT calls CDSPEED.EXE and specifies the following command line options:

- Name of the test file located on the CD ROM
- Requested transfer rate (ranges in value from 30 kilobytes per second to 150 kilobytes per second)
- Requested block size (ranges from 4 kilobytes to 64 kilobytes)
- Requested primer size (Matches the requested block size)
- Terse output mode switch
- Name of the log file

These batch files perform a basic set of tests. To perform additional tests, edit the batch files--including the appropriate conditions for your tests. Or, run CDSPEED directly from the DOS prompt.

To run CDSPEED from the batch files:

1. Start at the DOS prompt. If you are running Windows in the enhanced mode, exit from the Windows Program Manager. CDSPEED needs a more precise timer than is available from the Virtual Timer Device.

2. Type the following command at the DOS prompt and press Enter:

   CDTEST <FileName> <LogFileName>

   Replace <FileName> with the name of the test file to be accessed from the CD-ROM drive. Specify a file of at least 1.5 megabytes to obtain accurate results.

   Also, replace <LogFileName> with the name of a file to hold the results of the test.

The following example runs CDSPEED from the batch files and specifies the D:\IMMPOST\CNTRL.C00 file as the test file accessed by the CD-ROM drive. (If you want to use CNTRL.C00 as the test file, it is located on the Windows with Multimedia extensions MODK CD-ROM.) The example stores test results in the TEST.LOG file located in the current working directory. This test takes approximately 2-1/2 hours to complete.

```
CDTEST    D:\IMMPOST\CNTRL.C00    TEST.LOG
```

Running CDSPEED From the DOS Prompt

If you are running Windows in the enhanced mode, exit from the Windows Program Manager. CDSPEED needs a more precise timer than is available from the Virtual Timer Device.

At the DOS prompt, type the following command and press Enter.

```
CDSPEED <FileName> <options>
```

Replace <FileName> with the name of the test file to be accessed from the CD-ROM drive. Specify a file of at least 1.5 megabytes to obtain accurate results.

The string <options> represents command line options that you can invoke with CDSPEED. To see a list of the CDSPEED options, run CDSPEED without using any command line arguments--no test file name and no options.

The following list summarizes the command line arguments available for CDSPEED:

FileName	Specifies the path and name of file to test. The FileName is required for operation of CDSPEED.
/r:[TransferRate]	Specifies the transfer rate in bytes per second. The transfer rate can range from 1 to 4294967 with a default of 150 kilobytes per second. Use this value to specify the sustained data read rate that might be assumed by an application.
/b:[BlockSize]	Specifies the number of bytes (BlockSize) read in each read request. BlockSize can range from 1 to 65535 bytes with a default of 10 kilobytes.
/p:[PrimerBytes]	Specifies the number of bytes used to prime the buffer. This parameter fills a read-ahead buffer before the transfer rate timing begins. This is useful when testing transfer rates approaching the maximum rate available for CD-ROM drives. The number of bytes range from 0 to 65535 with a default of 10 kilobytes.
/a:[PercentBlocked]	Specifies the maximum percentage of read time interval that CPU should be blocked to perform the read request of data. Default is 40%. This value is used for scaling performance measurements but

	otherwise has no effect on actual values returned.
/t	Requests terse output. The output is useful for processing and analyzing with a spreadsheet program. Default is verbose (non-terse) mode output. The order for the terse output data is: <FileName>, <TransferRate>, <BlockSize>, <PrimerBytes>, <%BackgroundCPU>, <TotalBytesRead>, <MeasuredTransferRate>, <%TimeBlockedByRead>, <%OverallUtilization>

The following table describes the fields of data returned in the terse mode:

Field	**Description**
FileName	Name of the file used during the test
Transfer Rate	Requested transfer rate
BlockSize	Block size used for test
PrimerBytes	Number of bytes used to prime buffer
%BackgroundCPU	Percentage of background CPU time
TotalBytesRead	Number of bytes read for test
MeasuredTransferRate	Transfer rate determined by test
%TimeBlockedByRead	Percentage of time CPU blocked by the actual read
%Overall Utilization	The overall percent utilization including background activity of the driver
/?	Displays the list of command line options.

The values for the /r, /b, and /p switches use bytes as the default units. For convenience, you can append k or s after the number to indicate other units:

k or K Kilobytes (1024 bytes)

s or S Sectors (2 kilobytes)

For example, /p:4s is equal to 8192 bytes (4 x 2048).

Reading the Output From CDSPEED

CDSPEED's performance information is displayed on the standard output in either a detailed (verbose) or terse format. Terse format is suitable for analysis and charting using a spreadsheet such as Microsoft Excel. The terse format is described with the /t switch of the command line options. The verbose mode displays a table containing the results of the test. The following entries are listed in the table created by CDSPEED:

Returned Value	Description
Test File Name	The <FileName> specified.
Requested Transfer Rate	The <TransferRate> specified with the /r switch.
Delay Between Reads	The allowed time between reads to attain the requested data rate.
Allowed Percent Block	The maximum percentage of read interval time the CPU should be blocked to perform the read request.
Read Block Size	The <BlockSize> specified with the /b switch.
Primer Size	The <PrimerBytes> specified with the /p switch.
Preparing for tests	Status message advising of the progress of preparation.
Priming XX kb--waiting YY ms	Status message advising of the progress of priming the data buffer.
Performing transfer rate tests	Status message advising of the progress of the test.
Total Data Read	The number of bytes read. The number of kilobytes and sectors read are listed in parentheses.
Total Time Expired	The total elapsed time for the test.
Reads Exceeding XX ms	The number and percentage of reads exceeding the percentage specified for PercentBlocked; or the number and percentage of reads exceeding the read interval.
Longest Time Blocked by Read	Specifies the longest time blocked during a read.
Shortest Time Blocked by Read	Specifies the shortest time blocked during a read.
Overall Transfer Rate	The measure transfer rate in kilobytes per second.
Percent Time Blocked by Reads	The total percentage of time the CPU was blocked by reads.
Background CPU Usage	The percentage of background CPU usage.
Overall CPU Utilization	The overall percentage of CPU utilization during data streaming.

Using the Results of CDSPEED

Overall CPU utilization is displayed as a percentage of total time used per read request. Part of this time is the time required for the CPU to service the CD-ROM's interrupt requests to keep the read-ahead buffers full. Since elapsed time includes time for the overall system transfer (such as the DMA transfer rate), performance will vary on identical processors on different machines with varying DMA hardware, bus clock speeds, and other concurrent DMA activity, if any.

CDSPEED determines several characteristics of a CD-ROM drive. Based on the results of

CDSPEED, you might create the following performance curves:

- Transfer rate as a function of block size
- Transfer rate as a function of primer size
- CPU usage (% time blocked) as a function of block size
- CPU usage (% time blocked) as a function of expected transfer rate
- CPU usage (% time blocked) as a function of both block size and expected transfer rate

How CDSPEED Measures the Results

CDSPEED simulates an application that requests data from a CD at a user determined sustained data transfer rate. CDSPEED divides the read data block size by the desired transfer rate to determine the amount of time needed to read a data block. This time is called the *read interval time.* (For example, a 150 kilobyte per second data transfer rate and 15 kilobyte block size yields a read interval time of 100 milliseconds.) The CD-ROM system must read a block of data within the read interval time for it to maintain the requested data rate. Based on the values in the previous example, if a CD-ROM system takes exactly 100 milliseconds to complete a request, the CPU will not have any time available to process the read data before performing the next read operation. In this case, the CPU utilization is 100%.

At every read interval time unit, CDSPEED issues a data read and transfer request and measures the amount of time it takes to complete the task. This time is measured as time that is blocked. While blocked, the CPU can do no other task until the transfer operation completes. Because some CD-ROM systems buffer their data, the time blocked by reads on these systems can be artificially low. In this case, this measurement determines the time to read data from the buffer rather than the CD-ROM. To account for buffering (and any other background CD-ROM operations), CDSPEED also determines the background CPU usage.

To determine the background CPU usage, CDSPEED performs an internal task prior to performing any CD-ROM reads. CDSPEED uses the time it takes to perform this task as a baseline measurement. During the read interval period after the data read, CDSPEED performs the same task and measures the time it takes to complete this operation. The baseline measurement is then compared to the elapsed time measurement and any difference is attributed to the CD-ROM device driver maintaining it's fully cached or read-ahead data buffers in the background. This difference is the background CPU usage measurement.

Note: On the very first read, CDSPEED issues a read request for 4 kilobytes of data and then waits for the device driver to return. After this read, the device driver continues to fill (or prime) its buffer with the data and CDSPEED waits for it to complete. Priming the buffer compensates for the amount of time it takes for the drive to perform an uncached read and seek. After priming, the drive should be ready for subsequent reads of data blocks from the cached memory buffer and to maintain data streaming at the sustained data transfer rate.

CDSPEED continues executing read requests at calculated time intervals to sustain the specified transfer rate until it reaches the end of the test file. It then produces the summary results.

Error Messages

The following section summarizes the error message CDSPEED can display. These error messages are directed to the standard error stream. For error handling in batch files, each error

returns a unique DOS error return code. This value is listed with the error description.

**** THIS PROGRAM CANNOT BE RUN IN A WINDOWS 3.X DOS VM ****

You cannot run CDSPEED in an enhanced mode Windows DOS VM. Timers accessed through the Virtual Timer Device in enhanced mode Windows do not have enough precision to obtain accurate results with CDSPEED. DOS error return code: 1

Switch requires ':' separator.

CDSPEED expected a colon ":" between a command line switch and the numeric value following it. DOS error return code: 2

Memory allocation error.
CDSPEED could not obtain enough memory to make the measurement. CDSPEED requests memory based on the block size and primer size. Try reducing these values if you need to run CDSPEED with limited memory.If this cannot be allocated, then you get this error. DOS error return code: 3

Memory free error.
CDSPEED could not free the memory it has allocated. This message will rarely occur. DOS error return code: 4

Read error!
CDSPEED encountered a read error during operation. DOS error return code: 5

Unknown arguments on command line.
The command line arguments contain an option CDSPEED does not recognize. DOS error return code: 6

No test filename argument supplied.
The command line arguments did not contain a name of a data file. CDSPEED uses this file as the source of the data it reads during the test. The test file should be greater than 1.5 megabytes. DOS error return code: 7

More than one test filename argument supplied.
CDSPEED interpreted more than one command line argument as a filename. Make sure that all numeric arguments have switches and check that spaces are not inappropriately placed in the command line. For example, check that the name used for the file does not contain spaces or the numeric values do not contain any spaces. DOS error return code: 8

Can't open specified file.
CDSPEED was unable to open the test file. DOS error return code: 9

File is too small for test.
Although 1.5 megabytes is recommended as the minimum file size to obtain the best measurement, CDSPEED can use smaller files. The smallest file CDSPEED can use is 4 kilobytes plus twice the blocksize. (CDSPEED uses 4 kilobytes for the initial prime. CDSPEED then uses two data blocks to obtain an average reading.) DOS error return code: 10

Delay value between reads is too small for 1ms timer resolution.
CDSPEED cannot obtain a valid measurement if the delay (the ratio of the block size to the transfer rate) between reads is less than twice the timer resolution. The delay must be greater than 2 milliseconds. DOS error return code: 11

Delay value between reads is too large.

The delay (the ratio of the block size to the transfer rate) between data reads is too large to obtain a valid measurement. The delay must be less than 10 seconds. DOS error return code: 12

Warning Messages

The following section summarizes the warning message CDSPEED can display. These messages are directed to the standard error stream. All of these messages return 1 for the DOS error return code.

MSCDEX is not installed.
CDSPEED did not detect MSCDEX which is required for operation.

Rate must be > 0 and < 65536, defaulting to 150k bytes/second.
The value specified for the /r:[TransferRate] switch is outside the valid limits.

Block size must be > 0 and < 65536 , defaulting to 10k bytes/read.
The value specified for the /b:[BlockSize] switch is outside the valid limits.

Primer must be >= 0 and < 65536 , defaulting to 10k bytes/read.
The value specified for the /p:[PrimerBytes] switch is outside the valid limits.

Percent blocked must be between 1 and 99 %, defaulting to 40 %.
The value specified for the /a:[PercentBlocked] switch is outside the valid limits.

File size should be at least 1.5 megabytes for accurate testing.
For the most reliable results, the test file used by CDSPEED should be larger than 1.5 megabytes.

End.

Setup for TESTDRV

TESTDRV is a rigorous test utility for CD-ROM device drivers to verify that the drivers adhere to specifications. This driver test attempts to fully exercise all possible calls to the device driver and record the driver's progress.

TESTDRV assumes that MSCDEX and the appropriate device driver are installed. During initialization, TESTDRV reads the driver profile from the file TESTDRV.PRO which assigns the device status defaults for the test. The following example shows a typical TESTDRV.PRO file:

```
; This is a sample TESTDRV.PRO
; Comments start with ';' and continue to the newline

DriverName= MSCD000      ; The driver to test (specified
                ; as argument to the
                ; <drivername>.SYS command line

WriteDevice= f        ; This device is not writable

Redbook     = t       ; This device supports Redbook
                ; Addressing

RawMode     = t       ; This device supports raw
                ; mode data

Prefetch    = t       ; This device supports
                ; prefetching

AudioControl        = t       ; This device supports audio
                ; channel manipulation

Audio       = t       ; This device supports
                ; audio/video information

AudioChannels       = 2       ; Number of supported audio
                ; channels

Interleave  = f       ; This device does not support
                ; Interleave mode

InterleaveSize      = 0       ; Interleave size (may range
                ; between 0-255)

InterleaveSkip      = 0       ; Interleave skip (may range
                ; between 0-255)

Eject       = t       ; This device supports software
                ; eject requests

UPC         = t       ; This device implements UPC code
                ; reading
```

```
Output        = HEXDUMP.TXT          ; Output hex dumps to this file.
                ; Blank assignment sends output
                ; to stdout

RedReadSectors     = 3:8:3,8:2:4      ; List of sectors to read in
                ; ReadL tests (Redbook form)

HSGReadSectors    = 0024180c,00ff3421 ; List of sectors to read
                ; in ReadL tests (HSG form) hex
                ; only
; <EOF>
```

If the profile variables are not set in the TESTDRV.PRO file, they will default to the values shown above (except for the sector selections).

Running TESTDRV

To run the test simply install your device driver, initiate MSCDEX, and execute TESTDRV.EXE. The default operation of TESTDRV can be modified through command line flags and arguments. Either a hyphen (-) or a forward slash (/) denotes the flags. The following command line flags and arguments are available:

filename	Alternate driver profile. (default: TESTDRV.PRO)
/A	Attended operation, qualifying interactive tests. (default: unattended operation)
/I	Override disk recognition on control disk. That is, behave as if the disk is unknown even if it is a member of the Test Set. (default: if recognized, several data matching tests are qualified).
/T	Terse output, no hex dumps and fewer diagnostic messages.
/[#]	Where # is a digit between 0 and 7, the drive number.

In unattended (default) mode, all tests will be verified by both successful completion, given an acceptable request, and successful error recovery, given an unacceptable request. The output has the following format:
[Command Code.Subcommand Code] [Status] [Command[:Subcommand]]:[Test Comment]

For example, the test for the location of the driver head may return:

```
3:12 TESTING   IOCTLI: QInfor:      BUSY:DONE:
3:1   TESTING   IOCTLI:LocHead:      BUSY:DONE:
  #1 Qinfo: Cntrl 1, Track 19, P/Index 1, Track Running Time 0:0:0
Disk running time: 47:35:0
Location of Head 47:35:0
```

Commands that return sector data or device dependent data will dump output in hexadecimal. If the disk is a recognized test disk and recognition is turned on (default), sector data will be compared to correct values and only the status returned.

Attended and Unattended Operation

Several calls to the driver cause or report physical changes in the drive unit or require that audio disk information be played through audio channels like conventional audio CD players. These states should be confirmed by an operator. A series of YES/NO queries and simple directions allow the operator to quickly step through these tests. In order to allow for operator-free testing, a set of alternate best-guess tests can be executed instead of the ones that require confirmation. Attended testing is a super-set of unattended testing and should be considered the most

complete run of the test program.

For example, the following sequence occurs in the attended mode:

```
132    TESTING       PlayReq:    BUSY:DONE:
Playing track from 47:35:0
    Can you hear music playing? [Yncq]_
132 PlayReq: Request Completed Successfully.
```

For a successful sequence, music would play and the tester would respond with 'Y'.

Control Disk Verification

The test for verifying read data requires the Microsoft Bookshelf and Microsoft Programmer's Library to be used as control disks. The test procedure reads data from the control disks then compares both raw and cooked data for correspondence with archived data. If the test is run without the control disks, the data read is dumped in hexadecimal and ASCII format to the specified output.

Nonstandard CD-ROM Features

Several driver commands derive their results or actions from hardware dependent features of the driver. Since not all drivers can be supported in a general release, special features of a device driver may not be adequately tested. (For example, write commands apply to few CD-ROM drives and are only minimally supported by error recovery tests.) If the hardware dependent CD-ROM device driver document describes the results of a driver request as undefined, the request will be tested for simple completion and error recovery. Requests that return data will dump the data to the selected output in hexadecimal and readable ASCII format.

Other Tests For CD-ROM Drives

CD-ROM drives are a natural companion to multimedia applications. The performance of many multimedia applications is dependent on the rate that data is streamed from the CD-ROM. You can use the CDSPEED program described in the next chapter to test the data rate of CD-ROM drives.

As the user base and popularity of the Multimedia Extensions to Windows expands, the demand for compatible CD-ROM drives will increase. The MUSICBOX application provided with Windows with Multimedia 1.0 provides a good platform to test the operation of you CD-ROM drive and driver. In addition to identifying incompatible behavior between a CD-ROM driver and MUSICBOX, the following tests help verify that the CD-ROM drive can properly play Redbook audio.

1. Load Windows with Multimedia 1.0 on your system.
2. Start Windows.
3. If necessary, remove the disc from the CD-ROM drive.
4. Start MUSICBOX and observe the operation of the system.

   Some drivers hang the system for a minute or more when MUSICBOX is started. The delay should be no more than a few seconds.
5. Put an audio CD in the CD-ROM drive with MUSICBOX going.
6. Seek to the next track with MUSICBOX and observe operation.

Some drivers do not support seek. Other drivers seek when playing but not when its stopped. Others seek when stopped but not when playing. Users expect the CD-ROM drive to seek when they are playing audio CDs.

7. Put MUSICBOX on repeat and make sure it repeats. Some drivers do not repeat.
8. Check that the status (time, track) is accurate.

   Some drivers pass inaccurate information back to MUSICBOX.
9. Play to the end of the CD and let it stop.

   Some drivers fail when they play to the end of a CD.

End.

Overview of Printer Font Metrics (PFM) Files

The information in this article applies to
Microsoft Windows Device Development Kit for Windows version 3.1

Summary

Printer font metrics (PFM) files are the generic way of presenting device fonts information to a printer driver. Device fonts are resident on the printer; they are not generated by Microsoft Windows. PFM files provide a device-independent method of representing device fonts so that Windows or a printer driver can represent device-font-metrics data.

Windows also supports printer-cartridge metrics (PCM) files. PCM files are a collection of PFM files. There are three tools that can be used to create PFM files:

- The PFM Editor, explained in Chapter 3 (page 25) of the Microsoft Windows Device Development Kit (DDK) "Printer and Fonts Kit." The PFM Editor creates PFM (or PCM) files that are compatible with Hewlett-Packard (HP) Printer Control Language (PCL) printers.
- The PFM dialog box in UniTool, explained in Chapter 7 (page 104) of the Microsoft Windows DDK "Minidriver Development Guide." The UniTool PFM dialog box creates PFM files that can be used with all minidrivers.
- The PFM file generator explained in Chapter 5 (page 53) of the Microsoft Windows DDK "Printer and Fonts Kit." The PFM file generator builds PFM files from recognized downloadable fonts. HP drivers use the PFM file generator.

More Information

A good overview of the data structures used in standard PFM files is given in:

- Chapter 2 (page 9) of the Microsoft Windows DDK "Printers and Fonts Kit"

An overview of the data structures used in PFM files generated by the Printer Font Installer (HP drivers) may be found in:

- Chapter 5 (page 54) of the Microsoft Windows DDK "Printers and Fonts Kit"

A description of the data structures used in PFM files generated by UniTool for minidrivers is presented in:

- Chapter 7 (pages 103-115) of the Microsoft Windows DDK "Minidriver Development Guide"
- Appendix C of the Microsoft Windows DDK "Minidriver Development Guide"
- The data structures used in PFM files for PCL printers are documented in:
- Chapter 4 (page 41) of the Microsoft Windows DDK "Printers and Fonts Kit"

The data structures used in PFM files for PostScript printers are reviewed in:

- Chapter 7 (page 98) of the Microsoft Windows DDK "Printers and Fonts Kit"
- Additional reference words: 3.10

End.

Printing

by Ron Gery

Abstract

This article concerns the basics of printing under Windows. It does *not* cover the setting up of a printer and its Device Context (DC), concentrating instead on operations needed to get output to the printer after it is set up. The use of the main printing functions, **StartDoc**, **EndDoc**, **StartPage**, **EndPage**, **AbortDoc**, and **SetAbortProc**, are discussed, as are the principles of banding.

Basic Steps

The sequence of events used for a printing operation are as follows:

1. Create the Device Context (DC) for the printer
2. Set an AbortProc to handle possible abortive conditions (for example, out-of-disk-space or user cancellation)
3. Start the document
4. Output to the printer DC and advance through the pages as needed
5. End the document
6. Clean up (printer DC, AbortProc instance)

This article assumes that the first step has already been accomplished. This step can be greatly simplified by using the **PrintDlg** common dialog.

New Functions for Windows 3.1

While the methodology of printing has not changed from Windows 3.0 to Windows 3.1, some new functions exist in Windows 3.1 to simplify the application interface. Under Windows 3.0, the **Escape** function is the primary means of controlling the flow of a print job. While this interface continues to function under Windows 3.1, a set of functions was added to GDI to separate the printing control operations from the device-specific escapes. The following table lists the 3.1 functions, the escapes that they replace, and their functionality:

3.1 function	***corresponding escape***	***purpose***
StartDoc	STARTDOC	start a document
EndDoc	ENDDOC	end a document
StartPage	implicit part of NEWFRAME and STARTDOC	start a new page
EndPage	NEWFRAME	end current page
SetAbortProc	SETABORTPROC	set the AbortProc for print job
AbortDoc	ABORTDOC or ABORTPIC	abort a document

The mapping between the 3.1 functions and the escapes is one-to-one with the exception of the NEWFRAME escape, which is replaced by two functions, **StartPage** and **EndPage**. The NEWFRAME escape's function was to end the current page and start the next page; the STARTDOC escape was responsible for indicating the start of the first page. The **StartPage** and **EndPage** functions work to better define the interface and are paired around every page being printed, including the first.

While both approaches work under Windows 3.1 for backward compatibility, applications written for Windows 3.1 and later are strongly encouraged to use the functions interface.

Very Simple Example

The following code demonstrates the bare essentials of printing using the Windows 3.1 functions:

```
// for a print session this small, the AbortProc won't actually come into
// play, but this is an example of a non-aborting version.
NonAbort(HDC hPrintDC, short uCode)
{
    return(1);                  // continue printing
}

// this routine prints a single 100x100 rectangle in the upper left corner
// of the page.
BOOL PrintRect(HDC hPrintDC)
{
    FARPROC lpAbortProc;
    DOCINFO diInfo;
    char DocName[5] = "hello";

    lpAbortProc = MakeProcInstance((FARPROC)NonAbort, hInst);
    SetAbortProc(hPrintDC, lpAbortProc);

    diInfo.cbSize = sizeof(DOCINFO);
    diInfo.lpszDocName = (LPSTR)DocName;
    diInfo.lpszOutput = NULL;
    StartDoc(hPrintDC, (LPDOCINFO)&diInfo);
    StartPage(hPrintDC);
    Rectangle(hPrintDC, 0, 0, 100, 100);      // actual output to printer
    EndPage(hPrintDC);
    EndDoc(hPrintDC);
    // clean up AbortProc instance
    FreeProcInstance(lpAbortProc);
}
```

This example performs the bare minimum needed for a print job. The abort procedure is essentially empty and does not allow for user input, no explicit banding is supported, and only a single page is printed.

Start and End of a Document

An application starts a print job using the **StartDoc** function (or the STARTDOC escape). The function accepts two parameters, the printer DC and a **DOCINFO** structure that identifies the document. The **lpszDocName** field specifies the name of the document; the name is used for display in the Print Manager window. The **lpszOutput** allows an application to specify an output file for the printer that could be different from the one set up in Control Panel. For example, if a printer is set up to output to "FILE:" an application can query the user for the output file name and use the actual name to start the print job. This avoids the invocation of GDI's dialog box requesting the name of the output file. After performing some spooler preparation work and bookkeeping, the DC is ready for real printing to begin. The state of the DC is saved at this point; it is restored by GDI for the start of every page and band. This is slightly different in Windows version 3.0 where the state of the DC is saved at **CreateDC** time instead.

With the escape-only method, STARTDOC also defines the start of the document's first page.

The **EndDoc** function (or the ENDDOC escape) marks the end of a print job. All needed clean up is done at this point, and the DC is restored to its state immediately preceding the call to **StartDoc**.

Start and End of a Page

The **StartPage** marks the start of a new page of output. Some bookkeeping is performed by GDI to prepare the DC for the page, but nothing really exciting takes place. The Windows 3.0

interface did not have the pure concept of a page start; it was implicitly defined by an output operation following a STARTDOC escape for the first page, a NEXTFRAME escape for subsequent pages, or by a NEXTBAND request after the previous page was done banding.

Predictably, the **EndPage** function denotes the end of a page. This is same functionality as the NEXTFRAME escape or of the last NEXTBAND escape on a page. At the printer level, this function causes a page eject, while at the GDI level a virtual page eject is performed by restoring the DC to its state at the time of the **StartDoc** call. If GDI is handling banding for the application (more on that below), the end of the page is when the actual banding work is carried out. The DC is now ready for another page of output.

Banding

Banding is the process in which a single page of output is generated using one of more separate rectangles, or bands. When the bands are all placed on the page, a complete image is the result. This approach is often used by raster printers that do not have sufficient memory or ability to image a full page at one time. Banding devices include most dot matrix printers as well as some laser printers.

An application determines if a printer is a banding printer by calling the **GetDeviceCaps** function with the RASTERCAPS index and then checking the RC_BANDING bit. When this bit is set, the device is a banding device. If the bit is not set, the driver outputs a full page at a time. Because GDI can transparently handle any necessary banding for banding printers, applications are not required to explicitly support banding regardless of how the RC_BANDING bit is set.

The advantages of using banding over allowing the GDI simulations are improved printing speed and possible reduction of disk space needed for printing. Speed can be increased by selectively performing output based on the band rectangles, thereby eliminating unneeded calls. Disk space is saved by avoiding GDI's banding support, which uses a disk metafile to represent the page image.

Standard

An application indicates that it intends to explicitly handle banding by calling **Escape** with NEXTBAND as its operation. This tells GDI that the application is doing its own banding as well as requesting the next band from the printer. In response to the NEXTBAND escape, the printer returns the bounding rectangle for the current band; all subsequent output to this band are automatically clipped to this rectangle. The dimensions of the bands depend on the driver and sometimes the amount of memory available in the system. When there are no more bands to process on a page, the printer returns an empty rectangle for the band. Notice that the rectangles do not necessarily band in the y direction; because the banding is usually performed based on the paper as it is situated in portrait mode, documents using landscape orientation may band in the x direction.

In order to explicitly use the device's banding, an application loops through the bands on a page and outputs to each band as needed. Because output is clipped to the current band, the application does not have to limit its output to the band's rectangle, but pre-clipping often speeds up printing. Once the empty band is encountered, the application moves to the next page.

The NEXTBAND escape is supported on non-banding drivers by defining a single band that encompasses the full page in order to remain consistent with the banding model. On these printers, an application gets two bands per page, the first one being the full page band and the second being the empty band that marks the end of the page.

At the start of the document, GDI calls **SaveDC** to save the state of the printer DC. For every NEXTBAND escape, GDI calls **RestoreDC** so that any attribute changes performed during the band are undone, and the DC is restored to its state at the time the document was started.

The following code uses the Windows 3.1 printing functions in conjunction with the NEXTBAND escape to print multiple pages of a document while banding each page:

```
ret = 1;                          // assume nothing has gone wrong
for (i = 1; (ret >= 0) && (i < NumPages + 1); i++)
{
    StartPage(hDC);
    // for real bands, output the document
    // once the empty band is encountered, end the page
    while ((ret = Escape(hDC, NEXTBAND, NULL, NULL, lpRect)) >= 0 &&
            !IsRectEmpty(lpRect))
    {
        PageOutput(i, lpRect);   // output to lpRect on page i
    }
    EndPage(hDC);
}
```

There is one difference in the way the Windows 3.0 interface handles banding that also applies under Windows 3.1 when the printing-specific functions are not used (i.e. only escapes are used). Basically, an application is not allowed to use *both* the NEXTBAND and NEXTFRAME escapes. An application that wishes to do its own banding cannot use the NEXTFRAME escape; in order to proceed to the next page, the application calls the NEXTBAND escape again after the last band (the one with the empty rectangle) of the previous page. Continuous banding causes page breaks when appropriate. Similarly, an application using the NEXTFRAME escape is assumed to be working in full-page mode, and it should not use the NEXTBAND escape. An application written for Windows version 3.1 should call **StartPage** and **EndPage** for all pages, regardless of whether or not it is banding.

Ignoring Banding

An application does not need to be aware of banding. By using only the **StartPage**/**EndPage** (or NEWFRAME escape) interface with no NEXTBAND escapes, an application indicates that GDI should handle all of the banding procedures. To accomplish this, GDI records all of the output calls into a metafile that represents that page. When the page is finished (**EndPage** is called by the application), GDI steps through the band on the page (using the NEXTBAND escape) and plays the metafile into every band. When the page's output is completed, the metafile is deleted and a new one is created for the following page. All of the metafile processes are completely transparent to the application.

Using a metafile to simulate a full-page image may suggest that the output capabilities are restricted by the limits of metafiles, but this is not the case. Because the metafile is associated with a real device, any operation that can be performed on a regular output DC can be performed on this metafiling DC. Query functions return values based on the printer, and functions like **DrawText** can be used because they are simulated using queried values. Also, the scaling limitations of metafiles do not apply since the metafile is never scaled-- it is played at its defined size.

There is one subtle limitation in GDI's banding support. If an application blts a color bitmap directly to a color printer, the metafile simulation does not produce the same results as manual banding. The metafile is built with a record containing a DIB representation of the bitmap; if the printer driver does not support the **GetDIBits** functionality, GDI simulates by building a monochrome DIB, and the color information is lost. An application that finds itself in need of blting a color bitmap to a color printer has to perform its own banding in order to achieve the best results. If color DIBs are used, this limitation does not exist.

The banding metafile is not used when an application performs its own banding.

Text vs. Graphics and the BANDINFO Escape

Certain printers, notably the HP LaserJet family under low memory conditions, separate output into two passes, one for text and one for graphics. After the text composed of built-in and downloaded fonts is placed on the page, a second pass is made to output graphics primitives to the page. While the difference between the two passes is meaningless for an application that performs a complete page drawing to every band, applications that wish to selectively draw to bands should be aware of this behavior. A text operation is any output performed using the **TextOut** or **ExtTextOut** functions.

The BANDINFO escape is used to get information about the band. Returned is a BANDINFOSTRUCT that identifies the band. The **bifName** field identifies the name of the band. The **bifType** field specifies the type of band. It can be one of BIF_ROCK, BIF_REGGAE, BIF_OOMPAH, BIF_JAZZ, BIF_CLASSICAL, BIF_METAL, or any of the other types that can be arbitrarily defined.

The BANDINFO escape is used to get information about the nature of the current band. When called after a new band has begun (after a NEXTBAND), this escape tells an application whether the printer is expecting text or graphics or both for this band. A driver does not have to support the BANDINFO escape. If the escape is not supported, this indicates that every band handles both text and graphics. To determine if an escape is supported by a driver, an application uses the QUERYESCSUPPORT escape. Also, a driver not supporting this escape will simply return 0 if the escape is attempted

The BANDINFO escape accepts two BANDINFOSTRUCT parameters, one for input and one for output. On input (the *lpInData* parameter), the application specifies what type of output it intends to put on this page and, if graphics are to be output, a graphics bounding rectangle. This input is only meaningful when used with the first band on the page. It is used by the driver to optimize banding-- no graphics bands will be used if they are not needed. If the application passes a NULL *lpInData*, the driver will expect a full page of text and graphics and band accordingly.

The driver uses the output BANDINFOSTRUCT (the *lpOutData* parameter) to specify the nature of the current band. If only the text flag is set (**fTextFlag** set to TRUE), the band is a text-only band. All graphics output is ignored for this band. If during the text band, text is encountered that requires glyphs to be generated in the graphics band (see below for more details), graphics bands will be used even if the application specified that only text is being output to the page (see above). In bands where the graphics flag is set (**fGraphicsFlag** set to TRUE), the printer is expecting graphics operations. Text operations that involve fonts used in the text band are ignored. If both flags are set, the band is a graphics band, and the printer is also expecting text output that could not be performed in the text band.

The complication in the scheme is that depending on the font being used, it is possible to output text during the graphics band. The example found in Windows 3.0 is vector fonts; for Windows 3.1, TrueType glyphs are output to the graphics band if they are rotated or if the font is larger than the threshold for downloading or if the user selected "Print TrueType as Graphics" during printer setup. As a result of this expanded definition of what text output means, printer drivers often set *both* the text and the graphics bits when the current band is a graphics band.

Applications should not make the assumption that a text band always exists. Given sufficient memory, the Windows version 3.1 driver for the LaserJet printers usually has only one band total that combines text and graphics.

Optimizing

By using the rectangle information that defines a band, an application can optimize its output to eliminate unnecessary work by the system. For example, if the current band covers the first 300 lines of the page, outputting a graphics primitive that appears between lines 600 and 900 results

in the system doing much work to get the primitive ready but ultimately clipping it out. An application that pre-clips its output to the bands can speed up its printing on a banding printer, especially if the output requires extensive simulations in GDI. On the system end of things, a graphics or text object is rarely clipped to the destination before final output time, long after any needed simulations have been performed.

It is valid and efficient to avoid graphics operations during the text band. On the other hand, an area that deserves caution is optimizing text output on printers that have separate text and graphics bands. Because text may appear in the graphics band depending on the font used, it is not valid to assume that text output can be ignored in a graphics band (unless all text was output in the text band and the **fTextFlag** was not set for the graphics band).

Scaling Factors

It is common for printers that use text and graphics bands to allow the two bands to be at different resolutions (the text band remains at the highest possible resolution). For example, a LaserJet configured for 75dpi will output the text band at 300dpi while outputting the graphics band at 75dpi. This discrepancy allows printer fonts to always look their best while at the same time speeding up the printing of graphics by using less resolution. This effect is achieved by using a different scaling factor for the two types of bands.

The GETSCALINGFACTOR escape returns the scaling factor for the current band. The value is an exponent of 2 (so that when a value at text resolution is shifted right by the scaling factor, the result is the value at graphics resolution). If the escape is not supported or the scaling factor is 0, there is no scaling taking place. In the LaserJet at 75dpi example, the first band (the text band) has a factor of 0 and all subsequent bands on the page (the graphics bands) have a factor of 2.

At the application level, all output is performed at full resolution at all times, and applications do not have to worry about the scaling factor. Coordinates are scaled when appropriate as part of the normal coordinate mapping done by GDI to convert logical units to device units. The only time the scaling becomes a concern is while performing pixel-specific processing, where 4 pixels at a logical 300dpi could map to a single pixel at 75dpi. Also, because clipping regions are specified in device units, a new clipping region designed for the scaled resolution needs to be built for the scaled bands.

Aborting a Print Job

A print job can be aborted in a variety of ways (discussed below), all sharing the same result. Any data that has not yet been sent to the printer is flushed out. Data that has already reached the printer cannot be recalled and is printed. The printer is notified of the abort and recovers as best it can. If the Print Manager is being used, the job is removed from the queue. If GDI was handling the banding, the banding metafile is deleted. The DC is returned to its state at the start of the document. The application is still responsible for freeing its AbortProc function instance.

AbortProc

The **SetAbortProc** function (as well as the SETABORTPROC escape) is used to set up what is known as the AbortProc. This AbortProc is a function that resides in the application and is called by GDI during a print job to inform the application of spooler errors and to allow the application to abort the job when desired. The AbortProc is called with information about why it is being called; this value is either an error code from the spooler or zero, which indicates that the function is being called simply to allow an abort.

The AbortProc is called routine during several steps of the printing process:

- after every write to the printer port when printing directly to printer (no spooling)
- after every write to a file when printing directly to a file (no spooling)

- after every write to the spooler file when spooling
- periodically when out of disk space for spooling due to other spool jobs
- before playing every metafile record when GDI is performing banding simulation
- occasionally from some older printer drivers

When the AbortProc is called, the application has the choice of either continuing the print job by returning a nonzero value or aborting the print job by returning zero.

Only the fourth case in the above list actually results from an error condition during printing. In this situation, the spooler is indicating that sufficient disk space will exist for the print job to complete once one or more of other print jobs currently being spooled are completed. By returning a nonzero value, the application indicates that it wishes to continue waiting for that disk space.

A big problem with the AbortProc given in the simple example above is that it does not allow other applications to run during the print job. A more system-friendly AbortProc processes pending messages before returning:

```
NonAbort(HDC hPrintDC, short uCode)
{
    MSG msg;

    while(PeekMessage(&msg, NULL, 0, 0, PM_REMOVE))
    {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
    return(TRUE);
}
```

Application Aborts

An application that decides to abort a print job on its own (i.e. without using the AbortProc) can do so by using the **AbortDoc** function or the ABORTPIC escape (also called ABORTDOC). This has the same effect as returning a zero from the AbortProc. This approach to aborting a print job makes sense if the application decides to abort while performing intensive operations where it would not be convenient to wait for the AbortProc routine to be called.

Using a "Print Cancel" Dialog

Most applications present the user with a chance to abort a print job by providing a dialog box with a cancel button or a variation on that theme. There are several time slices during a print job that an application can use for checking for a user cancellation of the printing. When the AbortProc function is called, the printing process is yielding to the application for exactly such purposes; this is a good place for checking for user input. When the application itself is performing some time intensive operation, it can yield to the abort-checking code when desired.

Once the user has indicated that the print job should be cancelled, the application has the choice of using its AbortProc or directly calling the **AbortDoc** function. The **AbortDoc** function should not be called from within the application's AbortProc.

System Aborts

If GDI is handling the banding for an application, there exists a possibility that the printing will be aborted by GDI due to system errors. In these cases, GDI sends an ABORTPIC escape to the printer driver and cleans up the current banding metafile. The application is notified of the abort with an error return from the **EndPage** function (or NEXTFRAME escape).

End.

Windows Font Mapping

by Ron Gery

Abstract

This article discusses the Windows font mapper and how it controls the realization of fonts. In the process, the article also looks at what it takes to effectively create a logical font so that the font mapping is predictable and useful. Some of the information is specific to Windows version 3.1, but most of it applies to both versions 3.0 and 3.1.

Introduction

An application requests a font by creating and selecting a font object. In creating the font, the application uses the **CreateFont** or **CreateFontIndirect** function to specify a list of attributes that define the font. A detailed explanation of the specific attributes is found in the SDK The resulting font object is called a *logical* font; it defines an idealized font that may or may not be available on a specific output device.

When the application selects this logical font into a DC using the **SelectObject** function, the font is matched to a *physical font*, one that can be output by the device. This process is called *realization* and is where font mapping takes place. The goal of the realization is to find a font available for the output device that most closely resembles the logical font. Determining this closeness is what font mapping is all about.

Note to the squeamish: in the case that the idealized font does not exist, the Windows font mapper is not perfect, the main controversy being the definition of the closest match. As a result, many disgruntled developers have referred to it with such crafty misnames as "font mangler" and the like. As with most software, the trick to using the font mapper is to understand how it operates and to work with it instead of against it.

An application can get data about the the physical font that is actually realized by using the **GetTextMetrics**, **GetTextFace**, and **GetOutlineTextMetrics** functions (the last one only with TrueType fonts). First select the logical font, then use these functions to get the details on the realized font.

For the duration of the article, the font's attributes are referenced using their field names as defined in the **LOGFONT** structure. While this structure is not a parameter to the **CreateFont** function, that function's parameters map directly to the structure. Inside GDI, all fonts are defined by a **LOGFONT** structure.

The "Right" Way to Create Fonts

The ideal way for an application to create a logical font is to define one whose attributes exactly match those of an existing physical font. This allows the application to control the mapping because it has essentially already picked which physical font it wants to use. Normally an application takes this approach by enumerating all of the fonts in the system using **EnumFonts** or

EnumFontFamilies (the latter is only available in Windows version 3.1). These functions enumerate the fonts by specifying for each physical font a logical font that exactly describes how the system identifies the physical font.

Font enumeration, like font realization, is based on a specific physical device identified by a DC. Different devices enumerate a different set of physical fonts, so an enumerated logical font from one device may not exactly match a physical font on another device. For example, a given printer may enumerate a hardware-based font named "Courier 10cpi," but that font has no exact match on the display device so the font mapper is forced to choose the closest match among the fonts available to the display. The application can "help out" in this process by choosing an ideal match from the fonts enumerated by the display device and using that font for selection into the display device.

Usually the desired font is chosen by the user, a process that can be greatly simplified by an application's use of the **ChooseFont** common dialog. This dialog presents the user with the list of enumerated fonts and returns to the application a logical font to use. The application can control exactly which types of fonts are presented to the user, thereby limiting the enumeration as desired.

If enumeration and exact picking is not an option, an application still needs to describe the font as unambiguously as possible in order to get a reasonable level of consistency in the mapping process. It is a good idea to always specify an **lfFaceName**, **lfPitchAndFamily**, **lfHeight**, and **lfCharSet**. The use and abuse of default values is discussed in greater detail below.

Point Sizes and Cell Heights

The Windows font interface deals with pixel heights of font cells instead of the more typographically traditional point sizes (1/72nd of an inch). Converting between the two measurements is not too hard, though:

```
CellHeight = (PointSize * VerticalResolution) / 72 + InternalLeading;

and conversely,

PointSize = ((CellHeight - InternalLeading) * 72) / VerticalResolution;
```

Where:

- *CellHeight* is cell height of font. This is requested with the **lfHeight** attribute and returned in the **tmHeight** field of the TEXTMETRIC structure after the font is selected.

- *PointSize* is the point size of the font.

- *VerticalResolution* refers to the resolution of the device being used. Windows provides two possible numbers for this value, neither of which is entirely accurate on display devices. Most applications use the convention of LOGPIXELSY (available via **GetDeviceCaps**), which is an idealized number of pixels per "logical inch" and is used in nonscalable fonts' resolution specification. The alternative is to compute the number of pixels per "physical inch" using VERTSIZE and VERTRES (also available via **GetDeviceCaps**). Thess values do not provide a hardware-specific value either (Windows has no knowledge of the physical size of the screen), but they may be a better approximation. With printer devices, the values for both logical resolution and physical resolution are usually the same.

- *InternalLeading* is a Windows concept that specifies the difference between a font's actual point size (also known as the *EmHeight*) and its physical cell height. This value is returned

in the **tmInternalLeading** field of the TEXTMETRIC structure after the font is selected.

Gee, so how does an application account for the internal leading value that is not available until after the font is selected? The simplest way to overcome the internal leading concept is to specify the **lfHeight** as a negative value. The font mapper treats the absolute value of that number as the desired point size (i.e. the value defined by CellHeight-InternalLeading) rather than as the cell height. To define a height for a font with a given point size, use:

```
MyFont.lfHeight = -((PointSize * GetDeviceCaps(hDC, LOGPIXELSY)) / 72);
```

Also, the TEXTMETRIC stucture is returned during font enumeration, so it is possible to anticipate a font's internal leading if a specific physical font is being tartgeted. Anticipation is not possible with scalable fonts because the fonts do not enumerate specific sizes and may not scale linearly,

The Mapping Process

Now that the logical font is built, the font is selected into a DC, and GDI calls the font mapper to perform the realization. Well actually because GDI caches physical fonts, if a logical font is being re-selected and the DC has not changed (same device, same mapping mode), there is no need for a full mapping, and the cached realization information is used instead. Unfortunately, this can't always be done.

There is a fundamental difference in the font mapping process between Windows version 3.0 and version 3.1. Both versions share a core mapper that uses weighted penalties to choose the closest matching physical font, but version 3.1 also has a shortcut layer that attempts to bypass the core mapper by looking for exact matches and only resorting to the core mapper when a certain level of exactness cannot be achieved. The basic idea is that if an application requests a logical font in the "right" way (see above), finding a matching physical font can be quick and to the point without excess processing; the shortcut layer is only for speed purposes, the functional behavior is unchanged.

Possible Physical Fonts

There are four types of physical fonts that are available to the font mapper, system-based raster fonts, system-based vector fonts, hardware-specific fonts found on the device, and for Windows version 3.1, TrueType fonts. Fonts generated by most bolt-on font packages like Adobe Type Manager (ATM) appear to the font mapper as device fonts.

The system-based fonts are the non-TrueType fonts that are installed via the Control Panel (TrueType fonts are also installed via the Control Panel). A few are installed automatically when Windows is set up. The raster fonts usually come in several sizes per facename, and because the fonts do not scale, these are the only sizes that are available for mapping. The vector fonts that come with Windows (Modern, Roman, and Script) are scalable fonts defined by polylines and are the only fonts that can be printerd on a plotter device. They are commonly referred to as the "stick fonts" because they are made up of line segments, which results in rather poor typographic quality. Most applications avoid the vector fonts by requesting a font with **lfCharSet** set to something other than OEM_CHARSET (specifically, ANSI_CHARSET or SYMBOL_CHARSET), the charset of the vector fonts, and it is common to see these fonts screened out of the font selection lists shown to the user.

Device fonts are those fonts that are provided by the device driver, either via hardware fonts or by downloading to the device. Usually these only exist for printer drivers, but bolt-on font engines also provide the concept for display drivers. Device fonts are controlled more by the device than

by GDI. GDI only has access to descriptive information about the font and not to the actual bit information, and more importantly, the device driver acts as its own font mapper to choose the best match among the available device fonts. During the font mapping process, the Windows font mapper only inspects the one device font that was chosen by the device driver. Windows sends the device driver the requested logical font, and the device chooses the best match from its available hardware and downloaded fonts. If the device has no font choice, there is no device font for the font mapper to inspect.

TrueType font technology is new for Windows version 3.1 and introduces to the system a class of fonts that are scalable and compatible with all raster devices. On printers, these fonts are either built into the hardware, downloaded to the hardware, or drawn to a bitmap which is then blted to the page. TrueType fonts, like raster and vector fonts, are added to the Windows environment via the Control Panel and managed by GDI. Because of their scalability, TrueType fonts are not penalized for height, width, or aspect ratio, but they are treated like the other fonts in all other respects.

The Core Mapper

The mapper itself, the big kahuna, compares each one of the possible physical fonts to the requested font. Each of the physical fonts is identified using a set of attributes that roughly parallel those found in the LOGFONT structure, and each attribute is compared to the requested attribute. There is a penalty assessed for each mismatch, and the penalties are accumulated. The mapper tracks the physical font with the lowest penalty, and once all of the physical fonts have been inspected, one font remains as the closest match. In case of a tie, the first closest match encountered is the one selected.

The interesting part, of course, is the relative size of the penalties. At the end of this article is a table that lists the specific penalties and their values. A quick inspection of this table reveals that the penalties are heavily weighted to discourage physical fonts whose major typographical features are not compatible with the logical font. The big ticket items are charset, output precision, variable instead of fixed pitch, facename, family type, and height. The lesser penalties only come into play in cases where the logical font is too vague to hone in on a desired physical font.

The height penalty is often a cause for confusion. The goal for the mapper is to pick a font that is as large as possible without being taller than the requested height. For example, when using raster fonts (height problems only apply to fonts that do not scale arbitrarily), if the logical font has a height of 11 and the available physical fonts have heights of 10 and 12, the shorter font is chosen. If the logical font has a height of 8 and there was no font shorter than 10, then the font of height 10 is chosen. This "under-sizing" scheme can get undesired results if a super-small font such as SMALLE.FON is available on the system because the font maps to a 6-pixel hight font that is not exactly legible. Then again, an actual small "filler" font may be exactly what the application wants.

In the case where two different fonts have exactly the same penalty, the first font that was inspected is the one that is selected. An application does not have control on this ordering. Non-device fonts are assessed an extra penalty to encourage the selection of a device font over an identical non-device font. The selection priority is 1) device, 2) raster, and 3) TrueType.

Shortcut Mapping for Verstion 3.1

New for Windows version 3.1 is an attempt to circumvent the above penalty scheme by searching for exact matches instead. For this algorithm, a physical font is an exact match if it shares the

following attributes with the logical font:

- charset
- face name
- height
- italicness
- weight

Because an exact match is the goal, GDI's italic and emboldening simulations are not considered.

The shortcut mechanism is only invoked if the device is a raster device that can accept raster fonts or a device that can output TrueType fonts (presumably by downloading). Also, the logical font must specify either a face name or use the OUT_TT_ONLY_PRECIS or CLIP_EMBED_PRECIS flags (in its **lfOutputPrecision** and **lfClipPrecision** attributes, respectively) in order to be considered a candidate for an exact match. Obviously if no face name is specified, then an exact match cannot be found, but the two flags indicate that only TrueType fonts should be considered for a match so some of the standard mapping process can be avoided. If the shortcut method cannot be used, the standard penalty-based mapper is used to pick the font.

Similar to the full font mapper, the shortcut inspects all the possible physical fonts for a match, but instead of tracking penalties, it throws out of consideration any candidate font that does not exactly match the above attributes. So if the logical font is based directly on an existing physical font, the matching is very simple, which is the entire goal of the shortcut.

If no exact match is found, the font mapping defaults to the penalty-based system. Ties (that is, multiple physical fonts providing an exact match) are resolved based on the filter setting of the **lfOutputPrecision** attribute. If no special filter is specified, the default behavior is that device fonts beat out everything else and raster fonts beat out TrueType fonts. The default behavior can be altered to have the TrueType font win a tie by using the TTIfCollisions entry in the [TrueType] section of win.ini. The possible filters are discussed in more detail below.

Extra Commentary

Below is a variety of issues and suggestions that affect the application's control of the font mapping process.

TrueType

The existence of the TrueType font technology in Windows version 3.1 changes the font mapping game to some degree. The basic mapping process remains intact, but the availability of more fonts and the ability of these fonts to be scaled to any desired size greatly increases the chance for a very close font match.

This is especially true for loosely defined fonts, those fonts that use a lot of default attribute settings. For example, an application that assumes a logical font created by specifying a height of 12 and setting all other attributes to 0 will map to the system font is making a very bad

assumption. While it might work under Windows version 3.0, under version 3.1 at least 6 fonts can match that request exactly even if only the base TrueType fonts are loaded. The one that is chosen by the font mapper is suddenly not so obvious and depends on such untangibles as the ordering of the fonts during initialization.

It is up to the application to be specific enough in its definition of the logical font in order to avoid mapping confusion. The first thing to consider is that TrueType fonts are fully scalable, so an apparently off-size height can be matchedthere is no penalty for height (or for width or aspect ratio).

Because most TrueType fonts are designed with a normal, italic, bold, and bold italic version, a request for one of these variations maps directly to a non-simulated variant of the standard font. Many fonts are also available in non-traditional weights such as light, demibold, and black, so the exact **lfWeight** that is specified could be meaningful. Lastly, an application needs to be careful in choosing the **lfFaceName** to deal with the potential explosion of face names. The **EnumFontFamilies** function is designed to help an application sort through the face names and their family relationships. In the eyes of the font mapper, the **lfFaceName** specified by the application can be either the full face name (for example, "Arial Italic") or only the family name (in this example, "Arial"). Assuming that all of the attributes match (in this example, **lfItalic** needs to be set), the two names are considered equivalent.

TrueType doesn't really introduce anything new into the font mapping picture, but it does make the choice of physical fonts large enough to necessitate an application to carefully define its logical fonts.

Substituted Face Names

Font face names are usually copyrighted strings, which adds a level of confusion in trying to identify various fonts that are produced by different vendors but look remarkably similar. Because much of the font mapping process relies on the exact use of face names, the font mapper has the concept of *face name substitution* to allow one face name to be substituted for another in the mapping process. This is a feature new to Windows version 3.1. For example, Windows version 3.0 had a raster font named "Helv" that is renamed to "MS Sans Serif" in Windows version 3.1. With version 3.1's built-in substitutions, a logical font with **lfFaceName** set to "Helv" is matched by face name to the physical font with the name "MS Sans Serif".

Face name substitutions are used only for raster and TrueType fonts and not for device fonts. Also, when a face name is matched via substitution, it is not considered an exact match a penalty is assessed for substitute face names, and an exact match always takes precedence.

The substitutions come from two places: one list is predefined in GDI and a second is defined by the user in win.ini under the [FontSubstitutes] section. The user-defined list can override GDI's predefined list to customize the Windows environment to the user's font needs. GDI's list consists of approximately 20 substitutions that map "Tms Rmn" to "MS Serif", "Helv" to "MS Sans Serif", and various PostScript names to the corresponding TrueType fonts that ship in Windows and in the Windows Font Pack.

Filters

An application can, to some extent, filter which physical fonts are examined by the font mapper. Aspect ratio filtering, which is available in both Windows version 3.0 and version 3.1 allows an application to specify that only fonts designed for the device's aspect ratio should be considered by the font mapper. An application enables and disables this filter by using the **SetMapperFlags**

function. Because nonscaling raster fonts are designed with a certain aspect ratio in mind, it is sometimes desirable to ensure that only fonts actually designed for the device are used. When this filter is enabled, the font mapper does not consider any physical fonts whose design aspect ratio does not match that of the device. Aspect ratio filtering does not affect TrueType fonts because they can scale to match any aspect ratio. This filter affects all font selections to the DC until the filter is turned off. Aspect ratio filtering is a holdover from earlier times and is not a recommended approach in today's font world.

The second type of filtering is new for Windows version 3.1 and allows an application to choose what type of physical font should be chosen in the case of a facename conflict. These filters only affect the shortcut mapping described above; if no shortcut match is found, the font mapper continues with its standard matching scheme and ignores these filters. The application defines the filter in the **lfOutputPrecision** attribute of the logical font. Unlike the aspect ratio filter, this filter is specific to a given logical font. The three possible filter settings are OUT_TT_PRECIS for TrueType fonts, OUT_DEVICE_PRECIS for device fonts, and OUT_RASTER_PRECIS for raster fonts. For example, if the shortcut mechanism finds two fonts, one TrueType and one raster, that provide an exact match for the logical font and the logical font is defined with OUT_TT_PRECIS, the TrueType font is chosen.

Another shortcut filter, OUT_TT_ONLY_PRECIS, specifies that only TrueType fonts should be considered for the exact matching. None of the other types of physical fonts is considered. This filter is useful for an application that wants to guarantee that the realized font is a TrueType font.

Default Values

Many of the attributes that define a font have a defined default setting. For most attributes, the font mapper ignores default attributes in its penalty scheme so that no penalty is given for that attribute for any candidate font. This is a simple way to indicate that an attribute is not important in the font mapping.

There are times when this gets dangerous. Using DEFAULT_CHARSET for specifying the font's **lfCharSet** can lead to the font mapper choosing a non-ANSI font that may not contain needed characters and may not even be composed of alphanumeric glyphs. Internally, the DEFAULT_CHARSET is an actual charset definition, so all candidate fonts get an equally large penalty for not matching the charset. It is always a good idea to use a non-default charset when defining a logical font.

The font mapper treats a logical font with an **lfHeight** of 0 as a font that is 12 points high. The actual pixel height depends on the resolution of the device being used. With the advent of TrueType technology in Windows version 3.1, this default height becomes even less meaningful (more on that below). The moral: always specify a height.

By specifying no face name (**lfFaceName**) for the logical font, an application indicates to the font mapper that the exact face name of the physical font is not critical. An application that does not specify a face name is just asking for trouble. Because the face name is the most unique identifier of a font, ignoring it leaves the application open to more ambiguity in realizing a physical font. The more fonts are available on the system, the less control the application has over the mapping process. An application can overcome some of this ambiguity by defining the generalized look of the font using the **lfPitchAndFamily** field, but as more and more fonts become available in the Windows system, a font's unique face name becomes critical in its identification. The moral: always specify a face name.

The font mapper interprets an **lfWeight** of FW_DONTCARE as though it was really FW_NORMAL, and any corresponding penalty is tabulated.

It is generally considered a good thing to specify a default **lfWidth** by setting this attribute to 0. While for most fonts the character widths cannot be changed and are not actually affected, scalable fonts (TrueType and many device fonts) are altered in shape by a non-default width. This is usually not desired. Because height is more important in the matching process, a non-default width usually does not affect the matching, but it could. The font mapper compares this value to the average width value of the candidate font (accessible via the **tmAvgWidth** field of the TEXTMETRIC structure), a value which for nonscalable, variable pitch fonts is an approximation at best. For scalable fonts, the **lfWidth** field is used to define an *x*-scaling factor for the font (*x*-scale = **lfWidth**/**tmAvgWidth**) where an **lfWidth** of 0 means a scaling factor of 1.0, the default.

Font Rotation

An application specifies the desired rotation of a logical font using the **lfEscapement** and **lfOrientation** attributes. The main font mapper does not use these attributes in its font selection processno penalties are assessed for candidate fonts that are not rotated or rotatablebut the shortcut method does not select a raster font if either of the attributes is nonzero. The key issue here is that not all fonts, raster fonts especially, can be rotated effectively. Because font rotation is not a factor in the mapping, it is possible that the chosen physical font is not able to rotate as desired by the application. Fonts that do rotate are TrueType fonts, the vector fonts, and some device fonts. It is wise for an application that desires rotated fonts to specifically ask for a font that can actually be rotated.

Other Warnings

Most applications do not want a vector font selected as the physical font because vector fonts are ugly. Before TrueType, the vector fonts were the only fonts that could be arbitrarily scaled and rotated, and it could be argued that they maintained more dignity than a GDI-scaled raster fonts at very large size. Mostly, though, they are to be avoided. The simplest way to do this is to explicitly use ANSI_CHARSER or SYMBOL_CHARSET when defining a logical font. The penalty for the charset is large enough to keep the vector fonts from font matching contention. Vector fonts can also be culled out of an enumeration by ignoring any enumerated font that is not a raster, TrueType, or device font. With the **ChooseFont** common dialog, vector fonts can be removed from the user's view through use of the CF_NOVECTORFONTS flag.

The height and width of a logical font are defined in logical units. When the font is selected into a DC, these values are converted to device units before the font mapping takes place. As a result, the realized font may be larger or smaller, depending on the DC's current mapping mode. Also, if an application changes the mapping mode of a DC, the currently selected font is re-realized based on the new coodinate system. This is entirely consistent with GDI's handling of logical and physical coordinates and is something to keep in mind when specifying the height of a font. The one exception to this process is new to Windows version 3.1: if the logical font is one of the stock objects (available through **GetStockObject** function-- copies don't count), the height and width are always treated as MM_TEXT values, regardless of the DC's current mapping mode.

GDI simulations

GDI provides simulations for some font attributes. It emboldens a normal weight font by overstriking, italicizes a nonitalic font by shearing the output (TrueType fonts are sheared during rasterization), underlines or strikes-out fonts, and scales raster fonts independently in *x* and *y* by integral amounts. In cases where GDI's simulations help a font to become a better match, the font mapper reduces but does not eliminate the penalty for a mismatch of the simluated attribute.

This makes the font a more palatable choice. Unfortunately, those simulations that alter the shape of the font (emboldening, italicizing, and scaling) are not always typographically pleasing. With the exception of scaling, all of these simulations can be performed on any of the available fonts.

When GDI scales a raster font, it does so by simply stretching the font's original bitmap definition horizontally and vertically, as appropriate. Because the scaling is always integral, none of the really nasty side-effects of bitmap stretching occur, but a stretched font still looks very clunky, especially at larger scaler factors.

GDI can trivially simulate underlines and strikethroughs using horizontal lines, so these attributes are not a real concern. While TrueType fonts have special metrics for the placement of underlines and strikethroughs, GDI approximates these values for other fonts with the underline on the baseline and the strikethrough 1/3 of the ascent above the baseline.

Stock Object Special Case

Under Windows version 3.0, a font's logical height and width are always translated to physical units before font mapping, so a stock font could potentially map to an unexpected font if the DC's coordinate system results in a different-sized physical font. This can be a real "gotcha" for an application that assumes it can use the SYSTEM_FONT stock object (for example) without regard to the scaling being done. The physical font scales with the coordinate system.

Under Windows version 3.1, if an application selects a stock font into a DC, the requested font's logical height and width are not translated into logical units. The effect is that a stock font remains the same size regardless of the DC's mapping mode. A copy of a logical object does not qualify for this special feature, only the actual stock object does. Because objects recorded in metafiles are simply copies of objects, a stock font recorded in a metafile is an ordinary font when the metafile is played back.

Actual Weights

The following table specifies the penalty weights used by the font mapper in Windows versions 3.0 and 3.1. Penalties that are new for Windows version 3.1 are identified as such. Note very carefully that this information is *version specific* and is subject to change without warning in future versions of the Windows product. In the table, *requested* refers to the logical font, and *candidate* refers to the physical font to which it is being compared. The candidate with the smallest penalty is the one that is elected.

penalty description	penalty weight	comments
CharSet	65000	charset does not match
Output Precision	19000	request OUT_STROKE_PRECIS, but device can't do it or candidate is not vector font OR don't request

		OUT_STROKE_PRECIS, candidate is a vector font, and device doesn't support it
FixedPitch	15000	request fixed pitch, candidate variable pitch
FaceName	10000	request face name, candidate does not match
FaceNameSubst	500	new for 3.1
		candidate is a substitute face name
Family	9000	request a family, candidate's family is different
FamilyUnknown	8000	request a family, but candidate has no family
FamilyUnlikely	50	new for 3.1
		request roman/modern/swiss, candidate decorative/script or vice versa
HeightBigger	600	candidate non-vector font and bigger than requested
PitchVariable	350	request variable pitch, candidate not variable pitch
HeightSmaller	150	raster candidate and smaller than requested
		penalty * height difference
HeightBigger	150	raster candidate and bigger than requested
		penalty * height difference
Width	50	request a width, candidate doesn't match
		penalty * width difference
SizeSynth	50	raster candidate needs scaling by GDI
UnevenSizeSynth	4	raster font scaled unequally in width and height

		penalty * (100 * bigger multiplier / smaller multiplier)
IntSizeSynth	20	raster font needs scaling
		penalty * (height multiplier + width multiplier)
Aspect	30	candidate aspect ratio different from device
		penalty * ((100 * devY/devX) - (100 * candidateY / candidateX))
Italic	4	request and candidate do not agree on italic status, and the desired result cannot be simulated
ItalicSim	1	new for win 3.1
		request italic, candidate not italic but can be simulated to be italic
Weight	3	candidate's weight does not match
		penalty * (weight difference/10)
Underline	3	request no underline, candidate is underlined
StrikeOut	3	request no strike-out, candidate is struck
DefaultPitchFixed	1	request DEFAULT_PITCH and candidate is fixed pitch
SmallPenalty	1	new for win 3.1
		request rotated font, candidate needs bold or italic simulation and is a raster or vector font
VectorHeightSmaller	2	candidate vector font is smaller
		penalty * height difference
VectorHeightBigger	1	candidate vector font is

		bigger penalty * height difference
DeviceFavor	2	extra penalty for all non-device fonts new for win 3.1, request OUT_TT_PRECIS and candidate is not TrueType. Penalty is twice this value.

End.

Windows Mapping Modes and Tools

Suppose the following sequence of events:
1. BeginPaint returns a default HDC (map mode = MM_TEXT)
2. Select Pens, Brushes, and Fonts into the device context
3. Change the mapping mode using SetMapMode
4. Change the window & viewport extents

Question: Have the pens, brushes, fonts, etc already in the device context been updated by Windows automatically, since they were specified in "logical units" when they are created and I have changed the "logical unit" mapping? Are they still the same size,
(in device units) that they were before? Do I have to create new pens, brushes, and fonts in order to have the size updated? Do all three have to be recreated, or just some?

Answer: Pen widths are in logical units. When you change mapping modes, there is the possibility that a logical unit could cover more that one pixel. Windows does not change the width of the pen when the mapping mode is changed. The pen width is still the same logical units.

The change to a different mapping mode can cause the pen to expand or contract depending on the mapping mode, window extents and viewport extents. Fonts will act the same way as pens. A font will be stretched or compressed with respect to the mapping mode, window extents and viewport extents. The size of a brush is 8 pixels by 8 pixels. This will not change among mapping modes.

All in all, some items might have to be recreated and some might not. It mainly depends on how the logical units map on to the new map mode. Windows does not adjust the logical sizes of the pen ,font, or brush.

Question: The Windows SDK documents the calcultations used by Windows to transform Logical coordinates to device coordinates. Is this calculation performed all the time such that no matter what the mapping mode or extents, an output sequence of events to a device always has the same overhead/time, or is MM_TEXT more efficient because it maps directly to the internal system used by Windows and thus the transformation calculations are not done?

Answer: With respect to your question of mapping mode overhead:
For ALL mapping modes, Windows will translate logical coordinates to device coordinates using the following formulas:
xViewport = (xWindow - xWinOrg)*(xViewExt/xWinExt) + xViewOrg
yViewPort = (yWindow - yWinOrg)*(yViewExt/yWinExt) + yViewOrg
where (xWindow, yWindow) is a logical point to be translated and (xViewPort, yViewPort) is the translated point in device coordinates. If the device coordinates are client-area coordinates or window coordinates then Windows must translate these device coordinates to screen coordinates before drawing an object.

For the MM_TEXT mapping mode, the default origins and extents are:

Window origin	(0, 0)
Viewport Orgin	(0, 0)
Window extent	(1, 1)
Viewport extent	(1, 1)

The ratio of the viewport extent to the window extent is 1. No scaling will be performed between logical and device coordinates.

So the overhead is generally going to be the same since all mapping modes will use this translation. A good source on mapping modes is "Programming Windows 3.00" by Charles Petzold.

End.

PostScript Printer Description Files
&
Driver Limitations

The PostScript Printer Description file format, defined by Adobe and extended by the Microsoft, provides the basis for external printer descriptions. This ASCII file contains one or more statements, each consisting of a keyword and associated values. Each keyword defines a feature of the PostScript printer. The associated values specify whether the feature is available, or how the feature affects the operation of the printer. For example, the *FileSystem keyword specifies whether a printer supports fonts loaded on a hard drive. PPD files must be standard ASCII files with carriage return an
d line feed pairs terminating each line.

To ensure the best performance for a printer, the PPD file should be as complete as possible. Windows 3.1 has defined new PPD keywords that should be added to existing PPD files to create new WPD files. If these keywords are not added, the driver can still use the old WPD files, but it will assign default values to any keywords not explicitly
defined.

The following provides descriptions of the PPD keywords.

Keyword	**Description**
ColorDevice	Indicates whether the printer supports color.
DefaultFont	The name of the default font (that is, the font used if none is selected). This setting must appear before any *Font settings.
DefaultInputSlot	The default input slot.
DefaultResolution	The default resolution of the printer.
Duplex PostScript commands to set duplex mode. This feature is new for Windows 3.1.	
FileSystem	Reserved; do not use.
Font	The fonts resident in the printer.
FreeVM Amount of free memory in standard printer configuration. This feature is new for Windows 3.1.	
ImageableArea	The actual area that can be marked on for every paper size.
InputSlot	The PostScript code that is necessary to specify each input slot.
ManualFeed False	The PostScript code that is necessary to turn off manual feed. If present, it is assumed that manual feed is supported.
ManualFeed True	The PostScript code that is necessary to specify the manual feed operation. If present, it is assumed that manual feed is

	supported (and therefore PageRegion settings must be included).
NickName	The name that appears in the printer dialog box. It should be a unique description of the printer. This is also the name used for automatic printer recognition.
PageRegion	The PostScript code that is necessary to specify different page sizes when using manual feed.
PageSize	The PostScript code used to specify the different page sizes that are supported (when not in manual-feed mode).
PaperDimension	The size of all the used paper types. A Paper Dimension setting must be included for every size of paper supported. The sizes of the standard page types should be used. Only standard page types are recognized. *SetResolution Used to determine the hardware resolution(s) supported. This feature is new for Windows 3.1.
Transfer Normalized	The normalized transfer function used to generate linear gray levels. This must be present. If none is required, include the nul function { }.

The following is a list of PPD extensions:

Extension	Description
AcceptsTrueType:	True or false. If true, TrueType fonts will be downloaded natively using readhexsfnt operator. This is selected through the download option in the Advanced Options dialog box. If false, TrueType will not be displayed as a selection. This feature is new for Windows 3.1.
EndOfFile	Indicates whether ^D is required to indicate the end of the file. This is true by default, and only needs to be included if this is false (that is, *EndOfFile False).
SetPage	True or false. If true, the PostScript driver allows a custom paper size to be defined. This is implemented through the use of the setpage operator. If your printer supports the setpage operator in the same manner as Linotronic® printers, you can use this option.
TrueImageDevice:	True or false. If true, the PostScript driver takes advantage of some optimizations available in TrueImage™. Currently there is very little difference in the output. This feature is new for Windows 3.1.

The following are the paper keywords used to show the paper sizes supported:

Keyword	**Description**
10x14	10 x 14 inches physical size, oriented in portrait mode.
11x17	11 x 17 inches physical size, oriented in portrait mode. Can be used interchangeably with the keyword Tabloid.

A3	297 x 420 millimeters physical size, oriented in portrait mode. Refers to the International Standards Organization (ISO)/(JIS) A3 paper size.
A4	210 x 297 millimeters physical size, oriented in portrait mode.
A4	Extra 9.27 x 12.69 inches physical size.
A4	Small 210 x 297 millimeters physical size, but with a reduced-size imageable area of 7.47 x 10.85 inches that is centered on an A4 page. Supports the Adobe PostScript paper definitions.
A5	148 x 210 millimeters physical size, oriented in portrait mode.
B4	250 x 354 millimeters physical size, oriented in portrait mode. Refers to the Japanese Industrial Standard (JIS) B4 paper size.
B5	182 x 257 millimeters physical size, oriented in portrait mode.
Folio	8.5 x 13 inches physical size, but with a reduced-size imageable region, oriented in portrait mode and centered on the folio sheet. Supports the Adobe PostScript paper definitions.
Ledger	17 x 11 inches physical size, oriented in landscape mode (that is, the y-axis is on the shorter edge of the paper).
Legal	8.5 x 14 inches physical size, oriented in portrait mode.
LegalExtra	9.5 x 15 inches physical size.
Letter	8.5 x 11 inches physical size. Refers to the standard paper type.
LetterExtra	9.5 x 12 inches physical size.
LetterSmall	8.5 x 11 inches physical size, but with a reduced-size imageable region that is centered on the page. Supports the Adobe PostScript paper definitions.
Note	8.5 x 11 inches physical size, but with a reduced-size imageable region. This is used to reduce the size of the page buffer to give print jobs more memory.
Quarto	215 x 275 millimeters physical size, but with a reduced-size imageable region, oriented in portrait mode and centered on the quarto sheet.
Statement	5.5 x 8.5 inches physical size, oriented in portrait mode.
Tabloid	11 x 17 inches physical size, oriented in portrait or tabloid mode (that is, the y-axis is on the longer edge of the paper).
TabloidExtra	11.69 x 18 inches physical size.

For paper extensions, five standard envelope sizes are recognized. The two groups of numbers following the word Envelope indicate the size of the envelope in points (where each point equals 1/72 of an inch).

Extension	Description
Envelope.279.639	#9 Envelope (3.875 x 8.875 inches)
Envelope.297.684	#10 Envelope (4.125 x 9.5 inches)
Envelope.324.747	#11 Envelope (4.5 x 10.375 inches)
Envelope.342.792	#12 Envelope (4.75 x 11 inches)
Envelope.360.828	#14 Envelope (5 x 11.5 inches)

The following are the paper tray and bin keywords used to show and specify the input slots supported.

Keyword Description	
LargeCapacity	This one can hold more than a standard amount of paper.
Lower	If there is more than one tray, this one is on the bottom. Middle This one is in the middle.
OnlyOne	There is only one tray.
Upper	If there is more than one tray, this one is on top. The following list describes the paper tray extensions.
Extension	Description
AutoSelect	Printer can select automatically which feeder to use. This is followed by the code (or a nul command if no code is required) that is used to specify the autofeed mechanism.
Envelope	There is an envelope feeder.
EnvelopeManual	There is a manual envelope feeder.
None	There are no input feeders. This is treated as being the same as OnlyOne. The following keywords are required to support duplex printing.

Keyword Definition	
Duplex DuplexNoTumble:	statusdict begin false settumble true setduplexmode end
Duplex DuplexTumble:	statusdict begin true settumble true setduplexmode end
Duplex None:	statusdict begin false setduplexmode end
DefaultDuplex	None

If the PPD with correct *Duplex settings is built with the Windows 3.1 MKPRN utility, the driver expands the Options dialog box to give user control of duplex settings.

PostScript Fonts Have Two Font Names

A PostScript font has two font names:

1. The Windows name for the font, such as "AvantGarde," which appears in the font list in the Fonts dialog box in Control Panel.
2. The PostScript name for the font, such as "itc avant garde gothic," which can vary by printer

manufacturer and which the printer driver sends to the printer to select the font.

More Information:
If the font is a downloadable soft font defined in the printer font metrics (PFM) file format, the Windows name and the PostScript name for a font can be determined. The PFM file header has a field (dfFace) that points to the Windows font name and a field (dfDriverInfo) that points to the driver-specific PostScript font name.

If the font is an internal printer font defined for a specific printer, it is necessary to enumerate fonts in a given printer driver to find the Windows font name. There is no device-independent way to retrieve the PostScript font name for an internally defined printer font.

For more information concerning the PostScript PFM file format, refer to Chapter 4 of the "Microsoft Windows Device Development Kit Printers.

Supporting PostScript Features in Windows

The information in this article applies to:
Microsoft Windows Software Development Kit for Windows versions 3.1 and 3.0

Summary:

There are some issues involved when designing an application to provide support for PostScript printers. The application must determine if the PostScript driver is available by using an accurate detection system. If an application generates PostScript directly, the PASSTHROUGH escape can be used to send the file. This must be done with care because the application is communicating directly with the printer.

More Information:
The first issue is how to determine if a PostScript driver is an installed printer driver under Windows. An application cannot assume the PostScript driver is named PSCRIPT.DRV because this forces PostScript driver vendors to use the same filename. The correct method is to run code similar to the pseudocode below:

```
    bFound = FALSE;
    for (each device in [Devices] section of win.ini) {
/* extract the necessary fields from the ini line */ szDriverName = driver name extracted from ini line
            szModelName = left side of ini line (the key)
            szPort = port name extracted from ini line.
hIC = CreateIC(szDriverName, szModelName, szPort, NULL);
            if (hIC) {
                        /* see if driver supports GETTECHNOLOGY escape */
                        wEscape = GETTECHNOLOGY;
if (Escape(hIC, QUERYESCSUPPORT, sizeof(WORD), &wEscape, NULL)) { Escape(hIC, GETTECHNOLOGY, 0, NULL, &szTechnology);
                                    /* Check that the string starts with PostScript
                                            * by doing a case-insensitive search. Allow
                                            * for the possibility that the string could be
* longer, like "PostScript level 2" or some other * extension.
                                            */
                                    if (beginning of string is "PostScript")
                                                bFound = TRUE;
```

```
                        }
                        DeleteDC(hIC);
                }
                /* if the driver has been found break out */
                if (bFound)
                        break;
        }
        if (bFound) {
PostScript driver is szDriverName, model is szModelName, port is szPort.
        }
```

The second issue is how to print application-generated PostScript code. The mechanism from a Windows application is through the PASSTHROUGH escape. The PASSTHROUGH escape is documented in the "Microsoft Windows Software Development Kit Reference Volume 2," Chapter 12. In addition to the documentation, one requirement on the buffer passed is easy to miss; the first word must contain the length of the buffer. The contents of the data sent by PASSTHROUGH can alter the state of the printer.
To be safe, obey the following rules:

1. Surround PASSTHROUGH data by save/restore PostScript operators. 2. Do not embed GDI calls between PASSTHROUGH escapes. For example:

   ```
   PASSTHROUGH(save)
           Rectangle
           OtherGDIRoutines
           PASSTHROUGH(restore)
   ```

   Some driver code and software fonts are downloaded to the printer under certain conditions. The above operations could cause the driver and printer to loose synchronization, and potentially cause the job to fail. In general, no assumptions should be made concerning the code generated by a given GDI call.

Driver Limitations

The following are limitations placed on the driver by PostScript:
PostScript does not support most raster operations (Rops). However, it does support BLACKNESS, WHITENESS, and SRCCOPY.

PostScript has a 750-point polygon limit. This number is reduced by two when filling with a hatch or pattern. This is because a clipping path must be built as well as the path to fill and stroke. In cases where this limit is reached, the driver will request that GDI simulate the polygon. This is very slow. Applications should avoid generating large polygons.

End.

PFM Files for PostScript Printers

Because PostScript fonts are scalable, the PFM files for the PostScript fonts do not contain width tables.

The files have the following form:

```
PFMHEADER       Header;          /* font header */
PFMEXTENSION    Extensions;      /* extensions */
char            DeviceName[];    /* printer device name */
char            FaceName[];      /* font face name */
EXTTEXTMETRIC   ExtTextMetrics;  /* extended text metrics */
WORD            ExtentTable[];   /* unscaled character widths */
DRIVERINFO      DriverInfo;      /* driver-specific information */
PAIRKERN        KerningPairs[];  /* pair-kerning table (optional)*/
KERNTRACK       KerningTracks[]; /* track-kerning table (optional)*/
```

The PostScript driver for Windows assumes all PostScript fonts are scalable fonts, so it ignores the dfPoints and dfPixHeight members in the PFM header. The members dfAvgWidth and dfMaxWidth are in units of 1000 units-per-em.

Although the PostScript naming convention includes the attributes of the font (that is, bold and italic) in the font name, the attributes should be stripped from the font name and represented in the dfWeight and dfItalic members in the PFM header.

The extent table is an array of 16-bit values containing the unscaled widthsof the characters and assuming 1000 units-per-em. The range of the tableshould be from dfFirstChar to dfLastChar. The size of the table should bedfLastChar dfFirstChar + 1. Pair-kern values should be in the same 1000 units-per-em measurement as the extents. As of this writing, we do not know of any application that uses the track-kern table.
In the EXTTEXTMETRIC structure, the PostScript driver assumes the following values for each font:
dfVertRes = 300
etmMasterHeight = 300
etmMasterUnits = 1000

In other words, the driver assumes all fonts use Adobe's standard 1000units-per-em method for describing a font. You must build the extent tablebased upon 1000 units-per-em to be consistent with this restriction in thedriver.

The driver also assumes that the font may be scaled to any point size the application requests. We recommend that the true scaling range of the font be indicated in the etmMinScale and etmMaxScale members (in device units, at 300 dpi). The driver currently ignores these members.

Because the Windows PostScript driver assumes all PostScript fonts are scalable fonts, it ignores the etmPointSize member. The etmSize member is not the point size, but rather the size (that is, the number of bytes) of the EXTTEXTMETRIC structure.
As of this writing, we do not know of any application that uses the members in the EXTTEXTMETRIC structure except for etmKernPairs. If your font contains kern pairs, you must fill in the EXTTEXTMETRIC structure to indicate the number of kern pairs. Do not leave the other members blank; fill them in with reasonable values in the event an application does use them.

The driver-specific data structure pointed to by dfDriverInfo is a null-terminated string containing the PostScript name for the font. There are two names for the font:

1) The Windows name for the font which appears in the font list in the application's font dialog box.

2) The PostScript name for the font, which can vary according to printer manufacturer and which the driver sends to the printer to select the font.

Both strings must be null-terminated. The Windows name for the font is pointed to by dfFace and the PostScript name for the font is pointed to by dfDriverInfo.

To create a PFM file for the internally-supported printers, Microsoft converts Font Metric (AFM) files to the PFM file format. These files, provided with the DDK, are ASCII files that define the metrics for each font. There are 63 AFM files supporting 63 unique fonts. Printers with resident fonts that are not supported in the current version of th
e driver will need to define those files as soft fonts.

If you need to define a font that is not available currently in the PostScript printer driver, you will need to modify the driver sources to add additional AFM files. Please refer to the PostScript driver's MAKE file that is included on the DDK disks for more information on how the driver builds AFM and PFM files.

End.

Sending EPS (Encapsulated PostScript) to a Printer

Summary:

An application can perform the following steps to send EPS (Encapsulated PostScript) to a PostScript printer as part of a document:

1. Use the DEVICEDATA printer escape to send the information.

2. Frame the EPS with the gsave and grestore commands. If the EPS contains any complicated output, use the save and restore commands instead.

The PostScript printer driver preserves the order of its input. Also, because the PostScript driver does not perform any banding, the print spooler should not cause any problems.

Note: The application must download code to the printer to correctly position the EPS image on the page [that is, to change the current transform matrix (CTM)]. In addition, to avoid unwanted page ejections, the application must edit the EPS to remove commands such as ShowPage.

Supporting PostScript Features in Windows

Summary:

There are some issues involved when designing an application to provide support for PostScript printers. The application must determine if the PostScript driver is available by using an accurate detection system. If an application generates PostScript directly, the PASSTHROUGH escape can be used to send the file. This must be done with care because the application is communicating directly with the printer.

More Information:

The first issue is how to determine if a PostScript driver is an installed printer driver under Windows. An application cannot assume the PostScript driver is named PSCRIPT.DRV because this forces PostScript driver vendors to use the same filename.

The correct method is to run code similar to the pseudocode below:

```
    bFound = FALSE;
    for (each device in [Devices] section of win.ini) {
/* extract the necessary fields from the ini line */ szDriverName = driver name extracted from ini line szModelName = left side of ini line (the key)
            szPort = port name extracted from ini line.
hIC = CreateIC(szDriverName, szModelName, szPort, NULL);
            if (hIC) {
```

```
                        /* see if driver supports GETTECHNOLOGY escape */
                        wEscape = GETTECHNOLOGY;
if (Escape(hIC, QUERYESCSUPPORT, sizeof(WORD), &wEscape, NULL)) { Escape(hIC,
GETTECHNOLOGY, 0, NULL, &szTechnology);
                                /* Check that the string starts with PostScript
                                        * by doing a case-insensitive search. Allow
                                        * for the possibility that the string could be
* longer, like "PostScript level 2" or some other * extension.
                                        */
                                if (beginning of string is "PostScript")
                                                bFound = TRUE;
                        }
                        DeleteDC(hIC);
                }
                /* if the driver has been found break out */
                if (bFound)
                        break;
    }
    if (bFound) {
PostScript driver is szDriverName, model is szModelName, port is szPort.
    }
```

The second issue is how to print application-generated PostScript code. The mechanism from a Windows application is through the PASSTHROUGH escape. The PASSTHROUGH escape is documented in the "Microsoft Windows Software Development Kit Reference Volume 2," Chapter 12. In addition to the documentation, one requirement on the buffer passed is easy to miss; the first word must contain the length of the buffer. The contents of the data sent by PASSTHROUGH can alter the state of the printer.

To be safe, obey the following rules:

1. Surround PASSTHROUGH data by save/restore PostScript operators.

2. Do not embed GDI calls between PASSTHROUGH escapes. For example:

   ```
   PASSTHROUGH(save)
   Rectangle
   OtherGDIRoutines
   PASSTHROUGH(restore)
   ```

   Some driver code and software fonts are downloaded to the printer under certain conditions. The above operations could cause the driver and printer to loose synchronization, and potentially cause the job to fail. In general, no assumptions should be made concerning the code generated by a given GDI call.

3. Do not send a command to cause a page ejection.

Additional reference words: 3.00 3.10 3.x
KBCategory:
KBSubcategory: GdiPrnMisc

TrueType Support

The Windows 3.1 PostScript driver supports TrueType fonts allowing users to create and print document containing TrueType fonts on PostScript printers. The driver also provides a variety of advanced options that let the user determine how TrueType fonts are to be used with the printer. Depending on the capabilities of the printer, the user may want to convert TrueType fonts to an

Abode format, or substitute the TrueType font with a printer-resident font.

The PostScript driver may need to convert the fonts to PostScript device-font format before downloading to the printer. This is true if the printer does not support the TrueType font format itself. If the driver must convert TrueType fonts, it can convert them to either Adobe Type 3 bitmap fonts or Type 1 outline fonts. The user sets the type in the TrueType option in the Advanced Options dialog box. The TrueType to Type 1 conversion is not exact and may produce fonts of slightly inferior quality to the original TrueType font. However, using this conversion will reduce print time and require much less printer memory for documents containing lots of TrueType fonts at large point sizes.
The user can direct the PostScript driver to substitute PostScript printer-resident fonts for TrueType fonts by using the Substitution option in the Advanced Options dialog box. This option displays another dialog box that lists the TrueType fonts and lets the user choose the substitution font. Using substitution fonts reduces print time and requires
less printer memory.

If the printer supports the TrueType font format, the user can also direct the driver to download TrueType fonts to the printer by using the Substitution option in the Advanced Options dialog box. In this case, the user sets the Download as a SoftFont option for a TrueType font instead of choosing a printer-resident font. The driver converts the
Truetype font to a soft font and downloads it to the printer. This is the default for any TrueType font that hasn't been assigned a printer font in the substitution table.

End.

Information Structure and Layout

Before designing a topic layout, you should analyze the information and decide what is the best method for presenting the information. To lay out screen elements effectively, you must understand the structure of the information. Screen design involves using visual properties such as space, position, and size to communicate structural relations.
There are three principal relations between information elements that you can convey by using visual organization:

- For example, a caption associated with a specific picture.
- For example, a series of numbered steps.
- For example, a title in comparison with the rest of the topic.

To help you clarify these structural relationships when considering the layout of a Help topic, try the following exercise. Become a user, stand back from the screen, and ask questions such as:

- Which items are of the same kind?
- Which items are functionally related?
- In what order should the various elements be presented? Does the order agree with the information presented? Which are the most important items?
- Is there a hierarchy of importance?

Make a list of the kinds of information on the screen, going from the visually most dominant to the least dominant. Now consider whether this order makes sense given the message being put across. If necessary, change the order or emphasis of elements in the topic. Then redesign the topic to convey this new structure.

Visual Grouping
One of the most important aspects of screen design is visual grouping. Visual grouping refers to factors that cause some parts of an image to be seen as related. In visual terms, users are more likely to see parts that are similar in some way as related to each other than parts that are dissimilar.

An example of how visual grouping is applied to topic design is in the use of white space. Generally you should include more white space around a group of related items rather than between each item. The white space surrounding the items will help clarify the group structure of the information.

Visual similarity includes these factors:
In contrast, visual separation is based on a dissimilarity of these factors.

Visual Order
Determining the order of information in a topic depends on the message you want to convey. Usually, this order is a straightforward sequence from top to bottom, but it may involve more complex movements. In fact, you can influence how users read the information by the way you

order items in your topic layout. Guiding a users eye to the most important information is one of the key objectives you have in screen design.
When first looking at a topic, users generally scan the whole layout to get an overall impression. After that, users tend to focus on elements that have emphasis. The following list summarizes these tendencies. Keep in mind that these are tendencies and not absolutes but users tend to scan to the left and upward.

Issues
Instructions, cues, and important information should appear at crucial points in the order.

When deciding where to place information in the order, keep in mind that screen areas have this order of importance: left is more significant than right, top is more significant than bottom. Place items in the topic so that the information tells the story visually. But take care not to let primary elements appear in too many places, or the topic may begin to look chaotic and confuse users.

Visual Hierarchy
Very little information exists in which all the elements have exactly the same importance, and there is no visual hierarchy. More commonly, information has a definite hierarchy of importance, in which some elements receive more emphasis than others. This is especially true in instructional materials like Help.

The effectiveness of the various means of emphasis derives from their perceptual qualities. Visual emphasis relies to a large degree on the effects of contrast. If one item is larger than the rest, it stands out. But if all but one of the elements are large, then the single small element is the most prominent.

There are many ways to create emphasis; the important thing is to set up a consistent visual hierarchy in your Help topics by determining which elements to emphasize and to what degree. These principles can help you do that:

This will direct the attention of the user toward what matters most. For example, using a topic title in a larger type size and placing it at the top emphasizes it over the rest of the information in the topic.

Emphasis is influenced by what is usual. For example, because most Western people read from left to right and from top to bottom, information placed at the top left receives more emphasis than information at the bottom or right part of the Help window. The number of elements and each elements position in the topic affect how it is perceived in the visual hierarchy and what you must do to give it more emphasis. For example, a word buried in the middle of the topic requires more emphasis to bring it out than a heading at the beginning of a paragraph.

Too much emphasisor too many words emphasized in too many wayscan create the opposite effect from the one intended. Instead of getting the message, the user may simply ignore it. Also, if you try to emphasize too much, you may make everything roughly equivalent and lose all emphasis.

Creating Emphasis
There are a number of techniques for making visual distinctions between elements and emphasizing those of particular importance. Certain forms of emphasis are especially powerfulcolor, for exampleand should, therefore, be used with particular care. Try to establish a hierarchy of importance using these techniques, and then employ that hierarchy consistently throughout the Help file.

Consistency
Consistency is fundamental to effective screen design. It contributes to usability in a number of waysit facilitates learning, lessens the number of errors users make, and helps users develop an

accurate model of the system. Of course, it is rarely possible to be completely consistent within a Help file, so you will often have to determine your
priorities and make tradeoffs accordingly. For example, you should not be rigidly consistent if the decision results in an awkward design, when an inconsistent design would achieve the goal more efficiently.

The following guidelines can help you with consistency:

- because they clearly have different functions in Help, jump and pop-up hot spots are differentiated by the type of underlining that they display in the hot-spot text.
  The learning process is facilitated if the user can easily grasp the relationships between elements within the system. This can only occur if you use consistent terminology, provide consistent behavior for controls, and so on. For example, if users are to choose a blue circle that indicates that a tip is available, consistent use would mean that the blue circle always provides a tip and that no other cue would provide the same tip.

- if a certain information category, such as step-by-step procedures, requires you to vary from some principle established elsewhere in the Help file, at least be sure to use the feature consistently within all the topics of that category. If there are large inconsistencies between related systems, the users ability to generalize from one to another is reduced. It may also cause them to make more errors because they may assume that actions taken in one application have the same effect in other, related applications. Therefore, if your Help file deviates from Windows or Windows-based applications, users may find it particularly difficult to use and learn. For example, users expect to be able to click grey, 3D buttons, so dont include buttons in your Help file unless they have a similar function.

- if screen shots of the interface are not normally hot, and you include one that is, tell users that this particular graphic includes hot spots.

Foreground and Background
In any image, some parts tend to be seen in the foreground and some in the background. In a graphical user interface, active and inactive windows create the illusion of foreground and background. For that reason, you can use windows to separate different kinds of information, especially where you want to create content that is somewhat independent from the main Help window.

When placing information in Help windows, use the following guidelines:

- Because pop-up windows are displayed temporarily (until the user takes any action), they create less separation between the information in them and in the main Help window. That makes them ideal for displaying subordinate and supplementary information.

- Secondary windows share many of the same characteristics as the main Help window. For one thing, they can be displayed or dismissed independently of the main window. That makes them ideal for holding information that is tangential to the content in the main Help window. However, because secondary windows can exist with or without the main window, they should only be used when the information displayed in them is truly independent from the content in the main Help window.
  If you use secondary windows, consider carefully the size and relationship they have to the main Help window.

- Users will likely have other windows open besides Help windows. Too many windows on the screen at one time can confuse and disorient users. There is considerable usability data indicating that user performance deteriorates as the screen image becomes more complex. Users have a harder time finding the information they want, and they make more errors. In

general, try to limit the amount of space that additional secondary windows occupy to less than 25 percent of the active screen area.

White Space
White space is the designers canvas. It is the area in which you display your message. It also creates the information boundaries that users rely on to recognize the message. White space includes both negative and positive valuesnegative space is the background, or the part we seem not to see; positive space is the foreground space that lies at the center of attention and dominates.

In online designs, space is two-dimensional, even though many elements simulate a three-dimensional space. Two dimensions have only length and width, but you can use certain techniques to create the impression of depth and volume.

Learning how to use white space is one of the most important aspects of screen design. The key to using white space depends on its ability to group and separate elements into recognizable symbols and images. An easy way to understand this principle is to consider spacing in type. Normal spacing in type allows our eye to recognize the symbols (letters) that form understandable words, as in this example:

GO OUT TONIGHT

If we remove the normal white space between letters, we cannot easily understand the symbols as a message:

GOOUTTONIGHT

In the same way, the proper use of white space in your topic layout will help users recognize and understand the message. To create a good lay out, make sure that all topic elementstopic title, text paragraphs, pictures, white spacecreate understandable images. A topic will look crowded and cluttered if too many elements occupy the space and prevent closure; it will look open if the space is balanced and generous.

Margins
Margins are the blank areas surrounding the information in a topic. In normal screen design, you have four margins to consider: top, bottom, left, and right. There isnt much you can do about the right and bottom margins in Help, so the left and top margins should be your primary concern:

*Left margin*with overlapping windows on the screen, text from one window can become confused with information in another window because there is only a thin dividing line (window border) separating the two. This can make the screen look unnecessarily cramped and cluttered. An ample left margin helps separate Help text and graphics from information in other windows.

Text
The one portion of multimedia that everyone feels familiar with is text. Text files form one of the largest segments of the multimedia developer's base of information. Many multimedia applications will be mainly text driven developed by converting a book into an online multimedia application.
The target format for Windows with Multimedia is either straight ASCII or Rich Text Format (RTFoutput by Word for Windows). Whenever you convert text files, however, you invariably lose some original formatting. You still must add the proper formatting, indexing, and other referencing tags to make the text useful. With the proliferation of word processors, desktop publishing programs, and electronic typesetting systems, almost everything currently being printed also exists in one electronically readable format or another. If the issues associated with text

preparation seem overwhelming, you have another alternative: pay someone else to do it. There are a number of data preparation houses that will gladly take your money and your text and return a finished product.

Note:
The Rich Text Format (RTF) is documented in the Word for Windows Technical Reference, available from Microsoft Press.

Text

Typography is perhaps the most widely discussed area of graphic design. And not surprisingly. We have been using books for centuries, and all the while we have been improving and changing them. Much of the knowledge that has accrued in book design can be applied to the design of electronic text, but certainly not all of it. These days, Help designers must be as familiar with electronic text as book designers have been with paper and print.

Text is something that all Help systems use. In fact, because of limitationscost of goods, time, resources, display technology, and so ontext continues to be the dominant information element in Help files. For that reason, pay careful attention to the design issues that text raises. For example, how much text should you write for one screen? Which font should you use? Should you use color? Can tables and lists present information effectively online?

An example of what text is good for:

- It is often impossible to judge visual parameters precisely, for example, the exact distance between two cities on a map. Consequently, where accuracy is required, numbers and words are more effective. (For example, maps translate their visual scale into numbers, one inch equals 100 miles, to be more precise.)

Pictures are not often used for depicting abstract ideas, such as the meaning of happiness, where traditionally words are much better suited. Pictures and diagrams can only roughly approximate logical relationships. Words and mathematical notation are usually superior. For example, Venn diagrams are a useful way to illustrate the principles of set theory, but they are not the primary means of expressing those ideas.

However, text can look dull and unappealing when used alone. A wall of text is especially disastrous online because users become easily overwhelmed by the information. Text-only Help files also may present problems for poor readers. Legibility and Readability

For text, the most important factors are legibility and readability. Legibility describes how easily the individual letters and words can be differentiated from each other. Legibility involves many factors, but the most important factor is contrast. The higher the contrast, the more legible the text (if other factors are also considered). Most books achieve high contrast, and thus legibility, by printing black text on white paper. But printed materials also use inverse printing and color to gain high contrastwhite text on a black background or dark red text on a yellow background, for example. The same factors apply on computer screens, and so most word processors display black text on a white background. Colored text is also legible on computer screens if there is sufficient contrast.

Readability is similar to legibility but refers more to the ease and comfort with which the text can be read. Obviously the text has to be legible to be readable. But other factors affect readability. For example, the text may be perfectly legible but not very readable if the line lengths are too long or the spacing is too crowded. Another factor of readability is formatting. Users have more difficulty reading text if it has a lot of formatting changes. Too many colors or frequent formatting changes make the information less readable.
When reading any material, people organize the material by categorizing the various text

elements according to their importance. They usually do not read all the words with equal attention and in the same order in which they are presented. You can make reading considerably easier if the text elements are visually distinct and consistent within the Help file. For instance, white space provides visual relief for users and helps them assimilate the information; it also signals divisions within the material.

The ease with which users can read text depends upon several factors:

- A 10-point font is sufficient for most screen fonts.

- Paragraph leading should facilitate reading by emphasizing each line of text. If lines are too close together, they can disrupt reading because of interference from the lines above and below the line being read. Adding sufficient space between lines improves the perceptual grouping of characters into rows.

- How much leading you use depends on the type size and line length. For example, the longer the line, the more leading it requires because the users eye has a greater distance to travel when finding the start of each line.

Most of the time you can control only the first two factors, so pay particular attention to them to create the most readable text possible.

Issues
Right justification and center-justified text can be effective for single paragraphs or other special-case text, but it is not advisable for body text because it produces a ragged left margin that disrupts reading.

Is the user expected to read and remember the text, to locate particular items within it, or to react to some items without needing to remember them? For example, standard paragraph formatting is consistent with the users previous experience and is simpler to produce. However, if the user has to pick out one item from a set of items, the
items should be displayed as a list.

Help text is a form of hypertext, or nonlinear writing. Users move forward and backward through a series of connected Help screens. Instead of flipping pages directly; they flip them indirectly, on the display. Hypertext is usually more effective if you present text in smaller amounts, rather than in long chapters and sections as in printed books.

This means a topic may consist of 100 words, 50 words, or whatever amount is required to create a complete idea. How much text you place in a single chunk depends on several factors: the content of the information, the size of the Help window, the font size, and the amount of white space. Always keep these factors in mind when considering
the amount of text that you include in a single topic. Do not follow print-based conventions such as filling the page with type; these conventions usually have little or no meaning when creating Help files.

Lists
Lists organize information using vertical and horizontal alignment, much like tables. The alignment provides a recognizable structure that tells users the information is related. It also tells them to read vertically, starting from the top of the list.

Online lists arent significantly different from printed lists. You should consider the effects of using indents and nested lists, however. These techniques may work better on the printed page than they do in the Help window. Creating too many margins using indents may make the information more difficult to read. Usually white space is a better a

lternative than indenting for differentiating lists from the rest of the topic. Use the following guideline when creating lists:

Use an equal amount of space between list items and include plenty of white space around the list. Do not introduce different formatting elements within the group, such as bold or color, that differentiate elements and weaken the groups internal structure. In other words, always treat all the items within the group the same. You have more control in print because the page is fixed and is larger than the average Help window. A long nested list in a small window may lose its structure and look like blocks of text stacked irregularly on top of each other.

Tables
Like lists, tables structure information into recognizable groups. However, tables provide more control over the information and allow greater flexibility in how the information is presented.

Generally, information within a table invites users to scan in both directionshorizontally and vertically. Therefore, how you design the table depends on how you want users to read the information. In other words, you must decide which orientationhorizontal or verticalbest suits the information. Vertical alignment in a table suggests that th
e items within the column are the same type (related by information category); horizontal alignment signifies that all items in the row have the same functional relationship to each other.

In printed documents, tables are always a fixed size. In Windows Help, however, you can create relative tables. A relative table is one that adjusts the width of its columns as the user resizes the window. Because the column widths resize dynamically, use a relative table only if you want the information to wrap. If the information in the table re
quires a more controlled presentation, create a nonwrapping table and have the user scroll to see hidden information when the window is too small to see the whole table.
Use the following guidelines when creating online tables:

Although all tables have vertical columns, most tables are read horizontally. When designing horizontally read tables, use ample space between rows but only enough space between columns to separate them. Dont spread the columns out to fill the screen.

You can also emphasize a tables orientation by adding thin rules between columns or rows. Thin horizontal rules can help draw the users eye across the page. However, dont use vertical rules unless users should read the table vertically; the vertical lines draw the users eye down the screen and make horizontal reading very difficult
.
Using slightly different colored backgrounds for columns and rows also emphasizes the tables orientation but is generally a bad idea because the color may destroy the tables structure and interfere with readability.

Most tables use headings to show the tables orientation and to label the information in the columns or rows. If information in the table is read in both directions, you may want to use both row and column headings. These techniques should help make elements within a column more evenly matched. However, this implies that the information within the table can be divided into logical groupings.

Computers do not arrange text and graphics the way you might for a book or magazineby cutting and pasting things where you want them. Instead you have to rely on the word processor to position these elements in relation to each other. Tables provide a good solution to the problem by letting you place text and graphics precisely in a Help topic.

End.

Scanning Information

Acquiring Images

You have a lot of flexibility in how you acquire images for your multimedia titles. Alternatives range from purchasing images from an image bank or professional photographer to capturing images yourself. Costs, image quality, and licensing/copyright issues can affect your final decision. The following methods identify four common alternatives:

Purchasingbuying digital images
Creatingcreating electronic images using specialized software
Scanningdigitizing photographs or flat art using a color scanner Digitizing Videodigitizing video frames using a camera and frame-grabber

Resolution

Several kinds of resolution can affect bitmap quality: screen resolution, image resolution, and pixel resolution. You should understand the differences between all three when working with bitmap images.

Screen resolution is the maximum image area of the computer screen, expressed in horizontal and vertical pixels, for a particular video mode. The standard video mode for the Multimedia PC is 640 pixels by 480 pixels. You'll have to consider screen resolution when establishing a target image size for a scanned photograph.

Image resolution is the size of the digitized image expressed in horizontal and vertical pixels. The image resolution can differ substantially from the screen resolution. For example, say you display a 320-by-240 pixel image on a 640-by-480 pixel display. In this case, the image size is one-half the screen resolution, so the digitized image only fills one-half of the screen. When the image size and screen resolution are identical, the image fills the screen. When the image size is larger than the screen resolution, the screen can display only a portion of the imagerequiring the display software to support scrolling to see other portions of the image.

Pixel resolution can become a factor when you move images between different graphic display modes or computer hardware. Pixel resolution refers to the ratio of a pixel's width to its height (also known as the pixel's aspect ratio). This can cause unexpected distortions in an image that's transferred between machines with different pixel resolution
s. For example, if you capture an image on a device that uses rectangular pixels with an aspect ratio of 1:2, and later display it on a device that uses square pixels with an aspect ratio of 1:1, the image will be distorted. Fortunately, pixel resolution inconsistencies don't occur frequently as most displays use square pixels with an aspect ratio
of 1:1. Also, most capture devices let you adjust the pixel aspect ratio for your system.

Getting Better Results When Scanning

Here are four ways to get the best results with OCR:

- Quality of the original text
- Type of scanner
- Speed (power) of the computer

- Quality of the software

Start with High-Quality Printed Material
The process of optical character recognition is much like a person reading. Like a person's eye, the scanner analyzes light reflected from a page to create a pictorial representation of what is black (the ink) and what is not black (the paper). This picture of the page is stored in the computer where the OCR software tries to chop the black parts
up into individual letters and then guess what each one is.

Generally anything difficult for a person to read will be impossible for an OCR system to convert. This includes such things as smudged text, small type, strange fonts, characters that are too close together, and typewritten characters created with an old ribbon or dirty keys.

The bottom line to OCR efficiency: always start with high-quality printed material.

Choose an Appropriate Scanner
Given good clean text, the next critical element in the system is the resolution of the scanner. Typical scanner resolutions range from 75 to 450 dots per inch (DPI). The higher the resolution of the scanner, the higher the likelihood of accurate recognition by the software. Although some companies claim to be able to accurately recognize text at
200 DPI, 300 DPI is probably the lowest practical resolution.

Another factor that influences the suitability of a scanner for OCR is its method of feeding. The two standard methods are **flatbed and roller-fed**.

Flatbed scanners are like photocopiers in that the artwork to be scanned is placed on a glass plate, covered by a lid, and then passed over by a light.

Roller-fed scanners are like a typical FAX machine, in that artwork is scanned after being fed in through rollers.

Flatbed scanners are well-suited for bound, oversized, or especially small pages. Roller-fed scanners are good for bulk scanning of material with a consistent format. A flatbed scanner with an optional document feeder offers the best of both worlds. Use a Fast Computer.

The power of the computer used in an OCR system doesn't effect the accuracy of conversion. Nevertheless, we recommend the computer have at least an 80386 processor running at 25 Mhz. Also, since most OCR programs require large amounts of memory, having 4 MB of RAM or more can make life a lot easier.

Select Fast, Accurate OCR Software
There are dozens of OCR programs on the market. Most of these are simply software programs that you load into a computer and run, though some high-end programs work with a board that plugs into the computer. These hardware and software combinations are usually faster, more accurate, and more expensive.

There are a lot of options available when buying OCR programs. Some of the more interesting ones can do the following things:

- Read both mono and proportional spaced fonts
- Read dot-matrix output
- Learn new fonts
- Mix text and graphics
- Define frames of text to read

- Retain character formatting attributes Retain columns (tables)
- Save in various word processing formats Incorporate spell checking into the conversion process

Be aware, however, that character recognition systems rarely achieve more than 95% accuracy on anything but the cleanest of text. Even at 99% accuracy, there could be 20 mistakes on a typical 2,000-character page. So, budget time for spell checking and proofreading.

There are almost infinite combinations of specific scanners, computers, and software packages. If you have printed pages numbering in the hundreds, OCR may work for you. If you have massive amounts of printed text, you should consider hiring a data entry service to have it re-keyed.

Capturing from Video

Along with a scanner, you may want to capture images with a video camera. The camera focuses on the image and then transfers that image to the PC through a special interface board that converts the analog video signal to a digital format. This digitizer simplifies the image by combining palette values and setting the resolution. It also converts the digitized image into a standard graphics format such as TGA or PICT, which can be easily converted by the MDK's Convert tool. This setup lets you grab images with the video camera that can be read and enhanced by software the same as a scanned image.
Many different types of cameras exist and they can deliver images in many different formats (such as direct video, composite video, and National Television Standards Committee (NTSC)). After you've captured the image, you still must convert and enhance the image for incorporation within your application. The following paragraphs summarize the steps involved in this process.

Note:
A video camera usually digitizes images faster than a scanner, but it doesn't necessarily provide the best quality for the money. So, unless you're willing to get the best equipment, you might be better off buying a scanner for your basic image capturing.

Set up Your Copy Stand and Lights Properly
A proper setup and lighting can cut hours of image processing time. Using a copy stand as it comes out of the box invites trouble because quite often the lights are set to light the center of the stand. Often, one strong light (say, 1000 watts about five feet from the picture), will give you the proper, even illumination levels. Sometimes it is quite effective to tack the image to the wall and shoot it using a tripod-mounted camera.

Connect the Camera to the PC
To digitize images, you must plug a digitizer board into your computer, install the digitizer software, and connect the camera to the PC through the board. You may also want an extra monitor for viewing the camera image and some test equipment to ensure the best quality. Make sure all video equipment used is properly calibrated for color accuracy and reproduction.

Digitizers only accept RGB input, so you'll need to ensure your VCR or camera can output RGB. If you are capturing an NTSC signal, you'll need to convert it to an RGB signal first by using a special NTSC decoder.

Use a video camera whenever possible as it provides a more stable and higher resolution signal than a VCR. The next best signal is usually a live feed from cable, broadcast TV, or videodisc. Use VCR images only as a last resort. They produce highly unstable signals and low resolution output (around 240 lines of resolution).

Capture Frames
Focus the camera on the object(s) whose image you want captured. Use the video capture

hardware and software to grab the image and store it to disk. All frame grabbers tend to distort the image slightly due to motion. The frame grabbing software is designed to compensate for this to some degree, however, for best results use still or slow moving images. Avoid grabbing in pause as the image is the most unstable in pause mode.

If you are capturing an entire sequence of video frames for an animation sequence, you'll have to shrink the original frames to a size that can be played back at a particular rate by the hardware. The standard Multimedia PC can play back at approximately 10-15 frames per second, with a frame size of approximately 100 pixels by 150 pixels by 8 bits. Reducing the frame size is probably best done with the original videotape before digitizing since the entire sequence will then be a consistent size. Some video boards can shrink a full screen video source. Otherwise, you may need to hire technicians in a TV studio at a greater cost.

Correct the Aspect Ratio
Your targeted aspect ratio is the VGA monitor at 640:480 (or 4:3). The resolution of your camera or video image is bound to be different so you'll want to be sure the capture software you use can convert and correct the aspect ratio.

Enhance the Image
You'll need to crop the image to size and perform any necessary color, edge, or contrast enhancements. Edit the image to clean up any problems introduced when captured. Use the greatest color or monochrome depth possible. For instance, when touching up a 24-bit color image, use a software package capable of handling 24-bit colors. The Windows with Multimedia tools only handle 8-bit color manipulation.

Scale the Image
By scaling the image down to less than full screen, you won't notice the lower resolution of the original video image. If you capture multiple video frames for an animated sequence, you'll have to shrink the original frames to a size that can be played back at a particular rate by the hardware. Contact a television studio for details as this requires professional expertise.

Transfer Images to PC
If you capture digital images on a different computer than your multimedia application development system, you'll need to transfer your images to the development platform.

Collect and Register Resources

You can gather data resources from many sources. You can get images from clip art, scanned photographs, or original computer artwork. Audio can be created, copied, or taken from analog/digital recordings. And tremendous volumes of text exist in multiple formats. There's no shortage of available resources.

The problem is sorting out what you need from what you don't. Using a DBMS to register resources as they're collected can be extremely helpful.

You should design your database to ensure it contains the information about each resource you want to record. Details associated with each record may depend on the type of resource registered. For example, the database record for an audio resource may include whether it's been recorded at 22.05 kHz or at 44.1 kHz. The database record for an image resource might instead include the number of bits used for the image depth.
Another area helped by the use of a DBMS is tracking data resources built from parts of several different resources. This situation will not be uncommon. Your database design should let you register the new resource into the database. For example, say you register an image showing all U.S. presidents. You then use an image editing package to pull
out all presidents elected during the twentieth century. You now have a new image to enter in the database. You may want your database to identify the relationship between the original image

and its derivative image.

Convert the Text

Once you have the text tagged, you then must convert it to your target delivery format (e.g., ASCII or RTF). The many different word processing file formats has generated many different conversion programs. For example, Word for Windows supports the conversion of many different word processing formats into RTF and ASCII. Many word processors also provide utilities to either convert text to RTF or they let you save text in ASCII format. (Understand, however, that you lose all formatting when you save to ASCII.)

All methods of text format conversion involve pattern recognition of some sort. The source file is scanned until a significant pattern is spotted, and then some appropriate action is performedtypically deleting the pattern or replacing all or some portion of it with another pattern. You should investigate the following types of conversion programs:

Specific purposethese programs automatically convert from one specific format to another. An example would be a WordStar to Word conversion utility. To run one of these, you simply provide the name of the input and output files and let the software run. If you can find or write one of these programs that matches your needs, it will proba
bly be the fastest way to make your conversions.

General purposethese programs use some sort of look-up table to make conversions. By editing the look-up table, you can control the pattern searched for and what its replacement will be. These programs can support extremely elaborate patterns, but are only as good as your ability to initially recognize and uniquely define those patterns.

Word processors with Search and Replacea powerful word processor with a strong macro language can often be useful during some stages of text conversion. But it can also be slow and cumbersome if you are working with megabytes of text. Microsoft Word for Windows is an especially powerful example of this category.

Creating Bitmaps in a Program

Windows includes five functions that let you create a device-dependent GDI bitmap object in your program. The first is the CreateDIBitmap function. The others are:

```
hBitmap = CreateBitmap (cxWidth, cyHeight, nPlanes, nBitsPixel, lpBits) ;
hBitmap = CreateBitmapIndirect (&bitmap) ;
hBitmap = CreateCompatibleBitmap (hdc, cxWidth, cyHeight) ;
hBitmap = CreateDiscardableBitmap (hdc, cxWidth, cyHeight) ;
```

The CreateDiscardableBitmap function is rarely used and is not recommended.
In all cases, the cxWidth and cyHeight parameters are the width and the height of the bitmap in pixels. In CreateBitmap, the nPlanes and nBitsPixel parameters are the number of color planes and the number of color bits per pixel in the bitmap. At least one of these parameters should be set to 1. If both parameters are 1, the function creates a monochrome bitmap. (I'll discuss how the color planes and color bits represent color shortly.)

In the CreateBitmap function, lpBits can be set to NULL if you are creating an uninitialized bitmap. The resultant bitmap contains random data. In the CreateCompatibleBitmap and CreateDiscardableBitmap functions, Windows uses the device context referenced by hdc to obtain the number of color planes and the number of color bits per pixel. The bitmap created by these functions is uninitialized.
CreateBitmapIndirect is similar to CreateBitmap except that it uses the bitmap structure of type BITMAP to define the bitmap. The following table shows the fields of this structure:

Field	Type	Description
bmType	int	Set to 0
bmWidth	int	Width of bitmap in pixels
bmHeight	int	Height of bitmap in scan lines
bmWidthBytes	int	Width of bitmap in bytes (must be even)
bmPlanes	BYTE	Number of color planes
bmBitsPixel	BYTE	Number of color bits per pixel
bmBits	void FAR *	Far pointer to array of bits

The bmWidthBytes field must be an even numberthe lowest even number of bytes required to store one scan line. The array of the bits referenced by bmBits must be organized based on the bmWidthBytes field. If bm is a structure variable of type BITMAP, you can calculate the bmWidthBytes field by using the following statement:
bm.bmWidthBytes = (bm.bmWidth * bm.bmBitsPixel + 15) / 16 * 2 ;

If Windows cannot create the bitmap (generally because not enough memory is available), it will return a NULL. You should check the return values from the bitmap creation functions, particularly if you're creating large bitmaps.

The handle to the bitmap is not a handle to a global memory block, so don't try to use the GlobalLock function on it. The handle is instead a local handle to the GDI module's data segment. This handle references a small local memory block in GDI that contains a second handle to a global memory block containing the information in the BITMAP structure and the actual bits.

Once you create a bitmap, you cannot change the size, the number of color planes, or the number of color bits per pixel. You would have to create a new bitmap and transfer the bits from the original bitmap to this new bitmap. If you have a handle to a bitmap, you can get the size and color organization using:

```
GetObject (hBitmap, sizeof (BITMAP), (LPSTR) &bitmap) ;
```

This copies the information about the bitmap into a structure (called bitmap here) of type BITMAP. This function doesn't fill in the bmBits field. To get access to the actual bits of the bitmap, you must call:

```
GetBitmapBits (hBitmap, dwCount, lpBits) ;
```

This copies dwCount bits into a character array referenced by the far pointer lpBits. To ensure that all the bits of the bitmap are copied into this array, you can calculate the dwCount parameter based on the fields of the bitmap structure:

```
dwCount = (DWORD) bitmap.bmWidthBytes * bitmap.bmHeight * bitmap.bmPlanes ;
```

You can also direct Windows to copy a character array containing the bitmap bits back into an existing bitmap using the function:

```
SetBitmapBits (hBitmap, dwCount, lpBits) ;
```

Because bitmaps are GDI objects, you should delete any bitmap you create:

```
DeleteObject (hBitmap) ;
```

End.

Scanning Issues

Preparing and Converting the Text

The first step in creating your topics is to prepare the text for use by Word for Windows. Original text can be scanned or typed, or it might already exist in electronic format containing embedded attribute codes (such as typesetting or text-editor codes). Prepare your text based on the following guidelines:

- New text should be entered with Microsoft Word for Windows.
- Printed text must be retyped or converted to ASCII using an optical character recognition (OCR) scanner, then formatted and saved using Word for Windows.
- Electronic files must be converted to a format readable by Word for Windows.
  All text files must be saved in RTF format from Word.
- Text for a Viewer title can include most types of Word for Windows character and paragraph formatting, including tables. Table borders and titles are not supported. To preserve the alignment of table columns, you might want to format tables so they won't wrap if the user resizes the Viewer window.

To create a nonwrapping table:

1. Select the paragraph containing the table.
2. From the Format menu, choose Paragraph.
   The Paragraph dialog box appears.
3. Select the Together check box.
4. Choose OK.

You can also use the same method to format any type of text you don't want to wrap.

Purchasing Images

Digital image libraries are becoming quite popular. The quality, variety, and resolution of computer-generated and digitized or scanned images are going up as fast as the price is going down.

Note :
Think twice before digitizing images from magazines, books or televisionmost of them are copyrighted. If you find some images that you can't do without, ask permission to use them or you will be violating copyright law. The best places to start are the public relations or marketing departments of the companies claiming copyright ownership. Plan
ahead as it can take several months to obtain permission.

Scanning

Many bitmaps used by a multimedia application will come from photographs. A photograph uses continuous tones and shadescolors that blend smoothly from one to another. By using a scanner or special digitizing equipment, you can transform photographs into bitmap images.

Scanning is the most common way to rapidly create full-color electronic images from photographic prints, slides, or pieces of flat artwork. The main drawback of scanners is that the scanning process is time-consuming. Scanning a large image at high resolution can take up to a minute or more of processing time. This adds up when you have a few thousand images to digitize. If a speedy production cycle matters more than having the highest quality, consider buying a good video camera and a frame grabber.

Image Processing Software

After you've scanned or digitized an image, you may have to do additional processing to prepare it for use in your product. This is where image processing software and paint tools come in.

There is often a fine line of distinction between digitizing software, image enhancement software, and paint software. Some products have all three capabilities built in, while others specialize in one capability or another.

Digitizing Software
At its simplest level, digitizing software controls the scanner or digitizer you are using to capture an image. It may allow you to set the image size, select the portion of the image to digitize, specify the resolution and number of colors, and select the format in which to save the image file. Most of the high-end digitizing software also includes image enhancement and painting features.

You can probably get by just fine using simple scanning software and the MDK's BitEdit and PalEdit tools if you don't need to do extensive editing to enhance your images.

Image Enhancement Software
Image enhancement software is designed to convert images to different formats, spatial resolutions, and color resolutions; to modify saturation, hue, tint, contrast, and brightness; to sharpen or blur edges; to modify palette assignments; and to flip, rotate, crop, and resize. Think of the image enhancement software as a digital photo-retouching product. You normally use it to make global changes to an entire image, such as blurring the background, changing all blues to greens, and so on.

One function that you will probably use on every image you scan or digitize is color reduction. When you scan or digitize a natural image at high color resolution, you capture millions of colors. Most of these colors, however, are simply subtle shades of a relatively few colors.

If you want to display the image on a computer screen that supports only 16 or 256 colors, then you will have to merge or delete most of the captured colors. Scanning and image enhancement software provide a variety of ways to do this algorithmically and manually.

BitEdit and PalEdit from the Multimedia Development Kit provide simple edit functions for images and image palettes. See the MDK's Data Preparation Tools User's Guide for descriptions of these tools.

Paint Software
Paint software is used to actually edit the contents of an image. Paint software is sometimes used in the scanning process to add special effects to an image or to work on color gradations and hues at a pixel level. Use paint software to make minor changes to images. You can cut, copy,

and paste segments of the image, or use tools such as a paintbrush or airbrush to add elements to the image.

File Conversion Software
The images used in your multimedia application can come from a variety of sources and computer platforms. Although your application may be designed to import any graphic format, it will run most efficiently if it imports graphics in Windows DIB format. The MDK's BitEdit and Convert utilities let you convert from a number of the most common graphicformats to DIB and back.

The following table shows the different formats supported by BitEdit and Convert:

Format	Extension	Capability
Apple Macintosh PICT	.PIC	Read, Write
AutoCAD Import	.PLT	Read
CompuServe GIF	.GIF	Read
Computer Graphics Metafile	.CGM	Read
Encapsulated PostScript	.EPS	Read
HP Graphic Language	.HGL	Read
Lotus 1-2-3 Graphics	.PIC	Read
Micrografx Designer/Draw	.DRW	Read
Microsoft RIFF DIB	.RDI	Read, Write
Microsoft RLE DIB	.DIB	Read, Write
Microsoft RLE RIFF DIB	.RDI	Read, Write
Microsoft Windows BMP	.BMP	Read
Microsoft Windows DIB	.DIB	Read, Write
Microsoft Windows Metafile	.WMF	Read
PC Paintbrush	.PCX	Read, Write
Tagged Image File Format (TIFF)	.TIF	Read
Truevision TGA	.TGA	Read, Write

If your images are stored in other formats, you may need to use a two-step approach, first converting them to one of these standard formats and then to the final format.

Scanners

Anytime you want to turn a photographic print, slide or flat artwork into a digital image, use a full-color scanner and scanning software. The scanner builds a digital representation of the photograph and creates a corresponding image file. Scanning software can typically store images in PICT (Apple Macintosh) or PCX (PC Paintbrush) formats. Scanners can produce a far higher image resolution than most cameras. The best scanners digitize at least 300 dots per inch resolution with various color depths per pixel (1 to 24 bits per pixel).

Scanners come in two basic varieties: flat-bed scanners and slide scanners. Flat-bed scanners are used to scan printed materials and photographic prints. Slide scanners are used to scan photographic slides. Use a flat-bed scanner if the bulk of your images are flat art or prints. If most of your images are slides, use a slide scanner for quality reproductions.

Many different types of scanner hardware and software are available. If possible, use a scanner that can capture at 24 bits per pixel and at least 300 dots per inch resolution. You'll probably want to edit and archive your scanned images in the original 24 bit format, and then reduce the images to either 8-bit or 4-bit for the actual application.

This lets you work with the highest quality image until satisfied it's ready for conversion to the lower resolution.

Capturing and Preparing Images

Although scanning and digitization are both relatively straightforward processes, the quality of your final images and the amount of work required to produce them can vary drastically, depending on how well you plan the project. The next few pages describe the following aspects of the image preparation process:

Choosing images with the proper characteristics
Capturing images with a scanner
Capturing images with a video camera Enhancing captured images

Capturing with a Scanner

The best way to describe the scanning process is to explain the steps involved in converting a photographic print to a Windows DIB (Device Independent Bitmap). The DIB format is the standard format for all Windows bitmapsit is the recommended target format for all your bitmapped images.

Each image starts out as a separate photographic print with its own unique palette. The process of getting an image ready for the final application goes something like this:

Adjust Your Monitor

Your images may look great on your display, but if your production monitor is improperly adjusted the final image may not look right when displayed on other monitors. Since you can never be sure how well adjusted the delivery system monitor will be, make sure you have at least created the image in the truest, most accurate color.

The best way to adjust a production monitor is to buy a color bar generator that outputs pure RGB and plug it directly into your monitor. You can also look for software that generates a color bar, and then adjust your monitor settings accordingly. Without these tools, adjustments are purely subjective as they rely solely on your ability to visually evaluate color.

Choose the Image Depth

Many scanners offer several image-depth settings. Because Windows with Multimedia supports 1-bit, 4-bit, and 8-bit bitmaps, any palette can contain up to 256 unique colors. The quality of a scanned bitmap depends on how well a system can re-create the effect of a continuous-tone image using these 256 colors. Whenever possible you should always create and enhance bitmaps using large image depths.

Good color scanners can scan with a color resolution of up to 24 bits per pixel, allowing 16 million colors in the palette. Most scanning programs let you reduce the colors in the image from 24 to 8 bits or less. If you want to scan an image once and store the highest quality image possible, scan it at 24 bits and reduce it after you're totally sa
tisfied with the results. If you don't need the original high-quality image, you might as well reduce it as you scan, since it will require far less space to store.

Adjust Image To Proper Size

Set the size or resolution of the screen image by specifying the number of dots per inch (DPI) in the image. Adjust the DPI setting (sometimes called the scanning resolution) to be as close to the desired image size as possible. VGA screens with 640 by 480 resolution display images at about

72 DPI; a scan setting of 72 DPI produces approximately a 1:1 ratio in size.

You control the size of the screen image with the scanning resolution. To shrink the screen image of an illustration, set the scanning resolution to less than 72 DPI. To enlarge the screen image, set the scanning resolution to greater than 72 DPI.

A quick way to determine the desired scanning resolution is to divide the number of pixels you want to cover on the screen in one dimension by the same dimension of the area of the picture you want to use. For example, say your original image is 10-by-8 inches and you want it to fill half the screen (320 by 240). Divide 320 by 10 and you obtain a scanning resolution of 32 DPI.

Hint:
Although paint programs allow you to resize images, building a digitized image of the correct dimensions with the digitizing software gives you a better image. Paint programs make intelligent guesses when reducing or enlarging an image. In contrast, digitizing software doesn't guess; it uses information from the original illustration to build the
digitized image.

Identify Cropping Boundaries
Always pre-scan the image. Pre-scanning takes only a few seconds and provides a quick, low-resolution scan of the entire scanning bed. Pre-scanning your image lets you set cropping boundaries for the digitizing software and saves time during scanning. Pre-scanning is present in all decent capturing programs.

After pre-scanning, eliminate the portion of the picture you don't want. This not only limits the size of the image file, it also reduces the total number of colors included in its color palette. Since you'll probably have to adjust the image's palette if you want to display it with other images, a smaller palette can simplify this process.

Scan the Image
Now scan in the image. After you've scanned the image, look at it and see what adjustments you might like to make. Scanner software sometimes includes a paint or draw package to clean up any problems introduced during the scan. If you have the time, cut out all unnecessary elements of each image, especially in the background. Again, this makes for a smaller image file and a smaller palette.

Transfer the Image
If you capture digital images on a different computer than your multimedia application development system, you'll need to transfer your images to the development platform. There are several ways to move images from one system to another, but using a network is probably the fastest and most efficient method if you are moving lots of large files, such as images.

Video Frame Grabbing
You can use a video camera hooked to a digitizing board in your computer to capture images. The digitizing board, often called a frame-grabber, converts the analog signal from the camera to a digital format that can be read and enhanced by software the same as a scanned image.
The biggest difference between a scanner and a digitizer is that a scanner can only capture an image from a two-dimensional source, such as a photograph or slide, while a digitizer can capture any video image. A digitizing board usually captures two-dimensional images faster than a scanner, but it doesn't necessarily provide the best quality for the money. So, unless you're willing to get the best equipment, you might be better off buying a scanner to capture two-dimensional images.

Scanning Text

The concept of feeding printed material into a machine that recognizes each letter and feeds it into a text file sounds ideal. Fortunately, such technology exists and is called optical character recognition (OCR). Under the proper circumstances it can be a very efficient way to get text into a computer.

An OCR system consists of a scanner (quite possibly the same one used to scan images), a computer, and some software. The scanner converts a page of text into a bitmapped image, and the software analyzes the letter shapes and converts them into ASCII letters. The number of predefined typefaces is usually limited to less than a dozen, although many systems have a learning facility to include new characters and typefaces.

You scan every page and then you run various utility programs (such as a spell checker) to detect misreads and other scanner errors. For the final step, print and proofread the text. The following table lists some of the benefits and drawbacks associated with using OCR technology.

Benefits	Drawbacks
Scanning requires little upfront labor.	Some scanners can read only a limited set of typefaces. Other scanners read more typefaces, but you first have to train them by running samples through the scanner and then calibrating its interpretation of the text.
An OCR scanner can quickly convert large amounts of printed information into electronic files (less than a minute per page).	Assuming an accuracy rate of 99%, scanned text contains an average of one error for every two lines of text. This error rate can mean hundreds of thousands of errors for long texts. Make sure to schedule time for editing and proofreading.
Scanning usually costs less than re-keying.	You can lose special symbols (such as Greek or other foreign characters) and complex formatting (such as tables, mathematical formulas, or special fonts).

Retyping Text

Although typing is labor-intensive, it is often the most economical method to get large amounts of printed material into a computer. You can either have your staff re-key the text or you can contract with a service bureau. Typists enter the text directly into the system from the printed material.

There are a number of companies offering such a service. These companies generally claim 99.9% or higher accuracy through the use of double or triple key verification (also called double or triple blind typing). This means that they have two or three people type the same material, and then use a computer program to spot differences. The assumption is that several people won't make the same mistake at the same place in the file.

The cost of this service typically varies with the volume and complexity of the original material and the accuracy, turn-around speed, and extra services (such as format tagging) provided by the service company. The following table lists some of the benefits and drawbacks associated with re-keying the text.

Benefits	Drawbacks
Re-keying printed documents is an established type of service, and you can reliably estimate the costs and time associated with such a project.	
	Re-keying usually takes longer than scanning.
The typists can include formatting and structural information into the text as they re-key it. This can reduce the time necessary to prepare the text for the retrieval software.	
	A labor intensive job such as this can become costly. (However, the cost of verifying scanned data is also high.) You still must schedule time for proofreading and correction.

End.

Scanning Tools and Methods

You can use a number of methods to create pictures for your Help file. Each method will influence the visual appearance of the picture as well as your approach to design. If possible, try to match the strengths and weaknesses of the method used to create the picture to the kind of image for which it is best suited.

The resolution and accuracy of any device will affect the quality of the final image. Also, many methods require some ability to draw. Employing a professional designer for this work (unless there is one in the team) should be consideredthey will probably save time and produce better results.

For example, you can use any of the following methods to create pictures:

- Use the Windows screen capture facility to take a screen shot and then edit the bitmap in Windows Paintbrush.
- Use a graphics application to create the picture from scratch.
- Use a scanner to capture existing illustrations or clip art. Use a digitizing tablet to trace an image.
- Import a picture created using a Macintosh graphics application.
  Each picture that you include in the Help file must be saved as a separate file. One way to manage pictures for a project is to create a subdirectory where all pictures are stored, such as an \ART subdirectory in the project directory.

When building the Help file, the Help compiler can compress bitmap files (including .SHG files) if you specify Medium or High using the COMPRESS option in the [OPTIONS] section of the Help project file. On the average, you can expect 25 percent to 50 percent reduction in size using compression.
(For details, see Chapter 16, The Help Project File.)

Picture Formats
Pictures that appear in Help files must be in one of the following standard Windows formats.

Format	Description
.DIB	Windows version 3.x device-independent bitmap
.BMP	Windows version 2.x bitmap
.WMF	Windows metafile
.SHG	Windows Help hypergraphic bitmap
.MRB	Windows Help multiresolution bitmap

Several applications, such as Microsoft Word for Windows, Micrografx® Designer, Corel® Draw, and Aldus® PageMaker® and FreeHand®, and others can save files or export them as metafiles. However, some applications, Aldus FreeHand, for example, use a metafile format that differs somewhat from the standard metafile format for Windows. For that reason, you cant use their files directly. You must convert them to the standard metafile format for Windows or convert them to bitmap files.

Note:
If Help cannot display the file because it is in an unsupported format, you can take a screen shot

of it, paste it into Windows Paintbrush, and then save it as a Windows-based bitmap. If you cant display the picture in Windows, you must use a conversion utility to convert the file to a supported format.

Because Windows-based metafiles are not binary compatible with the Macintosh, you should use only bitmaps if you want to ensure cross-platform compatibility.

Creating Bitmaps from Screen Shots

Another easy way to create art for your Help file is to use the Windows screen capture facility. Using screen shots, you can create a bitmap from anything that you can display in Windowsa window, dialog box, or icon, for example. You can also use this method to convert unsupported picture formats into Windows-based bitmap format. For example, you may want to use a full-featured draw program to create art for your Help file but save the final pictures as bitmaps. You can do that just by previewing the finished art in Windows. When the image displays in Windows, you can copy it to the Clipboard and paste it into a graphics editor such as Windows Paintbrush. Then you use the graphics editor to clean up the image as needed, and save the image in an appropriate format.

To create a bitmap from a screen shot

1. Display on the Windows desktop the element you want to make into a picture.
   For some shots, this may require a special setupstarting the application, opening a document, scrolling to a specific location, and so on.
2. Press ALT+PRINTSCREEN to copy a picture of the active window to the Clipboard.
3. Open Paintbrush.
4. From the View menu, choose Zoom Out.

Zooming out ensures that Paintbrush will capture the entire screen dump

5. From the Edit menu, choose Paste.
6. Select any Paintbrush tool.

Paintbrush displays a zoomed out version of the screen dump

1. From the View menu, choose Zoom In.
   The screen shot appears full size.
2. Make any changes you want to this piece of art.
3. Use the Pick tool to select the portion of the drawing you want to save as the final bitmap.
4. From the Edit menu, choose Copy To.
5. Save the file as a Windows bitmap.

Creating Pictures from Scratch

Both paint and draw programs are now widely available. Paint programs are well-suited to freehand-type illustrations. They include many features such as facilities for brushes, lines and shapes, filling areas with colors and patterns, image manipulation, and so on. Draw programs are better suited for creating mechanical or structured images. Generally, they take longer to learn to use than paint programs, but they allow greater flexibility in editing the details of the drawing. Unlike pictures created by paint programs, pictures created in draw programs can contain multiple layers, each of which can be manipulated independently.

Painting and drawing programs are based on two very different ways of describing a picture. Painting software uses an array of color pixels (or bit maps) to represent the image. Drafting or object orientated software describes pictures as a list of graphic primitives such as line or circle which have a number of attributes such as position and size.

Creating graphics from scratch may pose a large problem for the inexperienced Help author because their use seems to require the ability to draw. Frequently there appears to be a bewildering variety of shapes, styles, colors, patterns, and specialized tools to choose from. However, creating effective graphics often depends to a large extent on applying sound

principles, rather than on pure artisitc skill. The documentation that comes with your graphics editor should provide some direction and guidelines for designing and creating graphics. And if you want to provide a more professional look to your pictures, you can have them created by a competent graphic designer or illustrator.
Microsoft Windows does not include a draw program, but it does offer a basic paint programWindows Paintbrush. You can use Paintbrush to create freehand illustrations and prepare screen images of your application or the Windows environment. The procedure for creating a simple drawing in Windows Paintbrush involves the following steps.

To create a simple drawing in Windows Paintbrush
1. Start Paintbrush.
2. Determine the size of the drawing area.

You define the drawing area by choosing the Image Attributes command from the Options menu. Frequently, it is a good strategy to use the default drawing area (equivalent to the entire screen) while you are creating the drawing so that you have more room to work. When you are ready to save the drawing, you can define the size of the final art.

3. Select a background and foreground color.
   You can change from a color palette to a black-and-white palette (or vice versa) by choosing the Image Attributes command from the Options menu. However, after you begin a drawing, you must stay with the palette you selected.
4. Select a drawing tool.
   In most cases, you will use several tools to create a picture.
5. Draw the picture.
   If you make a mistake while drawing, you can use the Undo command on the Edit menu and the Eraser tool.
6. Make any changes or edits to the picture.
   If you need to see the drawing in greater detail to make fine adjustments, you can use the Zoom In command on the View menu. The Zoom In command magnifies a portion of the drawing so you can change one pixel at a time.
7. When the picture looks the way you want it to, save it as a Windows 16-color bitmap.
   Make sure that you save just that portion of the drawing that you want to appear in Help and not the entire drawing area (if the drawing area is larger than the picture you are creating).

For complete instructions on how to use Paintbrush to create pictures, see the Microsoft Windows Users Guide, version 3.0 or 3.1. If you are using a different graphics editor to prepare your pictures, see the users guide for the graphics application you are using to learn how to create the pictures.

Creating Realistic 3D Graphics
When an object is displayed in two dimensions, as in a drawing, depth relationships, such as whether one line is in front of or behind another, are often ambiguous. On the other hand, when perceiving an actual solid object or scene, the human eye uses certain information, sometimes referred to as depth cues, in order to deduce the spatial structure.

You can include some of these cues, such as occlusion (closer objects overlap those that are further away) and shading, in order to make your pictures more realistic. There are various ways of showing depth with three-dimensional images. You can:

Creating Pictures from Scanned Images
Scanning images is one of the easiest ways to include graphics in the Help file because the most difficult taskcreating the imagehas already been done. You only have to work the scanning software and then perform final cleanup and editing in Windows Paintbrush or some other graphics application.

The problem with most scanned color images, however, is that they need at least 256 colors to look realistic when they are displayed on a computer screen. Because Windows Help does not support 256-color pictures, and also because 256-color pictures often do not display adequately on systems with standard VGA (16-color) video adapters, you are limited in the kind of images you can scan for use in Help files.

Sixteen-color pictures are suitable for diagrams, line drawings, cartoons, and simple drawings, most of which can be scanned successfully. Also, there is an increasing amount of public clip art available that would be suitable to scan. The goal is to offer the highest possible image quality to users with low-end video hardware but avoid short-changing users with more powerful video capabilities. If you can use a scanner to obtain pictures that meet Helps basic requirements, then scanning is a good way to create graphics for your Help file.

Creating Bitmaps with a Digitizing Tablet
A digitizing tablet is the best way to trace the outline of an existing image and ensures that the proportions are correct. A mouse or a tablet can be used to copy a picture by eye, but this requires more skill than tracing. Tablets are based on an absolute coordinate system, while mice use relative coordinates: this affects the physical drawing action and is the principal reason why mice are not good for tracing.

Digitizing tablets can be fitted with a stylus (like a pen) which is good for freehand drawing or a transparent cross-hair cursor which is good for accurate tracing and for positioning elements on screen. Mice are also good for positioning elements.
Creating Pictures on the Macintosh

Many graphics artists use the Apple Macintosh computer as their primary tool for creating art. Although the file formats most commonly used by Macintosh applications are not supported by Windows Help, you can use the Macintosh computer to create art for your Help file. Basically, you have two options:

1) You can use a Macintosh draw or paint program to create the art and convert the files to a standard Windows graphics format.

2) You can insert art in a Word for the Macintosh document and then import the entire document to Word for Windows and capture the art in screen shots.

Importing Art from Macintosh Graphics Applications
Most draw and paint programs that run on the Apple Macintosh have the ability to save files in PICT format. Since there are several ultilities that convert graphics files from PICT to DIB format and from DIB to PICT format, you can create art for your Help file using your favorite Macintosh graphics application and then transfer the art to the PC
so it can be built into the Help file. You can also move original art and screen shots Windows saved in DIB or BMP format to the Macintosh application and edit them there.
To move art back and forth between the Macintosh and PC, you need the following tools:

- A conversion utility that can convert the file formats you are using. Alchemy is one example.

- Network hardware and software that connects the PC and Macintosh computers or software like DOS Mounter that enables Macintosh computers to read and write to MS-DOS 3.5-inch floppy disks formatted on the PC. A palette file in the Macintosh application that matches the standard Windows 16-colors.

Some Conversion Issues
When importing art created by Macintosh applications, you should be aware of the following issues:

- When transferring files from Windows to the PC, the Macintosh graphics application may not be able to see the files in its Open list box. If this happens, it is probably because the converted file has the wrong TYPE and CREATOR set. You can change these settings manually with Macintosh utilities such as DiskTop; however, DOS Mounter and some network software (Microsoft LANManager 2.1, for example) perform automatic extension mapping, which assigns the proper TYPE and CREATOR to a file with a particular MS-DOS file extension whenever a file is moved to the Macintosh.

- Because Windows Help only supports bitmaps with 16 colors, you should create a custom Windows 16-color palette for your graphics application. That will ensure that the art you create and edit on the Macintosh will display with the correct colors when transferred back to Windows. If you want to use dithered colors that are found in Windows Paintbrush, you may need to create those as well, for example as custom brush patterns in Studio/8.

- Some graphics applications on the Macintosh save files with a minimum 256-color depth (8-bit color). If this is true, make sure that the conversion utility reduces the file to 16 colors (4-bit color) when it converts it to a bitmap. Or open the 256-color bitmap in Paintbrush and save the file as a 16-color bitmap. Storing bitmaps on the PC in 16 colors will also keep file sizes to a minimum and conserve disk space.
  MS-DOS only accepts filenames that have 8 characters plus a 3-character extension. Because the Macintosh does not have this limitation, you should follow the MS-DOS 8-character naming convention when naming the art you create on the Macintosh, and use the .PIC extension. Otherwise, the filenames of graphic files you transfer may be difficult to interpret.

Using Batch Files to Control the Conversion Process

If the conversion utility you are using runs in MS-DOS, you may want to create a batch file to control the conversion process. Batch processing ensures consistency in the conversion routine and also allows you to convert many files at the same time. The following example shows one way to set up a batch-file process for converting art between the Macintosh and PC.

The first batch file converts Windows BMP files to Macintosh PICT format:

```
@echo off
break on
if "%1" == "" goto ERROR
echo.
echo        Converting BMP file to PICT format...
echo.
:RESTART
for %%p in (%1) do c:\tadpole\alchemy -mo %%p
shift
if not "%1" == "" goto RESTART
goto MOVEFILE
:ERROR
cls
echo        This batch file converts Windows BMP files to Mac PICT format.
echo        Wildcards and multiple sets are allowed.
echo        Syntax is:
echo.
echo %0 file(s).BMP
echo.
echo where file(s) = separate file names and/or wildcards.
echo.
goto EXIT
```

```
:MOVEFILE
echo.
echo        Conversion complete.
echo        Now moving file to EDITS\PICT folder...
echo.
copy *.pic ..\edits\pict
del *.pic
:EXIT
echo.
echo        All done!
```

The second batch file converts Macintosh PICT files to Windows BMP format:

```
@echo off
break on
if "%1" == "" goto BIGERROR
set loc=%1
echo.
echo ---------------
echo !!! WARNING !!!
echo.
echo        If you are processing art for existing SLM
echo        art files (that have already been 'addfile'd), you MUST first
echo        check out the file(s). Otherwise the edits and changes will be lost! echo.
echo        Press the CTRL+C keys to stop this batch file, or pause
direxist \archive\%loc%
if errorlevel 1 goto ERROR1
goto PICT2ARC
:ERROR1
echo.
echo        \\TOAD\MAC!ARCHIVE\%loc% does not exist.
echo        Please check the spelling of the directory name
echo        or confirm you are starting from the net drive.
echo.
goto EXIT
:PICT2ARC
echo.
echo        Placing copy of PICT files in \\TOAD\MAC!ARCHIVE\%loc%...
echo.
copy *.pic \archive\%loc%
direxist c:\tadpole\%loc%\art
if errorlevel 1 goto ERROR2
goto PICT2C
:ERROR2
echo.
echo        C:\TADPOLE\%loc%\ART does not exist.
echo        Please check the spelling of the directory name.
echo.
goto EXIT
:PICT2C
echo.
echo        Moving files to C:\TADPOLE\%LOC%\ART...
echo.
copy *.pic c:\tadpole\%loc%\art
del *.pic
c:
```

```
cd \tadpole\%loc%\art
:CONVERT
echo.
echo            Converting PICT files to BMP...
echo.
for %%p in (*.pic) do c:\tadpole\alchemy %%p -wo -D -f c:\tadpole\bmp16.pal del *.pic
echo.
echo            Now, addfile or check in the .BMP files...
echo.
goto EXIT
:BIGERROR
cls
echo          This conversion starts from the server drive, copies ALL the
echo          PICT files to the net archive directory, then moves those files
echo          to your C: drive in the appropriate \TADPOLE SLM project
echo          directory and performs the conversion. Part of the conversion echo limits the colors
      of the PICT files to the basic Windows 16 colors. echo.
echo Syntax is:
echo.
echo               PREP4PC (location)
echo.
echo          (location) is the name of the application subdirectory.
echo          For example, the syntax:
echo.
echo               PREP4PC calc
echo.
echo          would place the final converted file(s) in
echo the C:\TADPOLE\CALC\ART directory.
echo.
:EXIT
set loc=
break off
```

Importing Art From Microsoft Word for the Macintosh

If you have art that is stored in Word for the Macintosh files, you can import that art into Word for Windows and use it in your Help file. To convert the art, you import the Word for the Macintosh document to Windows and then take a screen shot of the art and save it in Paintbrush as a Windows bitmap.

To import art from a Word for the Macintosh file:

1. Save the Word for the Macintosh file as RTF.
2. Open the RTF file in Word for Windows.
3. Select the picture you want to use in your Help file.

Note

If you are using Word for Windows version 2.0, do not double-click the piece of art, or Word will convert the picture into an embedded Draw graphic. If that happens, you will not be able to paste the picture into Paintbrush.

4. Copy the picture to the Clipboard.
5. Open Paintbrush.
6. From the Edit menu, choose Paste.
7. Make any changes you want to the picture.
8. When the art looks the way you want it to, save it as a Windows 16-color bitmap.

End.

An Introduction to Digital Typography Using TrueType

by George Moore

Abstract

This article describes the use of TrueType® fonts in the Microsoft® Windows™ version 3.1 graphical environment. It explains the concepts of digital typography and discusses the steps for displaying a bitmapped character on a target device, a process that is invisible to developers in the Microsoft Windows environment.

Introduction

In most digital type systems, whether implemented on a computer or in a printer, the system goes through a few basic steps to display characters on the target device:

1. Load the font outline and feed it to the rasterizer.
2. Scale the outline to the correct size.
3. Fill the outline with pixels, creating a raster bitmap.
4. Transfer the raster bitmap to the display device.

With TrueType® in Microsoft® Windows™ version 3.1, the steps are:

1. Load the font and feed it to the rasterizer.
2. Scale the outline to the correct point size for the correct resolution of the device.
3. Apply the hints to the outline, distorting the contours to build a hinted, or grid-fitted, outline.
4. Fill the grid-fitted outline with pixels, creating a raster bitmap.
5. Scan for dropouts (optional).
6. Cache the raster bitmap.
7. Transfer the raster bitmap to the appropriate display device.

Step 1. Loading the Font

A printer's basic input/output system (BIOS) loads a font when it has to locate the font in ROM, or it can download a version of the font stored in local printer RAM. In a computer, the operating system loads a font when it has to locate the font file on disk. In Windows version 3.1, TrueType fonts are stored as TTF files in the SYSTEM subdirectory of the Windows directory. In the Apple® Macintosh® world, these TTF files are known as the raw sfnt resources. On the personal computer, all TrueType fonts are stored and used in Motorola format using big-Endian byte ordering. All Intel® and other little-Endian implementations of the TrueType rasterizer swap bytes on the fly to maintain binar
y compatibility with the Macintosh. The user can install TrueType fonts by running the Control Panel or through an application that uses the new CreateScalableFontResource function in conjunction with the AddFontResource function.

When the user selects a TrueType font from a menu, Windows attempts to locate the TTF file on the disk and present it to the TrueType rasterizer that is a part of the graphical device interface (GDI) in Windows.

Step 2. Scaling the Outline to the Correct Point Size

Outlines

In a TrueType font, character shapes are described by their outlines. A glyph outline consists of a series of connected contours and lines, which are defined by a series of points that describe second order (quadratic) B-splines. The TrueType B-spline format uses two types of points to

define curves: those that are on the curve and those that are
off the curve. Any combination of off and on curve points is acceptable when defining a curve. Straight lines are defined by their endpoints.

Generally, most professional font foundries such as Monotype store their digitized outlines in their internal libraries in a format known as Ikarus. URW, a company in Germany, developed Ikarus, and it has become the standard international interchange format for unhinted digital fonts. Monotype uses SplineLab, a tool from Project Solutions, to convert the Ikarus data to a raw, unhinted, skeletal TrueType file. You can convert and manipulate TrueType outlines from a number of different formats with shrink-wrapped commercial tools such Font Studio version 2.0 from Letraset and Fontographer version 3.5 from Altsys.

FUnits and the Em Square

In a TrueType font file, the on and off curve point locations are described in font units, or FUnits. An FUnit is the smallest measurable unit in the em square, which is an imaginary Cartesian coordinate square in some arbitrary high-resolution space that is used to size and align glyphs. The number of FUnits per em, or more simply units per em, determines the granularity of the em square. The greater the number of units per em, the greater the precision available in addressing locations within the em square. The em is a historic measurement in typography that refers to the space that completely contains the capital letter M of any given font design. Although this is not strictly true in the world of digitized fonts, it is still a reasonable rule of thumb. All TrueType fonts distributed by Apple and Microsoft are built with an em square of 2048, but the TrueType rasterizer lets you build any font with any arbitrary em square size to a maximum of 32,767 from coordinate locations (16384,16384) to (16383,16383). Outline scaling is fastest if the units per em value is a power of 2.

Scaling a Glyph

The first step in this process is converting FUnits to device space, which depends on the physical resolution of the target output device.

The Cartesian coordinate values in the em square can be converted to values in the pixel coordinate system by the following formula:

(FUnit value) * (point size of the letter) * (resolution)
(72 points per inch) * (units per em)

For example, let's calculate the advance width for the letter p in a particular font (the advance width is how wide the character will be, including the space before and after the letter). This particular character is at 18 points on a VGA resolution screen (96 dpi). The advance width of the letter that is defined in the device-independent space is 1250 FUnits, and the units per em is defined for this font as 2048:
1250 * 18 * 96= 14.65 pixels wide 72 * 2048.

At 300 dpi, the same letter is the same physical size, but because the pixels are smaller, it takes 45.78 pixels to represent the same character. At 1200 dpi, it takes 183.11 pixels. The computed x-resolution size is used for metrics such as the widths of characters; the y-resolution is used for ascender, descender, and linegap values.

In reality, scaling actually involves using the above formula with a standard 2-by-2 transformation matrix. The transformation can be computed as:

Sx(Cos theta) Sy(Sin theta)
Sx(Sin theta) Sy(Cos theta)

Sx is the scale factor formula from above, with the resolution value set to the x-resolution of the target device. Sy is the y-resolution, and theta is the rotation angle in counterclockwise degrees from the vertical. This formula handles all rotations, shearing, and stretching of outlines. Applications for Windows version 3.1 can directly manipulate this transformation matrix with the

GetGlyphOutline function.

An em square size of 2048 may not sound particularly high, but it provides an effective resolution much higher than the actual resolution of the high-end 2500-dpi phototypesetters that are on the market today. If you do the math, you find that a motion of 5.89 FUnits in the em square space actually corresponds to a motion of just one pixel at 2500 dpi at 10-point. Thus, if you use 10-point text with an em square of 2048, your phototypesetter needs an actual resolution of about 15,000 dpi (or 225 million dots per square inch) before the granularity of the em square outline starts to affect the number of pixels turned on or off on the actual physical outline. Phototypesetters with a resolution of 15,000 dpi are not likely to come on the market in the next couple of years.

When you display type on a particular physical device at a specific point size, you gain an effective resolution measured in pixels per em (ppem). The formula for calculating ppem is: dots per inchppem = point size *72 points to an inch.

The raster bitmap patterns formed for a glyph at any particular resolution are the same regardless of how big those pixels are (assuming no optical scaling is going on, which I'll discuss later). For example, the ppem value for a 30-point glyph is 30*96/72, or 40 ppem. The 40 ppem bitmap pattern of the character at 30 points on a 96-dpi device is
exactly the same as the 10-point version of the same character on a 300-dpi device. This is because 9.6*300/72 equals 40 ppem, and 9.6 points rounds to 10 points (if that particular font tells the rasterizer to round values to the nearest pixel).

Step 3. Applying the Hints
Device independent outlines are stored in a TrueType font; unfortunately, most low-resolution devices cannot do them justice. Compromises always occur, usually in the form of the outline fitting poorly to the pixels on the output device. This is where hinting comes in. Hints are used to produce legible characters from the high-resolution outlines
on low-resolution physical devices below 800 dpi (depending on the size of the character).

Almost every outline-scaling system uses a simple method to determine which pixels to turn on or off: If the center of the pixel falls inside the outline, turn it on; if the center of the pixel falls outside the outline, turn it off. You will notice that the left vertical stem of the n encompasses the centers of a two-pixel column and will therefore be two pixels wide. However, the right vertical stem incorporates only a one-pixel center column; therefore, the left stem will be twice as wide as the right stem, even though both stems are exactly the same width in the original outline (shown in red).

Hints are used to distort an outline in a systematic fashion to produce legible text. In this case, the typographer might tell the hinting mechanism to move the two stems slightly further apart (Figure 2).

Figure 2.
In this way, both vertical stems are exactly two pixels wide. The higher the resolution of the device, the less you need hints, because you have more pixels to play with. If you assume the above example was on a 96 dpi screen, at 300 dpi you would have a little more than three times the grid density, and so an off-by-one pixel error at 300 dpi wou
ld only be 1/3 the size of the same off-by-one pixel at 96 dpi. However, even at 300 dpi, hints are important until you start to reach 100 ppem.

Hints distort outlines in the TrueType rasterizer by executing a rich set of instructions that let designers specify how to render character features. These instructions form an assembly language for font hinting. Within the TrueType rasterizer is a software-based interpreter, much like the hardware-based CPU in a computer. The interpreter process
es a stream of ordered binary instructions that take their arguments from the interpreter stack and

place the results back on that stack. All opcodes are one byte, but the data can be either a single or a double byte (word) depending on the instruction.

The following example of TrueType code is a subsection of the hints used for the letter B in Times New Roman®:

```
SVTCA[X]
SRP1[], 44
SRP2[], 4
SLOOP[], 3
IP[], 51, 36, 0
CALL[], 16, 31, 27, 22, 33, 35        // backward Serif 22->21->16
CALL[], 15, 31, 27, 9, 33, 34         // forward Serif 9->10->15 IUP[X]
IUP[Y]
RS[], 8
JROF[], #LRnd0, *
SRP0[], 0
ALIGNRP[], 1
```

The TrueType instruction set is fully programmable, with loops, conditionals, branching, math operations, and so on. As you can see from the example, Monotype has defined a subroutine that contains all the hints necessary to manipulate the serifs of Times New Roman (the CALL[] statements). Besides the good programming practice of saving code space, this also guarantees that all serifs in a particular font are rendered identically at any point size, thus preserving beauty and harmony within a typeface. Because most of the complex problems associated with executing hints can be resolved at compile time, the small, fast, dumb rasterizer can execute the code quickly at run time.

The executable portion of the TrueType code can be classified into three main sections (ranked from least to most specific):

1. The code that is executed when the font is first loaded. This code is called the font program and is stored in the fpgm table in the font file. In the Monotype-produced fonts for Windows version 3.1, the fpgm is generally used for subroutine definitions.

2. The code that is executed any time a point size, a transformation, a device resolution, or a spot size changes. This code is called the control value program and is stored in the prep table.

3. The code that is attached to individual glyphs and is stored in the glyf table.
   This arrangement offers great flexibility for certain operations that specific events can trigger within the rasterizer. The code to manipulate the font-wide serif subroutines would probably go in the font program (fpgm) and actually be called in the individual glyph program. However, the code that ensures that the center of the e doesn't close up at small point sizes would go only in the glyph program. Because the code can be localized to specific events, you can use similar subroutines stored in the font program for all members of a typeface family (light, regular, demibold, bold, heavy, black, and so on), which maintains visual consistency across the entire family.

The control value program is run to set up values in the control value table (CVT), which allows common control values to be applied across all glyphs in a font. For example, you might want all glyphs in a font to jump from a stem width of two pixels at 17-point to three pixels at 18-point, again for visual consistency. You can do so easily by plugging the right values into the CVT in conjunction with the move indirect relative point (MIRP) opcode. You can also use the CVT to pass information between the elements in a composite glyph, a special kind of glyph that can be built up from several elements within a font.

A good example of composite glyphs is the accented characters located above ANSI 0x7F, such as Á. In this case, you can simply take the glyph of the A and merge it with the acute accent to form a completely new glyph that is actually composed of two elements. This saves both code space and hinting time. You can also use composite glyphs in conjunction with any transformation to save space within a font. For example, if the font design allows it, you can create the comma, open single quote, close single quote, open double quote, close double quote, and baseline double quote with various transformations of a single font element that needs to be hinted only once.

If you change the size of the character, the resolution of the device, or the transformation of the outline (by rotation, skewing, and so on), you must reexecute the hints for that specific raster bitmap pattern. In the Microsoft and Apple TrueType system fonts, hints are turned off during any nonstandard outline transformation. Only for rotations of orthogonal angles (90, 180, and 270 degrees) are hints used, because the alignment of the outline to the grid is the same in those cases. Some hinting algorithms that are rotation invariant can be executed in TrueType, but we do not use them. Most other digital font formats do not provide hints under rotation either (however, some versions of the Intellifont rasterizer leave hints on under rotation so that they can use the "black distance" information to embolden the font if a bold version is not available). You can easily turn off the execution of hints under rotation with the INSTRCTRL[] opcode.
Writing raw TrueType instructions can be a time-consuming, error-prone chore, just like writing in any form of assembly language. Good tools, such as TypeMan from Type Solutions,* can help. TypeMan first makes a best-guess for the hinting of the font by running its own autohinter on the glyphs. It then lets you tweak the generated code in a high-level language format that is compiled into raw TrueType instructions.

Like any good compiler, it checks syntax, resolves linking problems, and does other housekeeping duties within the fonts. Because the TrueType instruction set is a superset of all known hinting methods, you can convert any other hints to TrueType hints. Several font foundries are using this approach to produce their initial set of TrueType fonts. They simply take their existing hinted fonts and recompile them in TrueType format. Thus, a large library of fonts can be created rather quickly. Because the hints are transferred directly, the quality of the resulting bitmaps is not degraded.

Other companies, such as Bitstream and Agfa Compugraphic, take their existing fonts, convert their own hints to TrueType hints, and then go back and do extra work on the font. They have developed a production environment that lets them take their existing library of outline fonts and add the hinting and font file structure needed to support new font technologies.

TrueType is the only digital type system available on the personal computer or the Macintosh that provides this arbitrary, user-defined, flexible, algorithmic hinting mechanism. It works for many languages, because the hinting algorithms for Latin fonts are different from those for Kanji, Chinese, Korean, Arabic, Hebrew, Devanagari, Telugu, Kannada, Sinhalese, Bengali, Gurmukhi, and so on.

Step 4. Filling the Outline and Creating a Raster Bitmap
To first fill an outline with pixels efficiently, decompose the curves into scanlines to create an edge list (sometimes called an edge table). The edge list contains all edges of the contours sorted by their smaller y-coordinates so that the filling algorithm does not have to calculate intersections with the outlines at every pixel location.
This is a quick step. All the filling algorithm has to do is to decide whether the center of each pixel is enclosed within the outline. If it is, you simply turn that pixel on; if it is not, leave it off.

Step 5. Scanning for Dropouts (Optional)
"Tune in, turn on, and drop out" was the motto for millions of people during the 1960s, and it can be applied to portions of digital typography today. Tuning the outline for the grid, turning pixels on, and checking for dropouts are three major components of any font-scaling system. A pixel dropout

occurs whenever a connected region of a glyph interior contains two black pixels that cannot be connected by a straight line that passes only through those black pixels.

You can test for potential dropouts by looking at an imaginary line segment connecting two adjacent pixel centers. If this line segment is intersected by both an on-transition contour and an off-transition contour, a potential dropout condition exists. The potential dropout becomes an actual dropout only if the two contour lines continue in both d
irections to cut other line segments between adjacent pixel centers. If the two contours join immediately after crossing a scanline (forming a stub), a dropout does not occur, although a stem of the glyph may become shorter than you want.

Depending on the glyph shapes, dropout control can be a time-expensive operation. Thus, it is an optional step that the TrueType SCANCTRL[] opcode can turn on or off. You can turn SCANCTRL[] on or off on a per-character basis, depending on the size of the glyph (its ppem value). For example, a character such as the letter 'I' generally does not have dropouts even at low ppem values; so you can speed its rasterization by turning SCANCTRL[] off. However, imaging the @ character is difficult because of its delicate curves; you might want to leave SCANCTRL[] on for even large sizes (in some Windows version 3.1 fonts, SCANCTRL[] for the @ symbol is left on up to 60 ppem). The individual font vendor can weigh the tradeoffs with dropout control. For example, a vendor selling barcode fonts probably wouldn't need it at all.

Step 6. Caching the Raster Bitmap
Steps 6 and 7 are not so much a part of the rasterizer as they are a part of the environment in which the rasterizer is implemented.

For most Latin fonts that have only about 200 characters, a cache of any reasonable size makes the speed of the rasterizer almost meaningless. This is because the rasterizer runs only once, the first time you select a font or a new point size; thereafter, the bitmaps are transferred out of the cache to the screen or page buffer. If you are editing a document with a particular font, chances are that 99.9 percent of the time the font is used on the screen as it is being transferred from the cache.

In Windows version 3.1, all TrueType bitmaps at all ppem values are cached automatically in the kernel's private memory area. Because the Windows kernel uses all unused memory in the system as its cache, you can have a font cache of several megabytes. If you are running in protected mode on an 80386, Windows also pages bitmap caches to the swap file when physical memory is filled. Thus, you could have 32 MB of virtual font cache on an 8-MB machine. With the advent of the new virtual disk driver in Windows version 3.1 running in protected mode (called fastdisk), loading cached bitmaps from disk is faster than re-rendering the character from scratch. The kernel uses a least recently used (LRU) system to discard old data when memory fills up or when an application asks for extra memory in low-memory conditions. But generally speaking, not much is ever flushed from the cache because most swap partitions are so large in relation to the data being cached.

This situation is in stark contrast to third-party font-scaling solutions that use a smaller fixed cache that cannot be dynamically allocated to other programs. These take memory away from applications in low-memory conditions and do not make the best use of free memory when it is available.

The caching keys that GDI uses in selecting the pre-rendered raster bitmaps are specific to the transformations applied to the glyph and to the effects of optical scaling (which is a venerable typographic concept that allows for changes in the shapes of the glyphs, depending not only on the ppem value but also on the point size).

Step 7. Transferring the Raster Bitmap to the Appropriate Display Device
This step is also implementation specific. Most printers have a dedicated blting chip that can

quickly blast bitmaps from the local printer cache to the page buffer. Under Windows version 3.1, the hardware in your machine determines how this step is accomplished. If you have a video card that contains a hardware blter (which is generally a swell thing to have under Windows anyway), the font bitmaps can be quickly moved from the cache through a direct memory access (DMA) transfer. If you have a normal video sub
system in your personal computer, Windows transfers the bitmap through software, which is still reasonably quick.

Communicating the metrics information in the font to applications in the system is as important as transferring the bitmaps to the screen. The new TrueType functions return many useful values, such as left and right sidebearing values for individual characters, x-height of the font, em square size, subscript and superscript size values, underline
position, strikeout position, typographic ascent, typographic descent, typographic linegap, and the italic angle of the font (useful for making an insertion cursor match the angle of the font).

For information about TypeMan and other tools, call 603-382-6400.

End.

Raster, Vector, and TrueType Fonts

Previous versions of Windows had two types of fonts: raster and vector. Windows version 3.1 introduces a third typeTrueType fonts.

Raster fonts are stored as bitmaps. These bitmaps are designed for output devices of a particular resolution. GDI typically synthesizes bold, italic, underline, and strikeout characteristics for raster fonts; however, the results are not always attractive. When GDI must change the size of a raster font, aliasing problems can also reduce the attrac
tiveness of the text. Raster fonts are useful for specialized applications in which TrueType fonts are not available. Another possible advantage to using raster fonts derives from the large number of raster fonts that are often present on a user's system; an application could look for the name of a particular specialized or decorative font and use a TrueType font if the specified font was not present.

When an application requests an italic or bold font that is not available, GDI synthesizes the font by transforming the character bitmaps. When an application using only raster fonts requests a point size that is not available, GDI also transforms the bitmaps to produce the font. Because TrueType font families include bold, italic, and bold italic
fonts, and because TrueType fonts are scalable to any requested point size, GDI does not synthesize fonts as frequently as it did for earlier versions of Windows. For more information about this subject, see Section 18.2.5, Font Mapper.

Windows version 3.1 contains a new set of raster fonts. This set, called Small Fonts, is for use at resolutions of less than 8 points. Although TrueType fonts can be scaled to less than 8 points, glyphs this small may not be legible enough for regular use. Because glyphs this small contain very little detail, it is more efficient to use the raster small fonts than to scale TrueType fonts to the small size. (GDI synthesizes bold and italic attributes for the raster small fonts, when necessary.)

Vector fonts are stored as collections of GDI calls. They are time-consuming to generate but are useful for such devices as plotters, on which bitmapped characters cannot be used. (By drawing lines, GDI can simulate vector fonts on a device that does not directly support them.) Prior to the introduction of TrueType fonts, vector fonts were also us
eful for applications that used very large or distorted characters or characters that needed to be perpendicular to a base line that was at an angle across the display surface.

TrueType fonts are stored as collections of points and hints that define character outlines. (Hints are algorithms that distort scaled font outlines to improve the appearance of the bitmaps at specific resolutions.) When an application requests a TrueType font, the TrueType rasterizer uses the outline and the hints to produce a bitmap of the size
requested by the application.

The default font for a device context is the System font, a proportionally spaced raster font representing characters in the Windows character set. Its font name is System. Windows uses the System font for menus, window titles, and other text.
It is possible to have multiple fonts in the system that have the same name (for example, a Courier device font and a Courier GDI raster font). However, applications typically do not present a font name to the user more than onceinstead, they discard duplicates. Applications can control which font is presented to the user when duplicate font nam
es occur by using the lfOutPrecision member of the LOGFONT structure.

Suport For Truetype by MS Windows 3.1

The information below applies to the Microsoft Windows Software Development Kit for Windows version 3.1:

Summary:

Microsoft Windows version 3.1 supports TrueType, a scalable font technology developed by Apple Computer and adapted to Windows by Microsoft. This lists the files in the Software/Data Library that provide technical documentation and tools for TrueType.
Note: To widely disseminate information about TrueType, these files are being distributed through a number of sources.

Each archived file can be found in the Software/Data Library by searching on the name of the file, the Q number of this article, or the file-specific S number listed in the table below. Each file was archived using the PKware file-compression utility.

TTSPEC1	S13442	TrueType Font Files Spec (1 of 3)
TTSPEC2	S13443	TrueType Font Files Spec (2 of 3)
TTSPEC3	S13444	TrueType Font Files Spec (3 of 3)
TTFDUMP	S13445	TrueType Font Display Utility
TTFNAME	S13446	Source Code Parses TrueType Fonts
TTWIN	S13451	TrueType Extensions to Windows
EMBEDDIN	S13447	TrueType Font Embedding Document
TTTALK	S13452	TrueType Technical Talks 1 & 2
LUCIDA	S13448	Lucida Font Typographic Info

More Information:

"The TrueType Font Files Specifications" is a 400-page book that details the following:

- How to construct a TrueType font from scratch (or build a font construction tool)
- The TrueType programming language
- The format of each subtable in the .TTF file and illustrations of table contents.

The TrueType specification is available in Microsoft Word for Windows 2.0 format. Windows 3.1 is required to print this document. The TTSPEC1 archive includes a README.DOC file with printing instructions. Work is under way to convert this document into a Windows Help file for online reference.

The TrueType specification is distributed as three archive files: TTSPEC1, TTSPEC2, and TTSPEC3. These files require 2.5 megabytes (MB) after decompression.
TTFDUMP contains a program for MS-DOS that displays the contents of a TrueType font. TTFDUMP can display an entire font or specific subtables. The TTFDUMP tool, in combination with the TrueType font specifications above, provides information to efficiently debug or explore any TrueType font.

For example, the TTFDUMP utility can display the "cmap" table (that maps character codes to glyph indices) of a font with the following command:

```
ttfdump <fontname>.ttf -tcmap -nx
```

The TTFDUMP utility provides a usage message when no parameters are specified.
The TTFNAME file contains C source code that demonstrates how an application can parse the contents of a TrueType font. Although this particular example opens a font file and locates the font name in the "name" table, it could be readily extended to parse any other structure in a font file. The TTFNAME archive also contains many useful header files that define structures corresponding to the tables of a TrueType font file. This code can be used in an application to

parse the TrueType data stream returned by the GetFontData function in Windows 3.1.

The TTWIN archive contains a 31-page Word for Windows 2.0 document targeted to an application developer who is interested in the capabilities that TrueType adds to Windows 3.1. This document contains many illustrations.

The EMBEDDIN archive contains a text file that describes all the information required for an application developer to add TrueType font embedding capabilities to an application. Font embedding enables an application to bundle TrueType fonts with a document. If a document is transferred to another system, bundled fonts are available for document display and printing regardless of the fonts installed on the target machine.

The TTTALK archive contains the TrueType Technical Talks 1 and 2. These text files describe some of the internal details of TrueType as implemented in Windows 3.1. The first document discusses processing from the time the user presses a key on the keyboard until the character appears on the screen. This processing includes scaling, hinting, drop-out control, caching, and blitting. The second document describes a unique feature of TrueType called non-linear scaling. Nonlinear scaling enables a font vendor to overcome some of the physical limitations of low-resolution output devices.

The LUCIDA archive contains useful typographic information about the 22 Lucida fonts included with the Microsoft TrueType Font Pack for Windows. This file provides tips on line layout, mixing and matching fonts in the Lucida font family, and some history for each typeface.

This information was written by the designers of the Lucida font, Chuck Bigelow and Kris Holmes.

TrueType Fonts Reading List

The information below applies to the Microsoft Windows Software Development Kit for Windows version 3.1

Summary:
This section contains a list of documents available from Microsoft that discuss TrueType fonts with reference to their use in Microsoft Windows operating system version 3.1.
The documents are broadly classified into three categories on the basis of their technical scope. Documents in the "End User" category have information about using TrueType fonts for the Windows 3.1 end user. Documents in the "Application Developer" category discuss the impact of TrueType fonts on Windows applications and contain technical details that let application developers use, and take advantage of, TrueType font technology in their applications. Documents in the "Font/Tool/Curious Developer" category contain information useful to developers interested in developing their own TrueType fonts or font-development tools, or interested in knowing about the inner workings of the TrueType font technology.

Each entry contains information to help the reader obtain a copy of the document. Many of the following documents will be available as part of the upcoming Microsoft Developer Network (MSDN) CD. For more information about this CD, call the Microsoft Developer Services Team at (800) 227-4679.

End User Oriented Documents

Title: *"Fonts in Windows 3.1"*

What to expect: This section gives a comprehensive introduction to the font technologies (including TrueType) used in Windows 3.1.

Finding a hardcopy: This document is available as an application note (WW0528); call Microsoft Product Support Services at (206) 637-7098 to request this application note. It is also available in the Software/Data Library (on CompuServe [GO MSL] and Microsoft OnLine) as document S13367. Similar information can also be found in Chapter 9, "Fonts," of the "Microsoft Windows Resource Kit" manual.

Title: *"Microsoft TrueType Font Pack User's Guide"*

What to expect: This document contains notes on installing and using TrueType fonts, designing documents with TrueType fonts, and a brief history of the TryeType font technology.

Finding a copy: This document is shipped with the Microsoft TrueType Font Pack for Windows. The font pack is an assortment of 44 TrueType fonts. Call Microsoft Consumer Sales at (800) 426-9400 for purchasing information.

Application Developer Oriented Documents

Title: Chapter 18, *"Fonts," in the "Guide to Programming"*

What to expect: This document contains a gentle introduction to using fonts in applications. It begins with a discussion of the fundamentals of fonts, introduces the three font technologies used in Windows 3.1 (with an emphasis on TrueType fonts), discusses TrueType font technology in the context of WYSIWYG and document/printer/platform portability, and finally delves into the details of using fonts (with due attention paid to TrueType) in Windows 3.1 applications.

Finding a copy: This manual is shipped with the Windows 3.1 Software Development Kit (SDK). All the information is also available in the SDK 3.1 online documentation (WIN31WH.HLP) under the subject headings "Font Fundamentals," "Fonts in Windows," "TrueType Font Technology," and "Using Fonts in Applications."

Title: *"Windows Font Mapping"*
Author: Ron Gery

What to expect: This document discusses the Windows 3.1 font mapper and how it controls the realization of fonts. Attention is paid to selecting TrueType fonts and the effect of having TrueType fonts on the fonts selected by applications developed for Windows 3.0 (which does not have TrueType fonts).
Finding a copy: This document is available as part of the MSDN CD.

Title: *"Using TrueType"*
Author: Ron Gery
What to expect: This document gives a basic introduction to using TrueType in an application under Windows 3.1. It discusses most of the TrueType functions, enumerating and selecting TrueType fonts, positioning TrueType characters in documents, and page layout.

Finding a copy: This document is available as part of the MSDN CD. It is also available on CompuServe as file T2API.ZIP in LIB 8 of the MSDNLIB forum.

Title: "TrueType and Microsoft Windows Version 3.1"
Authors: David Weise and Dennis Adler

What to expect: This document introduces some font concepts and details the various aspects of using TrueType fonts in applications developed for Windows 3.1.
Finding a copy: This document is available as part of the MSDN CD. It is also available in the Software/Data Library (on CompuServe [GO MSL] and Microsoft OnLine) as document S13451 and on CompuServe as file TT.ZIP in LIB 8 of the MSDNLIB forum.

Font/Tool/Curious Developer Oriented Documents

Title: *"Advanced TrueType: GetGlyphOutline"*
Author: Ron Gery

What to expect: This document presents an explanation of the GetGlyphOutline function. It complements the definition of GetGlyphOutline found in the Windows SDK 3.1 reference manual and extends the definition by providing explanations for those portions of the GetGlyphOutline function that were omitted in the SDK 3.1 manual.

Finding a copy: This document is available as part of the MSDN CD and on CompuServe as file GLYPH.ZIP in LIB 1 (and eventually in LIB 9) of the WINSDK forum.

Title: *"An Introduction to Digital Typography Using TrueType"*
Author: George Moore

What to expect: This transcript of a technical talk lists the basic steps that a typical computer system goes through in order to display characters on the output device and details these steps for TrueType fonts in Windows 3.1.

Finding a copy: This document is available as part of the MSDN CD and as in the Software/Data Library (CompuServe [GO MSL] and Microsoft OnLine) document S13452.

Title: *"Linear vs. Nonlinear Scaling"*
Author: George Moore

What to expect: This transcript of a technical talk describes the importance of being able to nonlinearly scale a font. It discusses how a TrueType font vendor can use nonlinear scaling so that low-resolution devices can display high-quality TrueType fonts.

Finding a copy: This document is available in the Software/Data Library (CompuServe [GO MSL] and Microsoft OnLine) as document S13452.

Title: *"TrueType Font Files Specification Version 1.0"*

What to expect: This specification details how to construct a TrueType font from scratch (or build a tool to do so), the TrueType programming language, and the complete format of each subtable contained in the .TTF file. It also contains illustrations.

Finding a copy: This document is spread over three documents (S13442, S13443, S13444) and is available in the Software/Data Library (CompuServe [GO MSL) and Microsoft OnLine). This specification is also available on CompuServe as a Windows 3.1 Help file (TTHELP.ZIP) in LIB 1 (and eventually in LIB 9) of the WINSDK forum.
Additional reference words: 3.10 true type
KBCategory:
KBSubcategory: GdiTTCustom

TrueType Font Converters and Editors

Summary:
The text below lists a number of commercial software tools that convert existing fonts to TrueType fonts or help build new fonts. The tools are listed in alphabetical order by name. None of these tools is recommended over any other, nor over any that are absent from the list. Each of the tools

has its own strengths and weaknesses. Before making any purchase, examine the tool and its documentation to see if it meets your needs. This list will be updated as more tools become available.

Some of the following tools run on an Apple Macintosh while others run on an IBM PC/AT or compatible computer. The same TrueType font can run on a Macintosh or with Microsoft Windows operating system version 3.1.

To convert a font from the Macintosh to Windows 3.1, save the data fork from a Macintosh font to a file, copy the file to an MS-DOS-formatted disk, give the file a .TTF file extension, and use the Windows 3.1 Control Panel to install the font.

The products included here are manufactured by vendors independent of Microsoft; we make no warranty, implied or otherwise, regarding these products' performance or reliability.

AllType V 1.0
Author:
Atech Software
5964 La Place Court, Suite 125 Carlsbad, CA 92008
(619) 438-6883
Description: Character-based application running under MS-DOS. Converts almost any font format to any other. Supported formats include TrueType, Type-1, Type-3, Nimbus Q, and Intellifont.
AllType V 1.0 is available from the Microsoft Library.

Evolution 2.0 for Macintosh
Author:
Image Club Graphics, Inc.
1902 11th St. SE, Suite 5
Calgary, Alberta, Canada T2G 3G2 (403) 262-8008
Description: Converts almost any font format to any other. Supported formats include TrueType, Type-1, and Type-3.

FontMonger for Windows
FontMonger for Macintosh
Author:
Ares Software
561 Pilgrim Drive, Suite D
Foster City, CA 94404
(415) 578-9090
Description: Converts almost any font format to any other. Supported formats include TrueType, Type-1, Type-3, and Intellifont. Also provides minor font editing by creating composite characters or rearranging the characters in a font. FontMonger for Macintosh V 1.0.5 is available from the Microsoft Library.

Incubator for Windows

Author:
Type Solutions, Inc.
91 Plaistow Rd
Plaistow, NH 03865
(603) 382-6400
Description: Supports adding effects to TrueType fonts. Contact Type Solutions for more information.

Metamorphosis Professional for Macintosh

Author:
Altsys Corp.
269 W. Renner Rd
Richardson, TX 75080
(214) 680-2060
Description: Converts almost any font format to any other. Supported formats include TrueType, Type-1, Type-3, and PICT format. One interesting feature allows the user to read a Type-1 font from the ROM of an Apple LaserWriter printer and convert the font to another format.

FONT EDITORS

Fontographer 3.5 for Windows
Fontographer 3.5 for Macintosh
Author:
Altsys Corp.
269 W. Renner Rd
Richardson, TX 75080
(214) 680-2060
Description: A complete font editing tool. Supports creating a font from scratch and modifying existing fonts. Supports a variety of formats including TrueType and Type-1. Includes an autohinter. Windows version shipped of July 1992.

FontStudio 2.0 for Macintosh
Author:
Letraset Graphic Design Software
40 Eisenhower Dr
Paramus, NJ 07653
(800) 343-TYPE or (201) 845-6100
Description: A complete font editing tool. Supports creating fonts from scratch and modifying existing fonts. Supports a variety of formats including TrueType and Type-1. Includes an autohinter.

TypeMan 1.0 for Macintosh
Author:
Type Solutions, Inc.
91 Plaistow Rd
Plaistow, NH 03865
(603) 382-6400
Description: Designed for font foundries, this tool provides precise control over the hints in a font and includes an autohinter. Supports specifying a font in a high-level programming language, which the tool compiles to the binary TrueType format.

End.

Lava Flow Simulator

Run Sample

Lava Flow Simulator

This code demonstrates palette animation and pop-up menus. Click on the program's window to get a menu. If you make the application 'full screen' (by double-clicking on the title bar) it makes the area the full size of the screen (and not just the usual client area which excludes the title bar). This allows creation of a full screen image.

The image can be copied to the clipboard and then pasted into paint brush and then saved as a DIB. You can then load the image (from control panel) as a desktop image. The image will continue to palette animate (even on the desktop) while the program is still running. You can iconize the application and leave the palette animation going.

Include with the windows SDK is the program MyPal. This displays the current physical palette from windows. This is an interesting program to run during palette animation.

End.

MergeDIB.c

MergeDIB main loop etc.

About MergeDIB.c

Merges a primary DIB (with any palette) and a secondary DIB (with a different palette) into a single, merged DIB (with special palette).

The special dib and palette are a combination of the two images and palettes so that when the palette is gradually crossfaded (animated), the first and second DIB are partially displayed. At complete fade, only one of the bitmaps is 'visible', while at a 50-50 mix, both are equally visible (merged). Pixels are not dithered betweent the images, but are mixed in the palettes.

This technique is limited in the number of pixels that have different targets between bitmaps, but it can create very nice effects when just text is 'faded' in for the target bitmap.

This code is limited so that the two DIBs must be the same size, but this limitation could easily be eliminated by creating an artificial bitmap that is the desired size with a 'blank' (where blank is a chosen color) background. The smaller image could be centered or otherwise placed in the background (easy to do using the DIB Driver).

End.

Midikeyb.c

DESCRIPTION:

This is a custom control that implements a MIDI style keyboard. It's not exactly the best keyboard for a jam session; don't expect to hit a bunch of hemidemisemiquavers at correct tempo...

In any case, this custom control responds to and sends MIDI short messages. This could be expanded on, but this is a good start. The right mouse button is used for 'sticky keys' and will toggle the state appropriately. If you drag the pointer off of a key with the left mouse button pressed, you will notice that the key stays down. This is because I do not SetCapture--and I should.

Another enhancement might be to add CTRL key and SHIFT key modifiers to allow key selection like a multi-select list box.

Before you can use this control, you MUST first export the window procedure for the control:

EXPORTS midiKeyBProc
You then need initialize the class before you use it:

```
if ( !midiKeyBInit( hInst ) )
die a horrible death
      else
you are good to go
```

The colors used by the control default to black and white (black for the accidental keys and white for the normal keys). This should work just dandy on all displays.

To select your own colors, you can send the KEYB_SETFGCOLOR and KEYB_SETBKCOLOR messages to set the foreground (accidental) and background (normal) key colors. The lParam is the RGB() value--wParam is a BOOL telling the control to repaint immediately if set to TRUE.

The layout of the keyboard is set using the MIDI key values with C0 being MIDI note number 0, C2 is 48 (middle C), etc. The default layout is: start = 48 (middle C), for 36 keys (3 octaves), ending on 83. I chose this by throwing darts at my MIDI poster...

You can set the layout to whatever you want by sending the KEYB_SETLAYOUT message to the control. The HIWORD() of lParam is the starting key, the LOWORD() of lParam is the number of keys. wParam is a BOOL telling the control to repaint immediately if set to TRUE (actually non-zero...). You can get the current layout by sending the KEYB_GETLAYOUT message--the LONG return value can be decoded as the lParam of KEYB_SETLAYOUT is encoded.

You also have the option of having labels printed on the keys for MIDI note referencing. Set the KEYBS_LABELS style flag to get labels.

The font used for the label text can be set using the standard WM_SETFONT message. You can get the current font at any time with the WM_GETFONT message.

For other messages and the documentation, see the header block for midiKeyBProc at the end of this source file.

End.

Some Notes on the Use of Palettes

Use DIB_PAL_COLORS with SetDIBitsToDevice to avoid color matching by GDI. Use an identity palette - this is detected by the device driver. The DIB driver expects RGBQUAD data structure for color matching. it does not use palette indexes.

DIBINDEX macro forces the DIB driver to use color index rather than color mapping. Use StretchDIBits to transfer data from the DIB driver image to the screen. BitBlt doesn't work because the DIB in the DIB driver is not owned by the same device driver as the screen.

StretchDIBits performance is enhanced by using DIB_PAL_COLORS and a one-to-one palette. This avoids any color matching.

DIB_RGB_COLORS: The DIB's color table contains RGB values DIB_PAL_COLORS:

The DIB's color table contains logical palette index values

PALETTERGB	(r,g,b)	02, red, green, blue
PALETTEINDEX(i)	01, index	
PC_RESERVED	0x01	/* palette index used for animation */
PC_EXPLICIT	0x02	/* palette index is explicit to device */
PC_NOCOLLAPSE	0x04	/* do not match color to system palette */

Memory DCs: using CreateCompatibleBitmap on a memory DC results in a bitmap the same format as the one currently selcted in the DC. (Default is mono). So if no bitmap has been selected into it, you will get a mono bitmap. Output device DCs always return the native format. GDI uses SetDIBits to convert DIBs to DDBs.

SetDIBits with DIB_RGB_COLORS will create index values by mapping the the DIB color table entries to colors in the current logical palette. I have noticed that doing this in a compatibleDC doesn't work, but doing it with a DIB driver DC works fine. SetDIBitsToDevice is not recomended. Use StretchDIBits instead. If StretchDIbits is slower than SetDIBits + BitBlt then the driver sucks.

Painting a DIB:
SelectPalette(hDC, hPal, FALSE); // select pal as foreground RealizesPalette(hDC); //map to hardware now. When drawing to a memory bitmap, you must select the correct palette first. BitBlt, color -> mono conversions. If the pixel color == the background color then the mono output is white (1) otherwise it is black (0).

BitBlt: You cannot use BitBlt to move data from a DC owned by one device driver to a DC owned by a different device driver. If one DC is a DIB driver DC and the other is a screen DC:
screen dc -> GetDIBits -> DIB dc -> StretchDIBits -> screen dc

So to be able to use blit operations like xor and and to do transparency masking onto a DIB dc, the sprite image and mask must also be in DIB DCs. Since the DIB driver works on the packed DIB format used by the clipboard as CF_DIB, it is most convenient to keep DIBs around in this form. For best screen update performance we need to use StretchDIBits to move data from the DIB dc to the screen DC with the DIB_PAL_COLORS option. This avoids a needless color translation. (Why do we still need an identity palette then?)

The app uses a single palette based on the color table found in the DIB used as the background. Sprite DIBs should be kept to <64k in size so that a FAR pointer rather than a HUGE pointer can be used to access it. That means a bitmap of 100x640, 200x320 or 250x250 is about as big as you can have. Be very careful when manipulating the off-screen DIB memory that you account for

the 64k segments or just use a HUGE pointer in C.

An address wrap while doing a fill operation on the DIB bits will cause the BITMAPINFOHEADER and color table to get trashed since these occupy the start of the memory area occupied by the DIB. Timing measurements are done with the MMSYSTEM function timeGetTime rather than the regular Windows function GetTickCount because timeGetTime returns a millisecond count accurate to the nearest millisecond and GetTickCount returns a millisecond count only accurate to the nearest 55ms (one DOS clock tick - hence the name).

Sprite timings in ms

Description	Cycle	Bkgnd	Sprite
StretchDIBits with DIB_RGB_COLORS	300	12	280
StretchDIBits with DIB_PAL_COLORS	75	35	35
Primitive C code rectangle copy	12	-	-
C code with transparency using huge ptrs	40	1.8	31
Using 16 bit assembler to copy lines	17	2	6
Using 16 bit assembler to copy blocks	16	1.6	5.5
Using 32 bit assembler to copy blocks	16	1.6	4.6
Optimized 32 bit assembler	5.6	0.6	2.6

End.

A Complex App

This is one of the most complex sample applications that I've seen. I tend toward liking 'real' examples, and I find that the simple samples don't really show people what is needed. Therefore, I have taken this (big) step into doing a pretty complex sample app.
If you have strong feelings about the complexity (good or bad) please send your comments to me (see below).

1) using the DIB driver to draw directly into DIB memory

2) writing 286 and 386 specific code (in assembler, obviously)

3) using custom resources

I wanted to expand this example to show a better display for the example, and I got into the 3D stuff. However, I didn't have enough time to get everything done. Instead of just releasing this file and none of the other parts, I decided to release both.

Start with Tri first. TriQ is much more complex. Also take a big grain of salt with the 4D libraries... almost nothing has been tested, but I thought you'd like to see them.
If you run into any bugs, and especially if you fix any of them,
please let me know.

To run this app, access it through the CD drive: \source\triq.

You can FAX me at 206 883-8101 or send US Mail to
Matt Saettler
Microsoft
One Microsoft Way
Redmond, WA 98052
Thanks,
Matt

End.

What Are Broderbund's Living Books?

- A step beyond ordinary electronic books, Living Books are more than spoken words and pictures. They're stories touched by magic and brought to life with characters and pictures that talk and move and sing and dance.

- Wonderful worlds, begging to be explored open up at each turn of the page. With sound effects, original music, humor and lots of animation, it's a whole new learning experience.

- Simple to use means kids just point and click. The kids are in control, so they can re-play a favorite part, go back, or skip ahead as they proceed at their own pace.

- Reading skills are increased through word recognition. Words or sentences can be played individually, and the computer reads (or even spells) words aloud, as they are highlighted.

- Multilingual versions are in each book, and changing languages is only a click away. Arthur's Teacher Trouble includes English and Spanish. Just Grandma and Me includes English, Japanese and Spanish.

Children don't just read Living Books, they live them.

Boderbund's Living Books

NOTE: To run demo copy all files to the hard disk.

Broderbund Software

Disclaimer

This agreement is made as of September 15, 1993 by and between Broderbunnd Software, Inc. and "Broderbund" and Microsoft.

Broderbund hearby grants Microsoft the permission to copy the "DPCDEMO" standalone (created in Broderbund's Kid Pix Companion for Windows) to their CD-ROM solely for use in promoting multimedia during their road shw, "Multimedia Jump Start Developers Conference." The road show occurs from Sept 28-Ocoteber 15, 1993. The above CD-ROM will be freely distributed to attendees of the "Multimedia Jump Start Developers Conference."

Rebecca Knievel,
Product Manager of Kid Pix Companion for Windows,
Broderbund Software, Inc.

BRØDERBUND SOFTWARE, INC. - FACTS

OVERVIEW
Brøderbund Software, Inc. (NASDAQ: BROD) is a leading developer and publisher of computer software for the home, school and small business markets. Founded by Doug and Gary Carlston in 1980, the company employs 350 people and is headquartered in Novato, CA.

PRODUCTS
The company has sold over 15 million units of software and maintains a reputation for producing easy_to_use, high_quality products for popular home computers. Brøderbund's award_winning titles include *The Print Shop*®, *Brøderbund's Living Books* and the *Carmen Sandiego*® series. Annual sales for fiscal 1992 were $75
million.

SENIOR OFFICERS
Doug Carlston: Chairman and CEO
Ed Auer: President and COO
Bill McDonagh: Senior Vice President and CFO

SALES OFFICES
Los Angeles and Novato, CA
Atlanta, GA
Chicago, IL
Boston, MA
Philadelphia, PA
Dallas, TX

AFFILIATED LABELS
Waterford Institute
Asciiware
New World Computing
Binary Zoo
Amtex
Books that Work

CONTACTS
Customer Service/Software Direct (800) 521_6263
Technical Support (415) 382_4700
Public Relations (415) 382_4569
Investor Relations (415) 382_4449

Kid Pix - A Program Overview

What Is Kid Pix?

Kid Pix is an amazing paint program created just for kids that combines special effect art tools, picture stamps, sounds and magic screen transformations. The program allows parents and kids to create wacky and wonderful works of art by combining an assortment of creative paint tools with zany sound effects. And Kid Pix features an easy_to_use palette of painting and special effect tools, including a "drippy paint" brush, a collection of magical erasers even a talking alphabet!

Key Features

- Every brush and tool has its own unique sound effect, from creaks and kerchunks to boings and booms
- From ice cream cones to frogs, kids can choose from over 100 full_color stamp images
- Talking alphabet in both English and Spanish helps kids learn their ABC's
- The extensive palette of over 20 *Wacky Brushes* allows users to create imaginative, highly textured drawings
- Users can record personal greetings, poems, music or sound effects*

Pricing

Available in stores for approximately $39.95

System Requirements

Macintosh: Mac Plus, SE, SE/30, II Series, Classic and LC; System 6.0 or higher; System 6.0.7 or higher for recording sound. 1 MB of RAM for monochrome monitors and 2 MB of RAM for color monitors.

MS_DOS: IBM/Tandy and 100% compatibles; Hard disk and mouse required; Video modes supported: VGA, MCGA, EGA Tandy; Sound card recommended.

Windows: IBM/Tandy and 100% compatibles; Windows 3.0, 3.0 with Multimedia Extensions, or Windows 3.1; 2 MB of RAM; Video modes supported: VGA. Hard disk, mouse, and sound card required.

For additional information contact:

Dawn Montoya (415) 382_4637, Brøderbund Public Relations

*Sound recording requires the Farallon MacRecorder® and System 6.0.7 or higher, or a Macintosh® IIsi or LC.

End.

Crawford Communications Demo

Crawford Communications is based in Atlanta, Georgia. They deal with all aspects of Multimedia design and production.

Sample AVI file from Crawford Communications

{ewl MVMCI2, ViewerMCI, [device AVIVideo][stdcontrol]samp_aps.avi}

Foundation Technology - Video clips

Foundation Technology sample video clip.

{ewl MVMCI2, ViewerMCI, [device AVIVideo][stdcontrol]samp_aps.avi}

Frost
Capital
Partners

Four Embarcadero Center
Thirty First Floor
San Francisco, CA 94111
Tel 415.274.8899
Fax 415.274.8885

Investment Bankers to
the Entertainment, Education
and Multimedia Software Industry

Frost Capital Partners

Founded in 1986, Frost Capital Partners is a private investment banking partnership providing financial and management advisory services to businesses and investors in the entertainment, education and multimedia software industry. The Partners have over thirty years experience in high technology, and have completed nearly one hundred transactions in service, manufacturing, development, and distribution.

FCP is a specialist in the entertainment, education and multimedia software industry and is, to our knowledge, the industry's most active firm in mergers, acquisitions, and debt/equity placements. The Partners are financial advisors to a *who's who* of the software industry and serve on the boards of several entertainment, education and multimedia software clients. The Partners' knowledge of industry dynamics, and ability to communicate that knowledge, has generated significant benefits for both investors and growing companies.

FCP is active in industry conferences, provides research on public offerings and transactions, performs valuations of public and private companies, and maintains a database of over 500 entertainment, education and multimedia software companies. The Partners are often requested to provide expert legal testimony involving software companies, and to comment on industry trends.

Value to clients comes from industry experience, a wide range of contacts, and financial expertise.

Ø **In-depth industry knowledge**. ***FCP*** understands the entertainment, education and multimedia software industry from the ground up. The Partners have helped many companies and investors define the factors which determine success in the industry, and ultimately, to achieve it.

Ø **Extensive contact network**. ***FCP*** knows the industry players, and those waiting on the sidelines. The Partners have identified over 500 firms in the industry and 200 potential investment groups. ***FCP*** experience in linking technology and capital has made the Partners a focal point of contact for both groups.

Ø **Bias toward strategic partnerships**. ***FCP*** has significant experience in aligning clients with investors who can provide not only capital, but ideas, distribution, content, and/or human resources.

Ø **Systematic approach**. ***FCP's*** step-by-step methodology, attention to detail, and exhaustive

follow-up with potential strategic partners ensures that an acquired company will choose among the best capital sources available, *and* that an investor will make the right acquisition.

Ø **Independent of capital**. ***FCP*** is not tied to any one capital source. As a result, the Partners continue to be objective in making recommendations to clients.

Ø **Understanding of corporate culture**. ***FCP*** understands the critical nature of the corporate environment. The Partners' transactions meet financial, strategic, *and* cultural criteria.

Frost Capital Partners' SERVICES

FCP maximizes client value by combining our strategic industry perspective with management experience and a detailed knowledge of transaction execution. The Firm is generally retained for one of the following classes of service:

Acquisitions: With nearly 100 successful transactions completed, the firm has substantial experience structuring, negotiating, and concluding acquisitions. ***FCP*** assists clients with all aspects of buy-side transactions, from analyzing market trends and pricing, to developing cash flow models and acquisition financing. Our experience as investors and operators in the personal computer industry enables the firm to develop acquisition strategies consistent with corporate goals.

Mergers/ Divestitures: ***FCP*** helps clients achieve full value when merging or divesting assets. The firm provides a comprehensive solution including: analyzing and positioning assets to enhance value, preparing offering documents, identifying buyers, negotiating with third parties, determining implications of alternate deal structures, and coordinating other professional services to complete the transaction.

Debt and Equity Financing: ***FCP*** is experienced in evaluating and arranging financing alternatives to grow profitable companies or restructure troubled situations. The firm's ownership in a merchant bank, and close relationships in the financial community provide clients with unique access to a variety of capital sources.

Distribution Access Consulting: ***FCP*** assists clients with identifying and gaining access to domestic and international distribution channels. ***FCP's*** ownership in several distribution channels provides clients with additional access to major markets, and a variety of hard to reach revenue sources. In selecting and negotiating an appropriate distribution channel, we balance the issues of cost, service, specialization, and market presence.

Corporate Development Strategy: As owners of personal computer industry companies, and consultants to start-ups and *Fortune 500* firms, ***FCP*** works with clients to determine growth, cost reduction, customer service enhancement, or other strategic objectives. The firm delivers a complete solution, from determining strategic objectives to developing and implementing programs to ensure these objectives are met. Particular areas of expertise include: manufacturing strategy, cost reduction, marketing, customer service enhancement, and growth and exit strategies.

Business Valuations: Determining fair market value for privately held companies is a critical component of many transactions and refinancing. ***FCP's*** proprietary databases, investment banking tools, and thirty years of industry experience enable buyers, sellers, and lenders to depend on us for well-reasoned analysis and sound judgment.

CAPITAL FORMATION:

BENEFITS OF USING FROST CAPITAL PARTNERS

FCP brings several important advantages to the process of finding an appropriate strategic investor or other capital source:

Ø ***FCP*** is deeply involved in the entertainment, education and multimedia software industry, and the Partners have an extensive network of contacts throughout this industry.

Ø ***FCP's*** investment banking expertise has been focused on arranging mergers, acquisitions, and private placements. As a result, the Partners have formed relationships with potential strategic investors in the entertainment, education and multimedia software industry, both domestically and internationally. Investors include major movie distributors, book publishers, major media/magazine publishers, cable operators, telecommunication firms, diversified entertainment enterprises, personal computer manufacturers, and other software developers.

Each ***FCP*** partner is responsible for ongoing contact, face-to face meetings, and written communication with these potential strategic partners. A current summary of the potential strategic partners/investors ***FCP*** has identified includes:

National and International Book Publishers	78
Print Media/Magazine Publishers	57
Movie Studios/Publishers	17
Board Game Publishers	23
Entertainment Software/Cartridge Developers	19
Telecommunication/Cable Service Providers	22

FCP's proactive approach has enabled the Firm to clearly understand the acquisition/ investment strategy of these strategic partners/investors. The Partners have also identified the appropriate group and individuals within these large organizations that would be responsible for analyzing an investment or acquisition in the entertainment, education and multimedia software industry.

FCP has in-depth knowledge of the entertainment, education and multimedia software industry and a solid understanding of the success factors that drive the industry. The Partners have a strong base of knowledge from which to provide valuation advice, prepare written material on an acquired company, and appropriately describe the industry to potential investors.

The Partners have developed an extensive, proprietary data base of the public and private transactions completed within entertainment, education and multimedia software.

FCP has created a systematic approach for completing transactions. The Firm's step-by-step methodology, attention to detail, quality control, and exhaustive follow-up with the potential strategic partners will ensure that an acquired company will choose among the best strategic partners available.

SELL-SIDE
FCP'S UNIQUE QUALIFICATIONS

Our current edutainment and multimedia software clients acknowledge that the firm gives them a critical advantage in a number of areas:

Pricing Knowledge of Recent Private Transactions Valuation is a critical component of any transaction. Pricing multiples that existed six months ago no longer pertain in today's environment. ***FCP*** has considerable experience in researching and analyzing valuation issues for the entertainment, education and multimedia software industry. In addition to maintaining a

database on public and privately-held transactions, the Partners speak on valuations for the industry, and provide expert testimony for litigators. In the past twelve months ***FCP*** has valued eleven entertainment, education and multimedia software companies.

We do not leave money on the table for our clients.

Strategic Partnering. ***FCP*** has a bias toward arranging strategic investments for clients. Strategic partner transactions have a key advantage: the synergies created by combining the right companies create additional value for all parties. This enables clients to obtain a higher valuation for their equity.

Extensive contact network. ***FCP*** has a wide contact base with the companies investing in interactive entertainment, education and multimedia. The Partners are in

Contact with over 200 firms in diversified entertainment, cable television, multimedia hardware, media publishing, telecommunications, and motion picture development who

have expressed interest in participating in interactive multimedia. The firm's contacts enable ***FCP*** to quickly identify the appropriate partners, and best fit, leading to greater value for all sides.

Transaction Execution. In total, the Partners have completed nearly one hundred acquisitions and divestitures. This extensive transaction history ensures that our clients will be represented by highly qualified and experienced professionals. ***FCP's*** experience also helps clients avoid common transaction-related pitfalls, reduce the time to closure, and present the Company in its most favorable light, thereby increasing value for the shareholders.

Timing. While other banks may *know of* the larger would-be investors in this industry, we speak to many of them on a weekly basis. Moreover, our knowledge of industry dynamics, and ability to communicate that knowledge to investors, greatly reduces the time required to obtain investment capital.

In-depth industry knowledge. ***FCP*** understands the entertainment, education and multimedia software industry from the ground up. The Partners have helped many companies and investors define the factors which determine success in the industry, and ultimately, to achieve it.

OVERVIEW OF A STRATEGIC PARTNER INVESTMENT

Strategic partnering transactions have several advantages over other funding sources such as, venture capital or private placement:

Synergies among the companies create additional value for all parties, and thereby enable the client to obtain a higher valuation for its equity.

The strategic partner is typically available for future capital.

The strategic partner can provide ideas, content, distribution or other non-capital advantages.

When desired, the strategic partner structure can provide a secure and profitable exit strategy, at a set time period in the future.

There are a number of ways to structure a strategic partner investment:

Structure

Example

Joint development of new product.

Time Warner and Silicon Graphics joint venture to develop a digital multimedia set-top device

Minority investment with the strategic partner gaining operational leverage.

Matsushita's investment in 3DO led to their position as lead hardware manufacturer for the new venture.

Minority investment today, with option to acquire additional equity (majority control or 100%) in the future.

Addison -Wesley, a book publishing company, acquired stake in ADAM Software, Inc. The two companies also agreed to develop multimedia educational products.

Minority investment. Investor wanted to acquire access to next generation software, without making a billion dollar "MCA" acquisition.

IBM's investment in Digital Domain, a producer of state of the art computer animation and "morphing" techniques used in film.

Majority investment in order to acquire technology.

Time Warner investment in MetroComm AxS, a manufacturer of fiber-optic networks.

One hundred percent acquisition today, using an average pricing multiple, with the shareholders of the acquired company able to obtain a significant upside, at an agreed upon formula, based on future results.

Sierra On-Line's acquisition of Bright Star.A premium price paid today for one hundred percent of the equity, with strategic partner providing ideas and capital to grow the company. Shareholders remain on as employees running the acquired company with good salaries and excellent bonuses based on future results.

Viacom's acquisition of Icom.

Over the past 18 months the ***FCP*** Partners have formed relationships with potential strategic investors in the entertainment, education and multimedia software industry, both domestically and internationally. Investors include major movie distributors, book publishers, major media/magazine publishers, cable operators, telecommunication firms, diversified entertainment enterprises, personal computer

manufacturers, and other software.developers.

BUY-SIDE TRANSACTIONS
BENEFITS OF USING FROST CAPITAL PARTNERS

SCREENING PROCESS:

FCP brings several important advantages to the process of finding developers/publishers with innovative technology or a unique competitive edge.
FCP is deeply involved in the industry and the Partners have an extensive network of contacts throughout the entertainment, education and multimedia software industry. *FCP* will be able to identify little known developers at an early stage of their corporate life-cycle, and often well before other potential investors.

In-depth knowledge of the entertainment, education and multimedia software industry and a solid understanding of success factors that drive the industry. *FCP* has gained valuable insights from a broad range of industry relationships, and is able to distinguish superior products and technologies.

Extensive proprietary data base. *FCP's* large and growing data base of entertainment, education and multimedia software companies ensures that clients will have access to well-known, multi-platform developers, as well as hard-to-find, highly talented start-ups.

Systematic approach. *FCP's* step-by-step methodology, attention to detail and exhaustive follow-up with potential targets, ensures that clients will choose among the best software publishers/developers available.

Using *FCP* to identify Targets on a proactive basis will often provide clients with an early first look before a potential target becomes visible to a large number of other acquirers. This "first look" relationship gives the buyer several benefits. Among them are:

Time to look at companies without the pressure of other buyers or industry investors.

Time to position the buyer correctly with these companies, who may know less about them than about other acquirers.

Ability to get to know the personalities and assess strategic and corporate "fit".

PRICING KNOWLEDGE OF RECENT PRIVATE TRANSACTIONS

Valuation is a critical component of any transaction. Pricing multiples that existed six months ago no longer pertain in today's environment. *FCP* has considerable experience in researching and analyzing valuation issues for the entertainment software industry. In addition to maintaining a database on public and privately held transactions, the Partners speak on valuations for the industry, and provide expert testimony for litigators. In the past twelve months *FCP* has valued eleven entertainment, education and multimedia software companies.

TRANSACTION EXECUTION

In total, the Partners have completed nearly one hundred acquisitions and divestitures. This extensive transaction history ensures that clients will be represented by highly qualified and

experienced professionals. *FCP's* experience will also help clients avoid common transaction-related pitfalls, identify and resolve key issues and reduce the time to closure.

In addition to these activities, *FCP* aggressively seeks out companies at an early stage to assess their technology and approach to the market. As a result, *FCP* is often aware of innovation before the rest of the industry. Most importantly, as industry specialists, we know innovation when we see it.

PORTFOLIO COMPANIES AND STRATEGIC PARTNERS

FCP occupies a unique position in the entertainment, education and multimedia software industry. As a principal in the Avatar/HCR family of personal computer hardware and software companies, ***FCP*** has access to substantial industry resources. While most financing, strategic partnering, and bundling services are instigated at the request of clients, ***FCP's*** relationships in the industry enable the Partners to anticipate opportunities and present them to clients in advance of the market.

Avatar International Merchant Bancorp, a closely-held merchant bank with offices in San Francisco. Avatar IMB specializes in financing to personal computer hardware and software companies in California and the Southwest.

Avatar Partners, a lead investor and deal syndicator in the personal computer industry. Avatar Partners is also a financial principal and market maker in the conversion and distribution of non-performing assets.

Azteq Systems Technologies, a manufacturer and systems integrator of IBM compatible personal computers. Azteq provides bundled multimedia and productivity solutions to government, non-profit, and corporate markets.

HealthCare Resources International, one of the largest value added resellers of multimedia health care software in the U.S. The Company has representatives in most states and all major cosmopolitan areas.

HCR (of California) Inc., distributes medical and dental software, provides practice management consulting and placement of personnel to the health care community. Other services include: hardware sales, computer systems training, and computer network installation.

Ivy league Software, a wholesale distributor of IBM and Apple entertainment, education, and graphics hardware and software in the Pacific Rim and Latin American countries. Ivy league sells and finances inventory to small distributors and resellers.

GENERAL PARTNERS

Ian D. Berman, CPA, MBA

Mr. Berman has thirteen years experience in investment banking and financial advisory consulting. In 1985, after spending five years in New York investment banking, Mr. Berman was recruited from Salomon Brothers Inc., to start a Capital Markets Group in San Francisco for the international firm of Touche Ross. The Group focused on serving Pacific Rim inbound buyers and investors, as well as local investor groups. Transactions have ranged in size from a $500K private placement, to a $300M acquisition of a national retail chain. Mr. Berman currently serves on the board of three companies, including an edutainment software developer, and two philanthropic

organizations.

Brian G. Feldman, MBA
Mr. Feldman has ten years experience in the development, marketing, financing, and acquisition of computer software and related products. At Hewlett Packard he was responsible for the acquisition and marketing of thirteen personal computer applications from independent developers. Mr. Feldman has managed all aspects of the product development process at hardware and software divisions of HP, and at Xerox's Artificial Intelligence unit. Prior to joining FCP, he served for three years as a Management Consultant with Deloitte & Touche. Mr. Feldman earned a BS degree in Computer Science from MIT and an MBA from the Wharton School.

Dean L. Frost, MBA
Mr. Frost has over six years experience providing investment banking and consulting services to the personal computer industry. Prior to starting *FCP*, Mr. Frost managed his own M&A and consulting practice (founded in 1986); served as a management consultant with Deloitte & Touche; and founded and later sold ASAP, Inc., the largest marketing and promotions firm to major oil companies in the U.S. Mr. Frost earned a BA in Economics from U.C. Berkeley (magna cum laude), and an MBA from Harvard Business School. Mr. Frost currently serves on the boards of several corporations, including entertainment software developers, hardware distributors, and a computer manufacturer.

FROST CAPITAL PARTNERS

Founded in 1986, **Frost Capital Partners** (*"FCP" or the "Partners")* is a private investment banking firm with a significant practice in the entertainment, education and multimedia software industry. The Firm is one of this industry's most active investment banks in mergers, acquisitions, and private placements. *FCP* Partners are financial advisors to a *who's who* of the industry, and serve on the boards of several entertainment software and hardware clients. In short, *FCP* knows the entertainment, education and multimedia software industry.

Our clients acknowledge that the Firm gives them a critical advantage in the following areas:

Transaction Execution. In total, the Partners have completed nearly one hundred acquisitions and divestitures. This extensive transaction history ensures that our clients are represented by highly qualified and experienced professionals.

Extensive contact network. *FCP* has a wide contact base with the companies investing in entertainment, education and multimedia software. The Partners are in contact with over 300 firms in diversified entertainment, cable television, multimedia hardware, media publishing, telecommunications, motion picture development, software publishing and software distribution who have expressed interest in participating in interactive multimedia. The firm's contacts enable *FCP* to quickly identify the appropriate partners, and best fit, leading to greater value for all sides.

Pricing Knowledge of Recent Private Transactions. Valuation is a critical component of any transaction. Pricing multiples that existed six months ago no longer pertain in today's environment. *FCP* has considerable experience in researching and analyzing valuation issues for the entertainment software industry. In the past twelve months *FCP* has valued eleven entertainment, education and multimedia software companies.

Strategic Partnering. *FCP* has a bias toward arranging strategic investments for clients.

Strategic partner transactions have a key advantage: the synergies created by combining the right companies create additional value for all parties. This enables our clients to obtain a higher valuation for their company's equity.

Timing. While other banks may *know of* the current and future players in this industry, we speak to many of them on a weekly basis. Moreover, our knowledge of industry dynamics, and ability to communicate that knowledge to investors greatly reduces the time required to find the right acquirer/acquisition.

Understanding corporate culture. *FCP* understands the critical nature of the corporate environment. The Partners' transactions meet financial, strategic, and cultural criteria.

Systematic approach. F*CP's* step-by-step methodology, attention to detail, and exhaustive follow-up with potential strategic partners ensures that an acquired company will choose among the best capital sources available, and that an investor will make the right acquisition

Additional information can be obtained by contacting:

Frost Capital Partners
Four Embarcadero Center, Thirty First Floor
San Francisco, CA 94111
TEL (415)274-8899
FAX (415)274-8885

IF IT CAN GO WRONG...

(Producer's Tips on Preparing for Disaster)

By Darlene Waddington, Producer, Game Designer

Congratulations?

So you're going to produce a multimedia project. You've got a concept, you've got the money (maybe)... you even found out what multimedia is. (What *is* it, anyway?) You're prepared for anything, except, most probably, failure.

But why think about that when you're still enjoying the salad days of product development, the days of planning and anticipation? Why? Because something will go wrong. You'd better face it now.

You can't eliminate problems, but you can anticipate them and reduce your risk by using certain techniques through the various stages of development.

The Concept Stage

Most concepts are team efforts. People in sales, marketing and finance usually want a voice in what you're producing, whatever it may be. But unless you work for a software company, chances are most team members have no experience in the process of collaborating on an interactive design.

You might know very little about it, too. But since you're the producer, you need to take the lead. How? Start with a project proposal.

Project proposal

Write a one to five page document outlining your concept. If you can, sketch some rough sample screens. Even if you don't really know what you want to do, write something. Any sort of documentation brings you one step out of the totally abstract and provides a point of departure for discussion. You might just list questions and issues for discussion. This approach is often more valuable than spending a lot of time developing a concept that someone doesn't like or simply can't be done.

Brainstorming

Now, call a brainstorming meeting. Flexibility here is critical, not only for you but for others in the meeting. Consider all options and play devil's advocate. Engage in discussion and debate so each member can understand the issues and contribute effectively to the process.

Listen carefully to what the others say they want, then work to find out what they really want. When somebody requests a feature, ask them why they want it and what goal it will achieve. Often people think there's only one way to get a certain result. Good designers consider all the options.

Sometimes, certain aspects of a project are already "carved in stone." For example, you're putting your company's video library on CD ROM, or the CEO of your company wants to be the narrator of a kiosk project for a trade show. These stipulations often cause the most problems. Is the video library too large to fit on a CD ROM? Can the company owner speak effectively?

You probably can't address the CEO's problem head on, so just keep his statements short and

sweet. But with the video library problem, you should discuss changing formats or including only critical footage. Nobody likes to spend time in meetings talking about sticky details, but you owe it to everyone to let them know when their expectations are unrealistic. Unfortunately with technical issues, you might not be aware a problem exists until you talk to the contractors. But don't worry, you can still catch this in time in...

The Design Stage

Now that you understand everyone's needs and everyone understands the issues that may impact the final product, it's time to work out the details.

Revised project proposal

Write a walk-though of the interactive experience and make a flowchart to track the branches. Also, create a rough story board. Usually, this process exposes flaws in the concept and raises new questions. If you can't solve these problems, contact the others on the design team and work them out.

Some issues may be hard to resolve because you don't know what you can really afford to do. It's easy to underestimate the time and cost necessary to produce interactive projects, especially when you're first starting out. Contact production studios, artists, programmers, actors, etc. as soon as possible to find out what's realistic.

Try to get recommendations on contractors, but don't rely on second-hand information. The best way to evaluate them contractors is by seeing their work at their workplace. If everything looks good, ask for bids and schedules. If your design is still so vague that firm bids can't be made, at least get ball park figures. Also, get some demo material to show the approval committee. Taste is a *very* subjective thing.

Brainstorming meeting #2

Bring the bids, schedules and demo material along with your design walk-through, flowchart and story boards to another meeting. Get a consensus on the design and the contractors. Now everyone's sharing responsibility for the project... for better or worse.

The Development Stage

By the time you've reached this stage, your design has probably evolved somewhat from the original concept. So now there will be no need for more changes, right? Wrong!

No matter how well you conceptualize on paper, you cannot truly evaluate an interactive product until you interact with it. A video segment might be too long; another segment goes by too fast; the audio is distorted; the programming's taking too long; the art looks crummy; some menu options were overlooked; the interface is too complicated... the possibilities are limitless. And even if you've created a bullet-proof design and schedule, you can't rule out the human factor. Someone comes down with the flu, and suddenly you're behind schedule. And if the human factor doesn't get you, don't forget the hardware factor. Computers have a funny way of acting up when burdened with do-or-die deadlines.

Smart scheduling

The best way to recover from problems is by creating safety nets. You should never create a schedule with no margin for error or revisions, but it seems like most projects are rush projects. If you're stuck in this bind, scale back on the project. A simple program that works great is much better than ambitious one that falls short. You need "slop time" in your schedule for dealing with unforeseen problems. This isn't cheating. This is reality.

How much slop time should you add? Ten to twenty percent of the total development time is a good average, but it depends on the project. In general, give yourself as much cushion as you possibly can. You'll never regret it.

Also, structure your schedule so that the trickiest and riskiest features are tackled first. This way, you have time to recover and create a new plan of attack.

Early review meetings

As soon as you have something to show, show it. In the early stages, you might just invite individuals to review certain parts of the project. But get the group together before you're halfway through---groups tend to form different opinions than people in one-on-one situations. Ultimately, the group will evaluate the project, so it's best to get its reaction early on.

It's also useful to show the project to people who aren't part of the team. They often provide the truest reflection of how your real audience will react. If possible, do some informal (or formal) focus group testing. Again, do this as early as possible so you can respond to problems.

Play testing

Play test your program throughout development as much as possible, then test it *ad nauseam* at the end. New bugs sometimes appear in debugged parts of a program when other features are implemented, so test everything again after making any changes.

The people closest to a project are most apt to overlook obvious bugs such as typos. This is especially embarrassing if the typo is something easy to take for granted like the company's name or product, so before going to final, find testers with fresh eyes and have them question everything. Try to break it, because if you don't, somebody else will.

The Aftermath

After the project is done, reassemble the team and review the experience. Is everyone happy with the final product? Would they do it differently next time? What did you learn?

Ask these questions so that next time your job will be easier. But remember... it's never easy!

Run the hilbers.avi file from the embedded Viewer AVI control.

{ewl MVMCI2, ViewerMCI, [device AVIVideo][stdcontrol]samp_aps.avi}

End.

InterMedia Interactive Software

Here are two animation's that we have created for use in a CD-ROM title we are pursuing.

These animation's were designed on the Amiga as 3D wireframe models.
The appropriate light sources and textures were then added in. Each frame was rendered as a 24-bit frame and then taken down to 8-bits. These 8-bit images were brought over to the PC, where the identity palette was inserted using BITEDIT.EXE.
The FLC file was assembled using Autodesk 2D studio.

We also use Autodesk's 3D Studio Pro on the PC to design animation's as well.
The choice of platforms is simply dependent on the animator's background.

Run the Faucet Animation
Run the Toggle Animation

Jasmine Multimedia

Read this before using:

1. This states the terms of a legal agreement between you and Jasmine.
2. You accept the terms of this agreement, and enter into this agreement, by opening the sealed disc.
3. If the terms of this agreement are not acceptable to you, or you refuse to be bound by the terms of this agreement, you should promptly return the software package unopened along with all written materials to the place of purchase.

LICENSE

Jasmine grants you the non-exclusive right to use the software subject to the following restrictions:

1. You may use the software on a single CPU on which the software is first used only.

2. You may make one copy of the software in any form for back-up purposes only, and not for use as, or a variation of, a stock music, video, or still-image library.

3. You may not sell, lease, rent assign,, or transfer the software without the prior written consent of Jasmine. You may sell or transfer the software to another party on the condition that the other party executes a written enforceable, sub-license agreement containing the terms and conditions of this agreement as well as an additional provision prohibiting any further sub-licensing or transfer. If you transfer the software, you must at the same time either transfer all copies, whether in printed or machine readable form, to the same party or destroy any copy not transferred.

4. With the exception of paragraphs 5 and 6 below, you may not reproduce and/or repackage any part of the software, including the music, video, or still-image libraries, for broadcast, retail, or theatrical use in any media, photographic or silverbased or otherwise, including without limitation cablecast, satellite or microwave transmission or videotape, without the prior consent of Jasmine.

5. You may make use of the video and still-image libraries in magnetic and/or optical form for retail use only.

6. You may make use of the software as part of a project where the software is not the predominant part of the project.

7. You may not make any use of the software which in any way may be considered defamatory or a defamation of any individual.

8. You may not reverse engineer, decompile, or dessemble the software.

9. Any portion of the software an/or music, video, and still-image libraries used in any program or project will continue to be subject to the terms an conditions of this agreement.

COPYRIGHT

The software and its music, video, and still-image libraries are owned by Jasmine and/or its suppliers and are protected by the United States copyright laws and international treaty provisions. You must treat the software like any other copyrighted material. Any violation of this license is a violation of the agreement and of the United States copyright laws.

THIRD PARTY BENEFICIARY

You are hereby notified that there are third party beneficiaries to this agreement to the extent that this agreement contains provisions that relate to your use of the software. Such provisions are made expressly for the benefit of said third parties, and are enforceable by then in addition to Jasmine.

LIMITED WARRANTY

1. Jasmine warrants:

a. That the software will perform substantially in accordance with the accompanying materials for a period of 90 days from the date of delivery of the software as evidenced by your receipt;

b. The workmanship under normal use of any hardware accompanying the software for a period of 90 days from the date of delivery as evidenced by your receipt.

2. In no event will Jasmine be liable to you for any damages, including any lost profits, lost savings, or other incidental or consequential damages arising out of the use of or inability to use the software.

3. This agreement will be governed by the laws of the State of California.
   The software is provided "as is" without any other warranty, either express of implied, including warranties of fitness for a particular purpose or merchant ability. Jasmine does not warrant that the functions contained in the software will meet your requirements or that the operation of the program will be uninterrupted or error free. The entire risk of the software is with you. Some states do not allow the limitation of implied warranties, so the above exclusions may not
   apply to you.

Dear Multimedia Enthusiast,

Jasmine Multimedia Publishing has spent the last five years working with the world's largest libraries of video, music, and stills so you won't have to worry about costly copyright clearances. Now you can take advantage of the finest professionally shot video, fully orchestrated music and stunning photographs from around the globe for a fraction of the cost of traditional stock sources. Each of our CD titles is packed with clips that can be easily edited into any computer file, document, training program, game or software product. Best of all, Jasmine's CD-ROMs of clip media can be used royalty free in any computer application you create. Our goal is to take the hassles out of multimedia production. If there is a way we can be of more service, please let us know.

Sincerely,
Jay Alan Samit
President
Jasmine Multimedia Publishing

Jasmine AVI Clips

Run the AVI files from the Viewer embedded AVI control.

{ewl MVMCI2, ViewerMCI, [device AVIVideo][stdcontrol]samp_aps.avi}
{ewc MVMCI2, ViewerMCI, [device AVIVideo][stdcontrol]samp_aps.avi}

{ewl MVMCI2, ViewerMCI, [device AVIVideo][stdcontrol]samp_aps.avi}

{ewc MVMCI2, ViewerMCI, [device AVIVideo][stdcontrol]samp_aps.avi}

Run the AVI file from the Multimedia Player.

End.

Mondo Media Clients

Microsoft Corporation: Mondo Media created animated sequences with audio for Encarta - a multimedia encyclopedia on CD-ROM. Animation's included Theory of Relativity, How an Airplane Works and Genetics. Mondo Media also designed and produced Video for Windows and Windows NT Autodemos on CD-ROM for distribution to Microsoft dealers.

Compaq Computer Corporation: Mondo Media created autodemos for the successful, low cost Prolinea and Contura lines. We also produced an animated, interactive demo to launch the COMPAQ Portable 486/33c. Mondo Media also designed and produced the Prodigy campaign for Compaq's PC lines.

Prodigy Services Company: Mondo Media is the largest external developer of content for Prodigy - the on-line interactive consumer service. Mondo Media has developed interactive campaigns for Hewlett Packard, Compaq, Intuit, Encyclopedia Britannica, IBM and others.

Hewlett Packard: Mondo Media designs and produces Hewlett Packard multimedia presentations and videos. Mondo Media also created the Prodigy campaign for HP New Wave.

Autodesk, Inc.: Mondo Media has created animated, multimedia presentations for Autodesk. We are also an active beta tester for all Autodesk multimedia products.

Intel: Mondo Media designed and produced an Indeo Video Demo to demonstrate Intel's new Video for Windows technology. We also designed an animated logo to use as a signature for ads and promotions.

Adobe: In process of creating a user demo for the new version of Photoshop for Windows.

The Clorox Company: Mondo Media provides an assortment of multimedia services to the Clorox sales and marketing divisions. Projects include point-of-sale demos for mobile sales force laptops and corporate multimedia presentations.

MECHEDAUS, the newly formed interactive entertainment company, has developed its first title for the MPC platform for the Christmas season.

Critical Path is the world's first interactive action-adventure CD-ROM to combine motion picture, video game and computer-generated animation elements in a single interface. The breakthrough title, which also features an original soundtrack and extensive special effects as part of the interaction, was developed for Media Vision.

Run FLC Clip1
Run FLC Clip2
Run FLC Clip3

Pixar Flying Logo Movie

Animator: Bill Kolomyjec, RenderMan Marketing Manager, Pixar
Software: Pixar's Typestry and Microsoft's Video for Windows

The Pixar Flying logo movie was created with Pixar Typestry using the keyframe animation feature built into the software. The story, objects and pencil tests for the animation were created, designed and produced over a weekend. Four separate Typestry project files were necessary because we wanted the Pixar logo to be shown in a variety of fonts and colors. The total running time of the animation is approximately 30 seconds (465 frames at a playback rate of 15 frames per second.) The output from the four separate Typestry project files were combined using VidEdit which is part of Microsoft's Video for Windows. Embedded into Pixar Typestry is Pixar's PhotoRealistic RenderMan technology, a world class rendering tool that automatically produces motion blur in each frame making the motion look so good.

Pixar
1001 W. Cutting Blvd.
Richmond, CA 94804
USA
510-236-4000

Example video files from Pixar.

{ewl MVMCI2, ViewerMCI, [device AVIVideo][stdcontrol]samp_aps.avi}

{ewc MVMCI2, ViewerMCI, [device AVIVideo][stdcontrol]samp_aps.avi}

End.

Personal Media International
Millennium Auction

MDK Demonstration Disk Readme

*****THIS DEMO WILL NOT RUN PROPERLY*****
FROM A CDROM DRIVE
PLEASE COPY ALL FILES TO A HARD DISK
ALL DEMO FILES MUST BE DELETED FROM YOUR
HARD DRIVE AFTER VIEWING

*****THIS DEMO REQUIRES THE FOLLOWING*****
SYSTEM CONFIGURATION

- Windows 3.1
- Super VGA mode (i.e. 640 x 480 x 256)
- 8-bit sound card and stereo speakers/headphones
- MCI capabilities. Minimum required MCI control
  panel drivers: [MCI] Autodesk Animation Player:>Version1.1
  [MCI] MIDI Sequencer
- MIDI card supporting General MIDI

NOTE: The demo is designed to run with a rich-voiced midi card such as the Turtle Beach Multisound or the Roland Sound Canvas. Tracks may be dropped on low end cards depending on the midi-map configuration (see section 4 below).

TO INSTALL & RUN THIS DEMO:

1) Create a directory on your hard disk for the demo files. Copy all the files from the CD to this demo directory. There are approximately 30MB of files that must be transferred.

2) Start up windows 3.1.(Make sure you are in super VGA mode)

3) From the program manager select File, Run.

4) Type the full pathname for

   a) BIDDERS.EXE
   b) ZEKE.EXE
   c) NURIA.EXE
   d) GONE.EXE

   for example "c:\PMIDEMO\ZEKE.EXE"

5) Click on the left-most triangular "PLAY" icon to start the demo.

NOTE: At the end of the presentation you will have to click on the right-most "EXIT" icon to return to Windows.

6) Sit back and enjoy the presentation.

If you have any problems please consult the next section.

SOME PROBLEMS AND SOLUTIONS:

This section describes possible solutions to the following problems:

1) Animations do not run
2) No Audio is heard and the presentation continues
3) No Audio is heard and the presentation stops
4) No MIDI is heard on the Millennium Auction title screen
5) Any other messages

1) Animations do not run and you get the message:
"Animations will not run please check the readme file"

If this happens you probably do not have the required MCI animation drivers loaded. To load the drivers follow these steps:

a) From the program manager select File Run and type:
c:\windows\system\sysedit.exe and hit enter
(This of course assumes that Windows is in the c:\windows directory).

b) Select the system.ini file.

c) Find the '[MCI]' section (You can use *Search Find '[MCI]'* Enter)

d) At the end of the section add the following line
Animation1=mciaap.drv

e) Select the win.ini file

f) Find the '[MCI]' section (You can use *Search Find '[MCI]'* Enter)

g) At the end of the section add the following line
flc=Animation1

h) Now exit sysedit, and save both the system.ini and win.ini files.

i) Next copy the files mciaap.drv and aaplay.DLL from the CDROM to your c:\windows\system directory (assuming that windows is in the c:\windows directory).

2) No Audio is heard and the presentations continues

Check the volume control using the mixer software supplied with your audio card.
NOTE: The snippets in this demo contain *MIDI* audio (background music) and *Wave* audio (speech)

3) No Audio is heard and the presentations stops

a) Check to make sure that all the files from the demo CD are in your demo directory.

4) No MIDI is heard at the Millennium Auction title animation

a) Check that you have a high end MIDI card.

b) Make sure that all 16 midi channels are available to the midi driver (if possible).

To do this:
Select the MIDI Mapper from the Control Panel.
Select Setups.
Click on Edit.
Select the available MIDI ports (make sure that the first 16 are active).
Restart windows and see if there is any improvement.

5) Any other messages or problems

If you encounter any other messages or problems please call Vatche Kalaidjian at (718) 884 7095

End.

R/GA Digital Studios

Mission Statement.

R/GA Digital Studios is an integrated network of creative studios that, because of its distinctive competence in the areas of live action, digital production and interactive multimedia (IM), can collaborate with clients as a creative and technical resource, enabling them to realize and maximize their creative potential.

We challenge the traditional preconceptions about how specific mediums should be used, and work across all media without being limited by technological constraints. Our only constraint is the power of imagination.

Integrated Studio Network.

The following independent, but fully integrated, studios make up the R/GA Digital Studios network:

- R/Greenberg Associates-Complete digital production including Special EFX, Digital Post Production, Computer Generated Imagery and Graphic Design for feature films, TV production and commercials.
- R/GA LA- A mirror image of R/Greenberg Assoc. servicing the West coast and the Special Venue/Theme Park market.
- R/GA Print- Offering the same digital post production capabilities, but for the higher resolution print medium.
- R/GA Interactive- Conceives, creates and produces Interactive Multimedia programming for the advertising and entertainment markets.
- R/GA Pictures- Developers and producers of effects driven feature films.
- Michael Schrom Associates- Creative solutions combined with high quality cinematography for TV commercial production.
- Savoy Commercials- Live action TV commercial production.

Mind's Eye Graphics, Inc.

The Company

Mind's Eye Graphics, Inc. (MEG) was founded by Mark Lambert in September of 1988 in Washington DC to develop computer graphics software and provide video and print services. On July 1, 1990, Mind's Eye Graphics in Richmond was formed. This studio offers direct online services for video paint and animation. May of 1992 brought the opening of our new headquarters in historic Shockoe Slip to provide electronic prepress and image retouching capabilities. Mind's Eye Graphics runs on 4 Silicon Graphics 4D series workstations as well as IBM and Macintosh computers to offer a wide range of graphics capabilities. Mind's Eye can now provide the regional market with computer artwork, photographic retouching and 3-dimensional animation services of the highest quality available anywhere.

Mark Lambert is president of Mind's Eye Graphics, Inc. and is one of the world's foremost experts in computer graphics. Mr. Lambert has a resume including the opening for CBS News Nightwatch, White House graphics for President Reagan's State of the Union addresses and national spots for clients such as General Motors and Time Life. He was the first Wavefront 3D animator in the Mid-Atlantic region and is experienced in a wide range of animation techniques and software including Wavefront, Alias, and Pixar's Renderman. As a result, he has often been called in by other animation companies to aid in the more difficult effects work for national commercial and feature films. He has also worked in the industry as an engineer and video editor, and has a background in all phases of video and special effects production. Mr. Lambert is also considered one of the leading animation software developers in the nation, with Mind's Eye Graphics software being used by CBS, Boss Films, CBN, Editel, Limelite and may other national and international studios. Mind's Eye latest software, WaveMan, is currently allowing some of these companies to produce groundbreaking effects for feature films, entertainment park rides, show opens and national television spots. MEG software is also incorporated directly into the Wavefront Alias and other leading computer animation packages.

3D Animation and Graphics

MEG is one of the nation's most advanced 3D studios. The company has built a hybrid collection of many different systems to offer an incredible versatility in modeling, motion and rendering capabilities. Wavefront's Advanced Visualizer software provides the primary 3D animation capabilities and is available on two of the four Silicon Graphics workstations. Wavefront is currently considered the leader in 3D animation software and is the most popular high end graphics system nationwide. The Alias system may also be used for it's superb modeling of curved surfaces. For the most demanding of projects, Pixar's Renderman software (used in such movies as Terminator 2) can be used to provide such features as depth of field and motion blur to an animation or graphic. MEG developed the interface between Wavefront and RenderMan and is currently marketing this technology for use in feature film and high end commercial animation. Together, these systems can generate 3D animations of photo realistic quality for show opens commercial tags, architectural fly-throughs, instructional graphics and print. The addition of a morphing package allows MEG to bend and deform a single raster image as well as transform one image into another using single or multiple frames.

The company also draws upon the wide range of software products that Mind's Eye Software has developed, and often writes custom software to accomplish effects unavailable at any other area facility. Not to be overlooked in all this incredible technology, the company has a highly creative staff of artists with a wide range of experience. The company works with clients based on their design needs, from simple suggestions and brainstorming up to complete design of logos and animations. with the powerful combinations of creativity, experience, and technology, the company can create almost any vision of the Mind's Eye.

Print and Prepress

The hardware, software and talent at MEG has combined to form one of the most advanced print special effects studios in the nation. Using packages such as Alias Full Color and Wavefront's Advanced Paint systems running on Silicon Graphics high-end workstations allow our artists to rapidly create a wide variety of images and effects unavailable at any other studio or printer in the region. These systems are capable of a full range of techniques, from high resolution photo retouch and image compositing to the complete creation of logos, airbrushed artwork, and technical illustrations for print. We have even added software to perform high quality image deformation (bending) for some of the hottest new looks around. With the variety of packages and the addition of in-house software development, the studio is geared toward creating the most advanced special effects currently possible on digital systems. Additionally, images such as the "flying flags" on our full color flyer can be created by the company's 3D systems (Wavefront, Alias and RenderMan) and then digitally transferred into the paint system. This allows the paint artist to utilize the strengths of 3D to rapidly create perspective views and complex scenes without the typical trial and error of paint-only systems. The artist even added true "motion blur" to the flag image in the 3D system rather than try to "smear" it in.

We have also completely created logos and images on the 3D systems for styles unlike current 2D techniques. CAD files can also be translated for the creation of such things as architectural and product visualizations. Macintosh and PC systems are directly networked with the Silicon Graphics workstations providing additional flexibility in image creation as well as offering hundreds of Postscript font selections. Most major Macintosh programs are supported to allow clients to send line art and type for incorporation into final pieces. Images may be input from a variety of sources including high-end prepress systems such as SCITEX and HELL scanners. Output can be to slide, transparency, separations or digitally transferred back to a SCITEX, HELL, etc. at a resolution of up to 8000 lines. We can also output direct digital to high quality poster size prints, at sizes up to 32'x46', saving the client costs by eliminating the need to generate intermediate transparencies.

To Contact:

Mind's Eye Graphics, Inc.
115 Shockoe Slip
Richmond, VA 23219

Tel. 804 643 0441
Fax 804 643 3738

WalkSoft Corporation

Run Demo

News In Motion

News in Motion is trademark of WalkSoft Corporation.

Contents

About this version of News In Motion

This version of News In Motion is included on the Microsoft Developer's CD-Rom and is intended for demonstration purposes only. The software and its component parts may not be redistributed without the express authorization of Walksoft Corp. All articles, photos, and animations are the property of their rights holders. They may not be redistributed without the express authorization of the copyright holders.

News In Motion Installation and Release Notes

1. **What you need to run News In Motion**

- 80386sx or better computer
- DOS 5.0 or 6.0.
- Windows 3.1.

- Four or more megabytes of memory.
- A hard drive with about six megabytes of free storage.
- VGA graphics display.

How to install the Newsstand software

1. At the DOS prompt (C:\>), type: win<Enter>

2. You should now be in windows, running the program manager. Select the Run command from the File menu. This displays the Run dialog box.

3. Place Installation Disk # 1 in your 3 1/2 diskette drive.

4. In the Command Line field type: a:install<Enter> or b:install<Enter>depending on which drive you placed the installation disk.

5. The News In Motion Install window will now appear. C:\NIM is the default directory to install the software. You may change the installation drive or path if you wish. Select Continue to go on.

6. Dialog boxes will appear which will ask you to confirm three directory names. It is important that you do not change any of these names. Select OK for all three times.

7. The installation program will now copy files to your hard drive.

8. You will be prompted to insert Installation Disk # 2. Insert this disk and select OK.

9. Installation will continue copying files. When this is complete, you may view the readme.txt file.

10. You will now get the message Installation is complete. Select OK.

11. The installation program should have created a News In Motion group in the Program Manager. In it, there should be an item with Sparky, the electronic newspaper carrier, entitled News In Motion. If for any reason this was not created, you may manually create the Program Manager icon with:

Description:	News In Motion
Command Line:	C:\NIM\TBOOK.EXE C:\NIM\NIM.TBK
Working Directory:	C:\NIM
Icon:	C:\NIM\SPARKY.ICO

12. Double-click on Sparky to start News In Motion.

How to install your first issue

1. If you have a VGA, 16 color only display system, place the Black and White Issue disk in your diskette drive, otherwise use the Color Issue disk. (Both are on the same disk.)

2. From the File menu, select Read Issue From Disk.

3. Make sure that the matching file type, Color NIM Files or Black & White NIM Files is selected in the List File Type Field.

4. Select the appropriate zip file from the File Name or field, such as nm930731.zip.

5. Select OK.

6. News In Motion will now copy the file to the Papers directory, and will unzip the files.

7. Press <Enter> to continue.

8. News In Motion will now Initialize and Index the issue.

9. When this process is complete, News In Motion will display the cover for the issue.

How to sign on

1. Find the subscription information sheet that came with the News In Motion package.

2. Select Settings from the Delivery menu. Enter your user id, password, and the telephone number of the server from the subscription information sheet.

3. If you are using a grayscale monitor, or a 16 color VGA display,select the Black and White button.

4. Click on Advanced button to verify your communications settings.

5. The settings will have been set automatically. You can override them, if you believe they are incorrect. Use the Auto Config button to have News In Motion auto-detect the settings. The Test Modem button will check to make sure that a modem responds using the current settings.

6. Since News In Motion is only available weekly(for now), make sure that the Weekly button is selected.

7. The Modem Initialization String field will display the default string. This should work with most popular modems. If your modem does not connect properly with the Newsstand Server, you may need to change this string. Consult your modem manual for the proper settings.

8. After you are done, click on OK to save your settings.

How to download the current issue.

1. Click the Current Issue button on the Newsstand page, or use the Current Issue menu item in the File menu.

2. News In Motion will confirm the issue to download. Click on the Download button.

3. News In Motion will download the issue. Your subscription will be decremented by one issue.

How to download a back issue.

1. To download any issue, click the Past Issues button on the Newsstand page, or use the Open menu item in the File menu.

2. Click on Download button.

3. News In Motion will request the issue to download. Type in the correct date using mm/dd/yy format(such as 8/7/93).

4. Click on OK.

5. You must next confirm the download by clicking on the Download button.

6. After the download is complete, your subscription will be decremented by one issue.

Menu item descriptions

File Menu

1. Open Issue.	Displays a list of all issues on your computer.
2. Current Issue.	Opens the current issue, or displays a dialog box from which you can download the current issue.
3. Read Issue From Disk.	Copy and create an issue from a floppy disk.
4. Print Article.	Prints the selected article (the article with the blue headline).
5. Print Setup.	Brings up the Windows Printer Setup Dialog.
6. Quit.	Exits News In Motion.

Edit Menu

1. Copy Article.	Copy text of article to the clipboard.
2. Copy Picture.	Copies the current picture to the clipboard (as a device independent bitmap).
3. Scroll Bars.	Toggles between News In Motion or Windows scroll bars for preferred viewing.
4. Font.	Allows the reader to select the font & point size which he/she is most comfortable with to display the text of articles.

Sections Menu Displays News In Motions chapters.

Delivery Menu

1. Newsstand.	Switches to the Newsstand page.
2. Subscription Info.	Displays News In Motion subscription information.
3. Settings.	Displays user setup and communications parameters.

Help Menu

1. About NIM.	Displays version information.
2. Copyright Info.	Displays copyright information about News In Motion.

<u>Notes</u>

1. You need to have a Windows 3.1 compatible sound system installed, if not, you won't hear the sounds included with News In Motion.

2. While News In Motion will run with only four megabytes of memory, it will run faster with more. If you have only four megabytes, you may wish to consider minimizing the amount of memory used by Smartdrv.exe. Smartdrv will work almost as fast with 256K of memory, as it will with one megabyte.

3. We have found that the color pictures supplied with each issue will appear wonderfully with full color display systems. They look very good with 256 color displays(if the palette is not frozen, they will look even better). Full color photos displayed in 16 color mode are less than optimum. For this reason, we have included two diskettes, one with a full color issue, and the other with a black and white issue.
   Please use the Black and White issue with a VGA-16 color system.

4. Occasionally QuickTime/Windows will neither play movies,nor show pictures on some displays. Please call us at 1-800-424-8250 if this occurs on your system.

5. Clicking on animations will start or stop them. Any time the cursor changes to a movie camera, it means that you can click on the image to start and stop the animation. This also applies to animations of Sparky, your electronic newspaper carrier.

6. Clicking on picture thumbnails at the top of articles will enlarge them, clicking on the enlargements will reduce them back to thumbnails.

7. You must enter the User ID, Password, and Telephone number, which will have been supplied on a separate sheet, before you may download any issues. If you have requested only a single issue, you will not have received this information. Please call us and sign up for a subscription.

8. The selected article's headline is shown in blue. You can use the up/down/page up/page down direction keys to scroll this article.You use the mouse or the tab key to switch between articles.

If you have any problems Please call us at 1-800-424-8250.

2. Walksoft and News In Motion

Walksoft was founded in May, 1992 to develop computer applications that manage or deliver information in unique ways.

Walksoft's first title is News In Motion, the world's first multimedia newspaper. News In Motion is based on the Newsstand software. The Newsstand is a publisher's tool that allows editors to arrange text, pictures, and movies in a variety of page layouts. There are over thirty standard layouts in the Newsstand, as well as other layouts that are customized for News In Motion.

The Newsstand software also allows users to download issues with a single command from the Walksoft Server. Other servers can also be supported. A typical issue of News In Motion contains 60 articles, ten pictures, and four animations. Each issue can be downloaded in approximately seven minutes at 14,400 bps, and ten minutes at 9600 bps. The Newsstand software can also open issues distributed on diskette or cd-rom.

Walksoft is licensing the Newsstand software to other information publishers. Typical uses of the Newsstand is for newspapers and magazines, as well as for multimedia reference material.

Please call Todd Chronis at Walksoft for further information at 1-800-424-8250.

3. Todd Chronis, President of Walksoft and Publisher of News In Motion

Todd Chronis received his B.S. in Physics at M.I.T. in 1980. He started his career working as a software application designer for various large corporations. In 1990 he founded his first company, Pendragon Technology, Inc., to design and market a computer notebook application. He is currently president of Walksoft Corp., a multimedia publisher. Walksoft's first title is News In Motion, a multimedia newspaper featuring color photos and animation. He decided to create and publish News In Motion because he could not get The New York Times delivered to his home while it still contained current news.

4. News by(tes) computer

Joshua Quittner and Lou Dolinar
Newsday

Readership is falling nationally and the recession is ripping up ad revenues. What's a newspaper or magazine to do?

Many of them are experimenting with the future, seeking new sources of income in CD-based publications, online news packages, audiotext and other services that someday may tailor news reports to individual subscribers.

Unless newspapers remake themselves electronically, many in the business fear, new online services will grab the classified advertising that is a major source of income.

"People in all forms of media are wringing their hands, trying to figure out where to go and what to do," said Roger Fidler, director of Knight Ridder's Information Design Laboratory, in Boulder, Colo. "For the first time, there's a serious threat to the printing press. No one knows where all these new technologies are leading us."

Fidler's design group has come up with a "portable information appliance," a prototype of what he believes will be the newspaper of the future. An 8 1/2-inch-by-11-inch computer screen, the device can display a perfect reproduction of a newspaper front page. By touching the screen, readers "open" pages and read the paper, laid out in familiar columns. Or the device can read aloud to them as they dress or
drive to work.

Video snippets, photographs and graphics supplement the text. The device could be attached to a network and get news feeds daily, or even hourly. That kind of "flat panel" paper is probably at least a decade away. But other new formats are at the electronic doorstep.

At the end of the month, in Rochester, N.Y., for instance, News In Motion, a weekly multimedia magazine, will start publishing news, opinion and photos from a half-dozen international news organizations, in addition to original video cartoons. For $250 a year, subscribers can call an 800 number and download each issue onto their computers.

According to Todd Chronis, who created the multimedia paper, news from the Washington Post, Britain's Financial Times, France's Le Monde, Japan's Asahi Shimbun, Germany's Der Spiegel and others will be included."I was astonished I could not already get something like this

online," he said.

Chronis said that he expects 12,000 subscribers in the first year. He plans to offer News In Motion daily before the end of the year. A Macintosh version is also expected in time for January's MacExpo in San Francisco.

"All the attention is being focused on the Time Warners," said Donald McNamara of New York City-based Liberty Fund, a consulting firm. "But the real innovation is going to come from people like these guys in Rochester."

5. What is 'News In Motion'?

This space is normally used to publish letters to the editor but since this is the first issue I would like to tell you about News In Motion.

News In Motion is the world's international electronic multimedia newspaper. That's a mouthful, so let's break it down and see what it means.

International -- That means we cover the world. We have coverage from the leading media sources from around the world, including Asahi Shimbun, Le Monde, Financial Times, L'Express, Der Spiegel, Reuters, Agence France-Presse, International Herald Tribune and lots more. We also do a great job covering the U.S., with sources including The Washington Post, the Los Angeles Times and Newsday, among others

Electronic --Iit's delivered to your computer, and you view it on your computer. The computer is the world's best tool for managing information. It's a natural for sorting and viewing a newspaper.

Multimedia -- We exploit the capability of today's personal computers to display full-color photographs and animations. Our dramatic color news photos are going to make it hard for you to turn back to those blurry, black and white newspaper pictures. When you see and hear Reflections, our featured comic series, you'll agree that it is a lot better than a few still frames of a comic strip.

We believe that News In Motion is the first and best of a new breed of news media. Here's what makes it so:

- We can give you in-depth articles. While others cut articles to a few paragraphs, we give you the full story.
- Where others will all be giving you the same view on the world, we will give you a diversity of views, from the best American and International sources.
- You can get News In Motion anywhere, anytime, in just the ten minutes it takes to download.
- We will give you full-color news photos. See the world in color. No other computer newspaper gives you photos. No one else gives you great animations. We have a staff of animators that are defining the world of news animation.
- It's surprisingly affordable. If you subscribe before November 1, you can get fourteen months of News In Motion for only about $4 an issue.
- It's convenient. The computer is the world's best information management tool. Now you can keep your newspaper right on your computer. No more clipping articles or recipes.

- It's got lots of great features, like a food column, an animated horoscope, album reviews (with sound!), and Catherwood (our favorite paranoid).

- Last, but not least, we've even got The New York Times crossword puzzle. We think it's no contest. Whether you're looking for great news coverage of the nation and the world, a full-color newspaper that doesn't insult your intelligence, or the truly unique animations and special features made possible by computer technology, News In Motion is right for you.

End.

Graphix Zone

CORPORATE FACT SHEET

BUSINESS — The Graphix Zone is a solutions-oriented organization dedicated to the education and marketing of advanced computer graphics systems, specializing in those employed in multimedia, prepress, presentation graphics and desktop graphics applications.

MARKET — The growing fields of Multimedia and PrePress are the hottest areas in computing. Multimedia offers users the power to integrate sound, graphics, text, video and vast information resources into a single medium, enhancing the personal computer and bringing together all of the most exciting aspects of the information age. PrePress brings power and control of color publishing to desktop computer users. The Graphix Zone positions itself to educate and inform the market on the opportunities available with these technologies, to the benefit of users, manufacturers, and dealers of these products.

MISSION — The Graphix Zone was created for the desktop graphics product manufacturers and end-users. The company provides the end-user and reseller with free educational seminars,demonstrations, customer support programs and individual planning sessions all designed to assist and direct the customer to his short and long range planning needs. With over 1,200 visitors to the company's resource canter each month, The Graphix company tracks these trends and makes them available to the media.

COMPANY — The Graphix Zone is a single source for graphics information and is comprised of five major service groups. Its Graphix Resource Center conducts ongoing free seminars and workshops aimed at end-users and designed to demonstrate the implementation and development of real-world multimedia, prepress, presentation graphics and desktop graphics applications. It is also designed to help vendors and resellers market integrated solutions. The Multimedia Production Group works with customers across the country to create--from concept to finished product--simple or complex multimedia communications projects such as animated logos and interactive graphical presentations. Creator Center is a do-it-yourself Multimedia Production Center where customers can rent the facilities, equipment, and talent needed to create simple to complex Multimedia presentations and projects at an affordable price. The company also offers Training and Support Services to end-users and resellers requiring more knowledge on how to use or sell multimedia solutions.

MANAGEMENT — The Graphix Zone's 7,500 square foot facility is based in Irvine, Calif. and was founded in 1989 by Chuck Cortright and Angela Aber who have a combined 30 years experience in advanced technologies. Both have held key positions with a number of high technology companies and are dedicated to expanding the Graphix Zone concept to cities throughout the United States.

CONTACTS — Angela Aber/Chuck Cortright: 714-833-3838 Beverly Lages, Lages & Associates: 714-756-6506

Center For Creative Imaging

Contents:

ABOUT THE CCI

The Center for Creative Imaging first opened its doors in May, 1991. Since then it has welcomed over 3000 students from around the globe. Artists, professionals, and novices alike have had their creativity inspired by a kaleidoscopic selection of courses in multimedia and digital imaging. Initially a Macintosh computer training facility, the Center has recently added PC and Silicon Graphics labs, as well as new curriculum for traditional video and photography programs.

The addition of our new PC and SGI labs heralds our commitment to providing the highest quality creative and technical instruction to all computer users. Our course offerings, address introductory to advanced levels of instruction on a variety of topics related to digital imaging and multi-media. Our new labs have been configured to offer the maximum amount of productivity through intensive, hands-on training. Courses range from two-day sessions on how to purchase and implement cutting edge computer graphics technology to five-day classes on advanced 3-D animation techniques. Other courses address topics from entry level approaches to desktop publishing to advanced uses of multimedia applications. Our instructors are flown in from around the world, allowing students the opportunity to learn from those who represent the latest in technical and creative achievement in their respective fields. All students are given twenty-four hour access (Monday through Friday) to their own state-of-the-art computer graphics workstation for the duration of the course.

Camden Nestled on the rugged Maine coast where the mountains meet the sea, the 200-year-old town has retained the charm and beauty of its history as a shipbuilding village even as it serves as host to thousands of visitors who have made it one of the most popular spots in the Northeast for sightseeing, sailing and hiking. With a population of just 5000, this is a true small town, complete with church steeples, village greens, and the sound of lobster boats entering and leaving the harbor. Camden is home to one of the worlds most famous windjammer fleets, with cruises available on a variety of vessels. You can spend a few hours viewing the breath-taking panorama of the Camden Hills, or take a week to explore Penobscot Bay and the islands. Camden is also a magnet for artists, photographers, and other craftspeople who have made this part of the Maine coast their home. All of this is only a short drive from the Portland and Bangor international airports, and less than a half-hour from Knox County Municipal Airport which has regular commuter flights to and from Boston.

Classrooms

The Centers classrooms are equipped with the appropriate audio visual equipment for the class being held. Photography classes will have light tables, lupes, presentation walls, slide projectors, and screens. Video production classes will have monitors, video recorders, and tape decks. Every classroom is equipped with comfortable chairs and can be configured with or without tables.

Studio

The Centers Studio is equipped with a variety of strobes, featuring both power pack and monoblock systems. There are enough watt seconds to supply ample light for view cameras and their demanding f-stops. A selection of hot lights are also on hand for those in search of constant light sources. Light stands, grip stands, booms, clamps, sand bags, reflectors, backdrops, gels,

and gobos await your assignments. For those who are learning the ins, outs, swings, and tilts of view cameras, there is a quality selection of cameras and lenses. All of this equipment can be packed into cases and taken on location.

Darkrooms

The Centers individual darkrooms have 4x5 enlargers and a selection of color and cold light heads. Each darkroom is equipped with its own functional sink, timer, easel, trays, tanks, reels, safe lights, and GFI outlets for your boom box. There is a common area that houses the chemistry mixing and supplies. Archival print and film washers stand ready at temperature-controlled sinks. A demonstration darkroom equipped with a large free-standing sink and enlarger provides everyone with a good view of the instructors demonstrations. A drying area for discussions and critiques is located nearby.

Video Production

Professional Hi-8 video camera packages are provided for crews of three to four students, depending on the demands of the shooting location. The package consists of a camera with interchangeable lenses, a shotgun mike with a boom, a lavalier mike, a tripod, and batteries. Radio remote mikes, monitors, and lights are available when needed. There is also a 3/4 SP off-line editing suite.

Computer Facilities

For digital imaging classes, the Center at this writing has four Apple Macintosh computer laboratories, although new laboratories featuring IBM PCs, Amigas, and Silicon Graphics workstations are planned.

Each lab is equipped with a number of student workstations outfitted with hardware and software appropriate to the courses taught in the lab. The instructor has a dedicated station, which is hooked up to a projection system. The Imaging Lab, with 24 student workstations, is the site of introductory and intermediate level imaging classes, in addition to classes of a more general nature, such as Naval Architecture. The Publishing Lab has eight student workstations, several output devices, and a scanning station. Classes such as Electronic Publishing and Advanced Publishing Technology are taught in this laboratory. Participants work with advanced video and audio digitizing technology, as well as with software and hardware to assemble this material in the Multimedia Lab. The Fundamentals Lab has eight workstations and is used for introductory level classes only. Introduction to Page Layout and Mastering the Macintosh Computer are taught in this lab.

Access to the computer labs is available only to students enrolled in a class taking place in that lab. For a catalog of digital courses and more information about current systems, call 800 428 7400 or 207 236 7400 for the Centers main 1993 course catalog, which contains full details. If you are interested in our plans for new labs containing other computer platforms, please call to have yourself put on our mailing list for a catalog of courses to be offered in these new labs.

Maine Coast Photo

The Center is affiliated with the Maine Coast Photo Lab, which can provide you with all photographic supplies, including professional films. Processing for all but lab classes will be done at Maine Coast Photo, which is a Q-LAB. The Lab has can handle C-41 and black-and-white films up to 4x5 and E-6 films up to 11x14.
High Resolution, Inc.

In November 1992, High Resolution, Inc., an imaging center and graphic arts production house founded in Camden in 1986, established a new mission in cooperation with the Center for Creative Imaging. A digital graphic arts laboratory was constructed, accessible across the

Megunticook River by a short footbridge under which runs a fiber optic line for data transfer.

Participants in several of the Centers courses may see the day-to-day operations of a Model Imaging Center, including real-world applications of color management, digital proofing, imagesetting, and final proofing of plate-ready, color-separated film. The cooperative mission of this organization is to provide a practical component to the educational efforts at the Center. A few classes take place at High Resolution; many others feature lectures and demonstrations on site. For information on courses offered at or in conjunction with High Resolution, please call 207 236 7400 or 800 428 7400 for the Centers main 1993 course catalog, which contains more information.

Free Programs

The Center offers numerous free events, available both to class participants and community members. For an updated listing and further information, please call 800 428 7400 or 207 236 7400.

Tours

Tours of the Center are conducted Monday through Saturday. Tours begin in the lobby of the Center and last a little over an hour. Space on scheduled tours is limited; please reserve a spot in advance by calling. Groups of five or more people require special arrangements. Please contact the reception desk at 207 236 7400.

Presentations and Demonstrations

Special evening presentations, including slide shows by visiting instructors, are offered almost every night at the Center. Throughout this catalog, instructors scheduled to speak in the main 1993 course catalog are noted in a yellow box in the calendars at the bottom of each weekly section.
Afternoon programs on selected Saturdays during the year feature demonstrations and lectures by members of the Centers staff. All of these programs are free and open to the public. For information and a current schedule, please call 207 236 7400 or 800 428 7400.

Gallery

The Center has a spacious gallery which features work by past and present participants at the Center, as well as special showings by some of the worlds best-known digital artists and photographers. On the first day of each show, there is a lecture at 6 pm, followed by an opening reception. All shows and openings are free and the public is encouraged to attend.

Weekend Programs

A few special weekend programs occur during the year and focus on special subjects or events. All programs begin at 7:30 pm on the day or days indicated. Please contact the Center for further information.

CCI Course Outline 1993-1994

Purchasing Imaging Technology for Business

Fred Shippey Nov 3-6, Feb 23-26
This course is for decision-makers looking for a comprehensive understanding of the rapidly changing world of digital imaging and its effect on their businesses and clients. The class is paced by lectures, product demonstrations, hands-on experimentation with the Centers comprehensive range of digital imaging equipment, and group discussions on topics such as

investment strategies, changing workflow processes, and risks and rewards of new technologies. Manu-facturers representatives may present short lectures or product demonstrations during the class. Participants should come prepared to work on a personal project to gain firsthand experience of the digital imaging process.
Software Adobe Photoshop
Tuition $1290

Imaging and Illustration

Creative Imaging: Tools and Techniques
Jack Burke Oct 4-8, Feb 7-11
Michael Johnson Oct 18-22, Jan 3-7
Sueki Woodward Nov 1-5
Sam Merrill Nov 15-19
Daniel Grotta & Sally Wiener Grotta Nov 29-Dec 3
This five-day course combines creative investigation with a survey of imaging technology. Issues that will be addressed include: color correction, scanning resolution, image restoration, masking and montage techniques. Students also learn a variety of techniques for creating high-quality digital images. Students are encouraged to bring 35 mm negatives, slides and photographic prints that they are interested in working with during the evening lab time.
Software Adobe Photoshop and/or Aldus PhotoStyler
Tuition $1490

Photo CD as a Publishing and Archiving Solution

Staff Jan 13-14
Participants investigate KODAK Photo CD technology with regards to desktop publishing applications as an image input device and as a final output media. Instruction includes an overview of the Photo CD system, YCC colorspace encoding and the contents of the Photo CD Image Pac and continues with hands on sessions of film scanning and image transfer. The class will evaluate the Photo CD system with regards to archival issues, real world applications and price performance standards.
Software Adobe Photoshop, Aldus Fetch!, Kodak Photo CD Access and Shoebox
Tuition $660

Introduction to PostScript Illustration

Christine Donoghue Oct 6-8
This course is an intensive study in using CorelDRAW! software to create two-dimensional illustrations and typographic arrangements. The programs use paths made of curves and straight line segments to define areas which may be filled with color or patterns, used as masks, or to frame other objects and type. Participants learn to manipulate PostScript language, outline typefaces, knock out color and generate clean, printable files.
Software CorelDRAW! and demonstations of other programs such as Adobe Illustrator and Aldus Freehand
Tuition $990

Illustration and Imaging

Mary Carter & Gary Preister Dec 6-10
This course is an intensive study in using bit-mapped based programs in combination with vector-based programs to create two-dimensional illustrations and typographic arrangements. Participants learn to manipulate PostScript language outline typefaces, combine them with photographic quality images and output professional quality image files. Photoshop, Illustrator, CorelDRAW! and Photo-Styler allow participants to work with a computer simulation of drawing and painting tools.

Software Adobe Photoshop and Illustrator, Aldus PhotoStyler, Fractal Design Painter and CorelDRAW!
Prerequisite Students should be comfortable with one of the bit-mapped and a vector-based programs mentioned above.
Tuition $1490

Advanced Creative Imaging Techniques

Jack Burke Oct 11-13, Feb 14-16
Michael Johnson Jan 10-12
This three-day class is designed for computer artists and designers with in-depth experience in digital imaging. Professional techniques and processes are demonstrated including; luminance and threshold masking, duotoning and halftoning, calculation functions, and importing or translating images for use in page layout programs. Conceptual and creative issues will be critiqued. Students are encouraged to bring specific projects they would like to work on.
Software Adobe Photoshop and/or Aldus PhotoStyler and Fractal Design Painter
Prerequisite Extensive digital imaging experience is mandatory
Tuition $990

Advanced Illustration, Drawing and Collage

Mary Carter & Gary Preister
Dec 13-17
The class is designed for the practicing artist and/or illustrator with advanced computer skills. Emphasis on using the computer to express sophisticated concepts and develop strong solutions for visual problems. Through the combination of traditional and digital media, students will create pieces to be combined with textures in order to develop a unique visual palette and style. Self-expression and class critique will be emphasized. Students should bring portfolios and materials they wish to work with.
Software Adobe Photoshop and Illustrator, Aldus PhotoStyler, Fractal Design Painter and CorelDraw!
Tuition $1490

Creative Vision: Montage and the Image Process

Michael Johnson Oct 25-29
Combining digital technologies with artistic creativity, this class will
challenge participants to create strong, effective and exciting images. The class offers intensive hands on instruction, customized for the needs and skills of the individual class participants with aesthetic critique. Specific techniques to be introduced include advanced masking, calculate functions and moving image files between applications. Artistic vision is the backbone of this class, resulting in full color collages and montages.
Software Aldus PhotoStyler, Adobe Photoshop and Fractal Design Painter
Prerequisites Complete confidence with the afore mentioned software and a deep interest in conceptual and artistic investigation.
Tuition $1490

Publishing

Introduction to Page Layout
Christine Donoghue Oct 11-13 Aldus PageMaker
Peter Koons Jan 3-5 QuarkXPress
Participants learn to use Aldus Page-Maker or QuarkXPress (please specify software when registering for class) to design and compose pages to complete one of the following projects: a newsletter, a brochure, business stationary package or other printed pieces. Basic typesetting techniques, such as hyphenation, kerning and tracking are taught. Participants incorporate small

gray scale images into documents, develop basic stylesheets and proof pages on a standard laser printer to understand how to produce professional quality publications.
Software Aldus PageMaker or QuarkXPress (as indicated)
Prerequisite Basic computer literacy
Tuition $990

Intermediate Page Layout

Christine Donoghue Oct 14-16 Aldus PageMaker
Peter Koons Jan 6-8 QuarkXPress
Continuing where Introduction to Page Layout left off, the course teaches the skills essential for page layout and design of color brochures, newsletters and catalogs. Students will learn to use master pages and in-depth stylesheets, to incorporate line-art illustration and photographic quality images successfully and how to optimize proofing to laser printers and color devices. Also addressed; file management, purchasing and management of typefaces and how to work with a Service Bureau in a professional and efficient manner.
Software Aldus PageMaker or QuarkXPress (as indicated)
Prerequisite CCI Introduction to Page Layout and experience in
traditional graphic design is helpful.
Tuition $990

Advanced Electronic Publishing

Christine Donoghue Oct 18-22 Aldus PageMaker
Peter Koons Jan 10-14 QuarkXPress
The production of accurate proofs, separations and consistent color will be the emphasis of this project-oriented class. Design and production variations will be demonstrated to develop professional problem solving skills. The class includes using keystrokes and palettes, dropping caps and hanging punctuation, using polygons and runarounds and troubleshooting the output process. Participants learn tricks such as printing files up to three times faster and copying formatting from one paragraph to another. This course includes information on managing and purchasing typefaces, hardware and software for efficient operations. Participants are asked to bring projects they are working with in QuarkXPress or PageMaker.
Software Aldus PageMaker or QuarkXPress (as indicated)
Tuition $1490

2-D and 3-D Animation

Introduction to 2-D Animation
Tom Rzonca Jan 20-22
This course explores the tools and techniques used to create and combine drawings, paintings and scanned images for 2-D animation. Traditional cell animation, rotoscoping and key frame animation techniques are addressed. Participants learn to synchronize elements, create lip-synched character animation and add sound effects.
Software Autodesk Animator Pro and Adobe Photoshop
Prerequisite 5-day Creative Imaging: Tools and Techniques or equivalent experience.
Tuition $990

Introduction to 3-D Illustration

Tim Forcade Nov 29-Dec 3 Autodesk
Dave Berry Jan 31-Feb 4 TOPAS
Using the latest 3-D rendering software, participants learn to conceptualize and create 3-D objects and characters. With these basic objects, participants will create scenes with multiple light sources and interesting camera angles. Techniques that will be addressed include: extrusion and lathing of objects, creating and modifying textures, wrapping images and type around objects, and

the control of lighting, reflection, and transparency. The instructor presents different uses for 3-D illustration, including magazine and technical illustration, fine art, and photographic simulation.
Software Autodesk 3-D Studio, Strata Vision 3-D and Adobe Photoshop
Prerequisite 5-day Creative Imaging and/or computer confidence and competence
Tuition $1490

3-D Animation: Tools and Techniques

Dave Berry Dec 6-10 Autodesk
Dan Sayer Feb 7-11 TOPAS
Participants learn to combine images and type into complex animation sequences. The class explores a variety of 3-D form synthesis techniques, concentrating on abstract and technical modeling. Participants may explore both mechanical models of real-world objects or fantastic organic forms to develop characters for 3-D animation. Emphasis is on developing skills to use texture, lighting and motion effects appropiately. Practical issues are also addressed, including frame-by-frame versus real-time recording, legal NTSC colors, resolution issues and hardware configurations.
Software IPAS, Autodesk 3-D Studio or Crystal Animator/TOPAS
Prerequisite 5-day Creative Imaging and/or Introduction to 3-D Illustration
Tuition $1490

Advanced 3-D Illustration, Rendering and Modeling Techniques

Dave Berry Dec 13-17 Autodesk
Dan Sayer Feb 14-18 TOPAS
This course covers all of the elements involved in creating photorealistic 3-D renderings. Using the latest 3-D rendering software, participants learn to conceptualize, create and animate 3-D objects and characters. Participants create effective scenes implementing multiple light sources, interesting camera angles, and special effects such as fog and flare. Phong shading and ray-tracing approaches are covered, with an in-depth analysis of textures, lights, and atmospheric effects. Image processing techniques, such as compositing and motion blur, are addressed for still and animated images.
Software IPAS, Autodesk 3-D Studio or Crystal Animator/TOPAS
Prerequisite Introduction to 3-D Illustration and 3-D Modeling or
equivalent experience.
Tuition $1490

Multimedia

Introduction to Multimedia
TBA Feb 24-26
This course is an introduction to the tools and techniques necessary to develop a strong multimedia presentation. Issues to be addressed include creating an effective production team, storyboarding for efficient production, combining text with images, video and audio, sound editing and importing 2-D and 3-D animation sequences. The end result is to assemble all the pieces, creating a finished and effective multimedia piece. Participants are encouraged to bring still images, sound pieces and video materials to work with during the evening lab time.
Software Asymetrix Compel, Adobe Photoshop, and Autodesk Animator Pro
Prerequisites 5-day Creative Imaging: Tools and Techniques or the equivalent computer imaging experience.
Tuition $1490

Intermediate Multimedia Design & Production:

Making Good Multimedia!
Ted Evans Jan 24-28

TBA Feb 28- March 3
This course covers the steps and issues in creating a multimedia design. Areas covered include interactive storyboards, flowcharting, brainstorming, and working with software and hardware constraints. Emphasis is on shaping an idea into a well thought-out design that works as a multimedia experience. This course offers a detailed look at production issues that affect multimedia projects from the design phase to the end-users final click of the mouse. The class considers the processing power of different computer models, the use of large color images, sound and the incorporation of video.
Software Adobe Photoshop or Aldus PhotoStyler, Autodesk Animator Pro and Asymetrix Compel
Prerequisite 5-day Creative Imaging: Tools and Techniques, Introduction to Multimedia and/or equivalent experience.
Tuition $1490

Corporate Audio-Visual Presentations

Ted Evans Nov 8-12
TBA Feb 28-Mar 3 HSC Interactive
This course covers the role and implementation of computer multimedia for corporate communications and speaker support. Participants learn techniques for building complete multimedia presentations and digital slide shows, including developing a creative concept, scanning and manipulating images, creating original artwork, and incorporating sound, video, animation and 3-D images to create vibrant, highly effective presentations. The instructor shows a variety of professional work to demonstrate different techniques and possibilities.
Software Instructor will use some of the following programs Adobe Photoshop and Illustrator, Autodesk 3-D Studio, Aldus Persuasion, Asymetrix Compel, HSC Interactive, Freelance Graphics and Macromedia Action
Prerequisite Basic knowledge of some of the above programs or 5-day Creative Imaging: Tools and Techniques and/or Introduction to Multimedia.
Tuition $1490

Professional Interactive Multimedia Authoring

Ted Evans Jan 31-Feb 4
This is the class that puts it all together! A project-oriented class in which basic programming techniques, animation control and software engineering principles for the multimedia authoring environment are introduced. Students learn to combine images, type, video and audio with a dynamic interface, creating a professional animation presentation. Learn to create a user friendly interface and program interactive links. Practical hardware and software issues will also be addressed.
Software StrataVision 3D PC, Autodesk 3-D Studio and Animator Pro and Asymetrix Compel
Prerequisite 5-day Creative Imaging and Intermediate Multimedia Design and Production classes, or equivalent experience.
Tuition $1490

TO CONTACT THE CCI:

51 Mechanic Street
Camdem, Maine 04843
Tel. 207 236 7400
Fax 207 326 7490

End.

Release Notes for Motion Works MultiMedia Enabler Toolkit for Visual Basic

Version 1.00 Lite (Beta)

Run Demo

This document contains release notes for Motion Works MultiMedia Enabler Toolkit version 1.00 Lite for Visual Basic. This document contains last minute information and corrections to the other on-line documentation on the CD-ROM.

Welcome to the Motion Works MultiMedia Enabler Toolkit for Visual Basic. The Motion Works MultiMedia Enabler Toolkit is designed for those who want to create MultiMedia applications such as CD-ROM titles, Presentations, training materials, etc...

Provided on the CD-ROM is a Lite version of the Motion Works MultiMedia Enabler Toolkit as well as a demo Visual Basic application which demonstrates the use of the custom controls in the toolkit.

Please keep in mind that this lite version of the Motion Works MultiMedia Enabler Toolkit is intended to demonstrate the creation of MultiMedia applications. This beta release is not intended to be a commercial product and is not for the purpose of resale. In no event shall Motion Works International be liable for any damages of any kind including, without limitation, any special, incidental or consequential damages, even if Motion Works International has been advised of the possiblility thereof..

A complete version of the Toolkit is available from Motion Works International. Please contact Motion Works International at the following address for more information on the full version of the MultiMedia Enabler Toolkit:

Motion Works International
Suite 130 - 1020 Mainland Street
Vancouver, B.C.
Canada, V6B-2T4
Phone: 604-685-9975
FAX: 604-685-6105

The following is a list of the custom controls provided in this Lite version:

Interactive Animation Control (IANIM_LT.VBX)
Interactive PictureBox Control (IPICT_LT.VBX)
Picture Button (PICTBTNL.VBX)
Interactive Video Control (IAVI_LT.VBX)
Sound Annotation Control (SNDANLT.VBX)

The following is a list of the editors provided for use with the custom controls:

PROmotion for Windows Lite Version (PROMOT.EXE)
Interactive AVI Editor Lite Version (IAVIEDIT.EXE)
Sound Annotation Editor Lite Version (SANEDIT.EXE)

The following lists the demos included with this release:

MWMEDEMO.EXE (Demo for the custom controls)
BUMBLE.MWF (PROmotion for Windows file)
PERFCARS.MWF (PROmotion for Windows file)

Installation

Use the following procedure to install the Motion Works MultiMedia Enabler Toolkit onto your hard disk:

1) Select the "Run" menu item from the Program Manager or File Manager and type the following in the edit-box provided:

   d:SETUP.EXE

   Where "d:" is the letter of your CD-ROM drive

2) Follow the instructions provided by the SETUP program to install the toolkit.

Notes, Tips and Corrections

PROmotion for Windows Lite:
Included with the Toolkit is a version of the PROmotion for Window animation playback and editing environment. PROmotion for Windows is a powerful tool for creating animations for use with the Interactive Animation Control. A help file is provided to guide you on how to use the Lite version of PROmotion for Windows.

The lite version of the PROmotion for Windows has a number of features deleted from the full version. The following is a list of the features not in the lite version:

- No sound editor is provided (use the Sound Recorder App to manipulate sounds for import into the animations)
- No complex path point manipulations (smoothing, scaling, distributing, etc...)
- No Cel Sequencer (for frame by frame control of an actor's Cel size and Cel to show)
- Limited number of objects (4 Actors of 8 Cels each, 4 Props)
- Only a single instance of the Paint Editor can be opened at any time
- Limited prop transitions

DLLs
PROmotion for Windows is composed of a number of DLLs. The following is a list of the DLLs required to run the editing envirnment:

PROENG.DLL
CTRLPAD.DLL
TIMELINE.DLL
PRO_DLG.DLL
IMPEXP.DLL
PATH_SEQ.DLL

In addition the following EXE files are provided:

PROMOT.EXE (PROmotion for Windows Editor Application)
PLAYER.EXE (Stand-Alone player for MWF files)

PROENG.INI

PROmotion for Windows requires an INI file to specify some run-time parameters. The following is a listing of the default PROENG.INI file that is placed into the WINDOWS directory:

```
[LIMITS]
MEMORY=4096

[TOOLS]
CONTROL_PAD=CTRLPAD.DLL
TIMELINE=TIMELINE.DLL
PATH_SEQUENCER=PATH_SEQ.DLL
PAINT=PAINT.DLL
ENGINE_DIALOGS=PRO_DLG.DLL
IMPORT_EXPORT=IMPEXP.DLL
```

The TOOLS section specifies the tool DLLs required by the animation editor. PROmotion for Windows first looks for a PROENG.INI file in the current directory before looking in the WINDOWS directory. This allows a local PROENG.INI file to override the settings in the PROENG.INI file in the WINDOWS directory. For a run-time PROENG.INI file, the TOOLS section should be deleted since none of the tool DLLs should be shipped with a completed application that uses the PROmotion for Windows animation engine.

RUN-TIME

The following files are required to run the PROmotion for Windows animation engine:

PROENG.DLL
PROENG.INI

These files should be placed in the current directory of you application or in the WINDOWS directory of your computer.

HELP

The help file included in this lite version is not up-to-date in some areas as there were a number of last minute changes made. If a particular feature indicated in the help file cannot be used then that feature has been removed from the lite version.

IANIM_LT.VBX

The IANIM_LT.VBX is the lite version of the Interactive Animation Control. This control utilizes the animation files created with PROmotion for Windows Editor.

NOTE: This control can cause a UAE if the filename property is invalid.

This control cannot be the parent to any Visual Basic controls. If you want to have a control located on top of the IAnimLT control then you must create the control outside the IAnimLT control and reposition it over the animation control.

RUN-TIME

The following files are required to run the IANIM_LT.VBX custom control:

IANIM_LT.VBX
PROENG.DLL
PROENG.INI

Interactive Video Editor

The Interactive Video Editor is used to create a data file specifying interactivity data for use with the IAVI_LT.VBX custom control.

The lite version of the Interactive Video Editor has a number of features deleted from the full version. The following is a list of the features not in the lite version:

- No custom overriding cursor support (cursors are specified from within the interactivity data file)
- No multiple rectangle support for objects (objects that are not rectangular cannot have transparent(hotspot) areas. Multiple rectangles would allow for far greater control over the hotspot on the frame)
- Layering of the objects cannot be controlled (overlapping object hotspots may cause the wrong object to be returned)

IAVI_LT.VBX

The IAVI_LT.VBX is the lite version of the Interactive Video Control. This control utilizes the data file (IVD) created with the Interactive Video Editor (IAVIEDIT.EXE) and the associated AVI file to provide interactivity.

NOTE: This control can cause a UAE if the filename property is invalid.

Sound Annotation Editor

The Sound Annotation Editor is used to create data files for use with the Sound Annotation Control (SNDANLT.VBX).

The lite version of the Sound Annotation Editor requires you to open a sound annotation data file (.SAN) first before you can link sections of sound from the .WAV file with text.

If you have any suggestions or problems that you would like to report, please contact Motion Works International at the following address, or phone between 9:00AM and 6:00PM PST:

Motion Works International
Suite 130 - 1020 Mainland Street
Vancouver, B.C.
Canada, V6B-2T4
Phone: 604-685-9975
FAX: 604-685-6105

Or send mail via Compuserve to:

ID: 73232,467
User: Selwyn Wan

End.

VIDEO FOR WINDOWS DEVELOPMENT KIT

CONTENTS

1. INSTALLATION NOTES

Use the following procedure when installing the Video for Windows 1.1 software:

1. Run the Setup program in the \WINVIDEO directory of the development kit disc. This installs the Video for Windows runtime and tools on your system.

2. Run the Setup program in the \VFWDK directory of the development kit disc. This installs the development files on your system.

For information on using the capture and editing tools, refer to the individual applications' Help files.

Setting Up A Development System

Required Programming Tools
Microsoft C 7.0 or later is required for building the C and C++ samples. Microsoft MASM 5.1 is required for building the assembly language modules. To edit the Visual Basic samples, you must have Visual Basic version 2.0 or later. The custom controls (VBX) included with the development kit are compatible with Visual Basic 1.0 (as well as Visual C++).

Setting Environment Variables

The Video for Windows development kit includes header and library files needed when using the Video for Window programming interfaces. To reference these required files, modify your INCLUDE, LIB, and PATH environment variables as follows:

- The INCLUDE variable must reference the INC subdirectory of the development kit directory.

- The LIB variable must reference the LIB subdirectory of the development kit directory.

In addition, to run the compiled sample files included with the development kit, you should modify your PATH environment variable to reference the BIN subdirectory of the development kit directory.

For example, assuming the development kit is installed in the default C:\VFWDK directory, you can set the environment variables as follows:
SET INCLUDE=C:\VFWDK\INC;[previous include line]
SET LIB=C:\VFWDK\LIB;[previous lib line]
SET PATH=C:\VFWDK\BIN;[previous path line]
Reading AWM Files Using the Gold Disk File Handler

To read AWM files using the Gold Disk file handler, carry out the following procedure:

1. Add AWMFILE.REG to the registry. To do so, start REGEDIT.EXE, and choose Merge Registration File from the File menu. Choose the AWMFILE.REG file, which is contained in the MISC directory.

2. Make sure the AWMFILE.DLL, AWMFILE2.DLL, and AWMFILE3.TSK files are contained in the SYSTEM directory of your Windows directory.

3. In your WIN.INI file, add the following line to the [MCI EXTENSIONS] section:
   awmfile=AVIVideo

To play AWM files using MPlayer or any AVI player:

1. Run REGEDIT with the /v switch and choose Add Key from the Edit menu. Specify .AWM for the key and MPlayer for the value.

2. CONTENTS OF THE VIDEO FOR WINDOWS DEVELOPMENT KIT

DIRECTORIES ON THE CD

The Video for Windows disc includes the following directories:

Directory	**Contents**
MULTILNG	Contains a sample AVI file with multiple audio streams. Media Player or any other AVI player can play the English audio track. Use the sample application LangPlay to play it with the other audio tracks. (Setup will add a LangPlay icon to your VFW 1.1 DK Program Manager group.)
RUNTIME	Contains the files needed to play back Video for Windows clips. These files include the set of drivers, DLLs, and applications that are installed on an end-user's system. This directory also includes the source files for the runtime setup application.
VFWDK	Contains the Video for Windows development kit.
WINVIDEO	Contains the Video for Windows runtime and tools (video capture and editing).

DIRECTORIES INSTALLED ON YOUR SYSTEM

By default, the Setup routine for the Video for Windows runtime and tools creates a C:\ WINVIDEO directory on your system. This directory contains VidCap and VidEdit. The Video for Windows runtime files are installed in your Windows SYSTEM directory.
By default, the Setup routine for the development kit creates a C:\VFWDK directory on your system. This directory contains the following subdirectories:

Subdirectory	**Contents**
ACMHELP	Contains ACM help files that you can build into your application's help file. For more information, see section 4, USING AUDIO COMPRESSION MANAGER HELP TOPICS.
BIN	Contains compiled versions of the programming samples, along with system files (VBX, DLL) required to run the samples.
DOC	Contains a Microsoft Multimedia Viewer title, the Programmer's Guide. Also contains documentation on custom stream handlers. For more information on the Programmer's Guide, see section 3, ONLINE DOCUMENTATION.
TOOLS	Contains the following useful tools: **RIFFWALK**, an MS-DOS utility that displays the contents of a RIFF file, with special support for AVI and WAV files. Type RIFFWALK -f [filename] to display AVI or WAV headers. Type RIFFWALK to display the program's full syntax and parameters. **UUIDGEN**, an MS-DOS OLE 2.0 utility that generates globally unique IDs for AVIFile handlers. **DDTEST**, a Windows application for testing DrawDib functions, display drivers, and installable video codecs. The DEBUG subdirectory contains debugging versions of MSVIDEO.DLL, MSVIDEO.SYM, MSACM.DLL and MSACM.SYM.
INC	Contains C header files.
LIB	Contains C import libraries.
MISC	Contains drivers, custom controls, and other useful utilities for Video for Windows developers. For information on the contents of this directory, see CONTENTS OF THE MISC DIRECTORY below.
SAMPLES	Contains a series of programming samples. For information on the samples, see SAMPLE APPLICATIONS below.

CONTENTS OF THE MISC DIRECTORY

The MISC subdirectory of the Video for Windows development kit contains a collection of drivers, custom controls, file handlers, and other useful tools. The following sections describe these files.

MCI Drivers — The MISC directory includes the following MCI drivers, which you can install using the Drivers option of the Control Panel.

Driver	Description
MCIPIONR.DRV	MCI driver for the Pioneer laserdisc player.
MCIPANAS.DRV	MCI driver for the Panasonic laserdisc player.
MCIVISCA.DRV	MCI drivers for Sony VISCA devices (VCRs). For more information about this driver, consult the documentation for the development kit.

To install these drivers, use the Drivers option of the Control Panel for adding an unlisted or updated driver.

Visual Basic Custom Controls
The MISC directory contains the following Visual Basic custom controls (VBXs):

Control	Description
MCIWNDX.VBX	Custom control for playing MCI devices. Provides better support for windowed video devices than the MCI.VBX included with Visual Basic.
CAPWNDX.VBX	Custom control for capturing AVI sequences. These custom controls support Visual Basic versions 1.0 through 3.0, as well as Microsoft Visual C++. For more information on these custom controls, consult the documentation for the development kit.

MCI Command Tables
The MISC directory contains the following Media Control Interface (MCI) command tables:

File	Description
DIGITALV.MCI	MCI command-table resource for digital video devices.
DIGITALV.RC	Resource script for using the digital video command table in a digital-video MCI driver.
VCR.MCI	MCI command-table resource for VCR devices.
VCR.RC	Resource script for using the VCR command table in a VCR MCI driver.

File Handlers Used with AVIFile
The MISC directory contains the following file handlers which are used as part of the AVIFile interfaces:

File	Description

AWMFILE.DLL	Animation Works Interactive file handler.
AWMFILE.REG AWMFILE2.DLL AWMFILE3.TSK	These files are developed and supported by Gold Disk.
FLIFILE.DLL	File handler for Autodesk Animator files.

SAMPLE APPLICATIONS

The development kit includes the following programming samples:

Sample	**Description**
ACMAPP	Displays wave file format, and plays, records, and converts wave files.
AVICLIP	Reads frame data and copies it onto the Clipboard.
AVIEDIT	Simple AVI editing application using the editing APIs in AVIFile.
AVIVIEW	Simple AVI viewing application using the read/write APIs in AVIFile.
BRAVADO	Sample capture driver for the Truevision Bravado board.
CAPCPP	Sample Visual C++ capture application that uses the AVICap capture window class and MCIWnd window class for playback.
CAPTEST	Sample capture application that uses the AVICap capture window class.
DSEQFILE	AVIFile file handler for DIB sequences. Implemented in C++.
ICMAPP	Shows how to call the ICM APIs.
ICSAMPLE	Sample of an installable compressor. Good starting point for creating a codec.
ICWALK	Sample application that shows all the codecs installed on the system. Uses the ICM APIs.
IMAADPCM	Sample ACM codec driver.
LANGPLAY	Plays back multi-audio stream AVI files.
MCIPLAY	Simple video playback application. Uses MCIWnd.
MCIPUZZL	Application lets you make a 15-square puzzle of a playing video sequence. Shows how to use installable draw procedures.
MCIVISCA	Sample MCI driver for the VCR command set.
MOVPLAY	Simple application for playing movies using MCI. Builds two versions, one using mciSendCommand and another using mciSendString.

MPLAY	Simple AVI playback application. Includes a subset of the features in Media Player and uses the MCIWnd window class.
MSFILTER	Audio filter for ACM.
MSRLEC	Example of an installable compressor. Uses the RLE compression technique.
PALMAP	Stream handler that translates video streams into new streams of a different bit depth--for example, it can translate from 24-bit to 8-bit.
TXTOUT	Text stream draw renderer for rendering text data in AVI files.
VBCAPTST	Sample capture application implemented using Visual Basic and the CAPWNDX.VBX custom control.
VBMCITST	Sample playback application implemented using Visual Basic and the MCIWNDX.VBX custom control.
WAVEFILE	Sample AVIFile file handler for waveform audio files.
WRITEAVI	Example showing how to use the AVIFILE interface to write files such as those that AVIView reads.

3. ONLINE DOCUMENTATION

Programming documentation for the development kit is provided in a Microsoft Multimedia Viewer title, the Programmer's Guide. The Setup routine for the development kit installs an icon for this title in Program Manager, in the Video for Windows 1.1 DK group. For information on browsing the titles, see the following sections.

Navigating Through the Table of Contents

At any time, you can return to the opening screen: just choose the Contents button.
From the opening screen, click on one of the headings to display a table of contents for that topic. From the topic table of contents, click a name to display a list of subtopics. Then click on a subtopic name to display the information.To move up one level in the hierarchy, choose the Up button.

Navigating Among Topics

To move from topic to topic, choose the browse left << and browse right >> buttons.
To display a list of related topics, choose the See Also button. Then, click a topic name to jump to that topic. To return to the last topic viewed, choose the Go Back button.
To display a list of topics viewed, choose the History button. You can double-click a topic name to jump to that topic.

Displaying an Index of API Elements

To display a list of all functions, messages, and data structures, choose the Index button. To jump to the topic, double-click the API element name. You can scroll the API list or type a partial API name in the dialog box.

Searching the Title

To perform a full-text search on the title, use the following procedure:

1. Choose the Search button.

2. In Search by Word, type the word or phrase you're looking for. For information on using wildcards, Boolean operators, and phrases, choose the Hints button.

3. Choose OK.

4. A Search Results window is displayed listing the results of the search. To jump to a found topic, select the topic name, then choose the Go To button.

4. USING AUDIO COMPRESSION MANAGER HELP TOPICS

Setup installs the Audio Compression Manager help files in the ACMHELP directory. The help project file is ACMHELP.HPJ, and the RTF source files are ACMHELP.RTF, FILTSRC.RTF, and CHOOSSRC.RTF. The ACMHELP.RTF file contains information on using the help topics. The FILTSRC.RTF and CHOOSSRC.RTF files contain the help information for the filter and chooser topics.

You can use the RTF help topics in your application's help file. For example, you could include the three RTF files under the [FILES] entry in your application's help project file. That way, ACM help topics will be built into your application's help file when you compile it.

5. VIDTEST APPLICATION

The Samples disc includes a multimedia performance testing application called VidTest. You can use VidTest to test the performance of multimedia components on your system, including the CD-ROM drive, audio hardware, and video hardware. To allow your customers to obtain useful metrics on the performance of their multimedia systems, you can distribute the VidTest application with your product.

VidTest only works with this version (1.1) of Video for Windows. It does not work with Video for Windows 1.0.

Location of VidTest

VidTest is located on the Video for Windows Samples disc, in the \VIDTEST directory. The application includes a set of sample AVI and files which are stored in the same directory.

Running VidTest

You can run VidTest directly from the Samples disc. Use Program Manager or File Manager to start VIDTEST.EXE from the \VIDTEST directory. For information on using the application, choose the Help button.

Distributing VidTest

If you distribute VidTest with your product, you can use one of the following setup options:

- You can leave VidTest (and the its sample files) on your distribution disc. Your users can run VidTest directly from the disc, in the same way you can run VidTest from the Video for Windows samples disc.

- You can install VidTest (and, optionally, the accompanying sample files) on your user's system.

Leaving VidTest on the Distribution Disc

With this option, you must create a directory on your distribution disc containing the same set of files stored in the \VIDTEST directory on the Video for Windows sample disc. No further modifications are needed.

Installing VidTest on the User's System

If you install VidTest on your user's system, you must make the necessary changes to your Setup routine to install the files. You can install the complete set of VidTest files on the user's system, or you can just install the following files:

- VIDTEST.EXE
- VIDTEST.HLP
- PERFTEST.DLL
- VIDTEST.INI

In either case, you must make the following changes to the VIDTEST.INI file:

- In the VOLUMELABLE= entry, enter the label of the disk on which the sample AVI and WAV files are stored.

- In the VOLUMENAME= entry, enter the name of the disc. This text is displayed in a prompt if VidTest fails to find the specified disc.

- In the file= entries (rlefile=, msvc8file=, msvc16file=,indeofile=, and cinepakfile=), change the value to specify the full path name (without the drive letter) for the sample AVI files used with VidTest. For example, if you place the VidTest samples in a TESTFILE directory, your RLEFILE= entry would read as follows:
  rlefile=\testfile\rlefile.avi

When searching for the VIDTEST.INI file, VidTest first looks in the directory containing VIDTEST.EXE. If the VIDTEST.INI file is not there, VidTest looks in the Windows directory. You should install VIDTEST.INI in the same directory as VIDTEST.EXE.

6. DISTRIBUTION OF RUNTIME FILES

The RUNTIME directory on the Video for Windows development kit CD contains the complete Video for Windows runtime files. You may distribute the runtime files with your product.

CONTENTS OF THE RUNTIME DIRECTORY

The RUNTIME directory contains the following subdirectories:

Subdirectory	Description
DISKS	Contains disk images of the runtime files. The DISKS subdirectory contains USA, FRN, and GER subdirectories containing the English, French, and German versions of the runtime files.
SETUPSRC	Contains the source files for the runtime setup application. Using the Setup Toolkit included with the Microsoft Windows Software Development Kit, you can modify these source files to customize Setup to install your own product along with the Video for Windows runtime.

USING THE DISK IMAGES

You can use the disk images in the DISKS subdirectory to create a Video for Windows setup disk to distribute with your product. Simply copy the contents of the appropriate DISK1 subdirectory to your Video for Windows setup disk.

CUSTOMIZING THE RUNTIME SETUP PROGRAM

The SETUPSRC directory contains the source files for the Video for Windows Setup program. Using the Setup Toolkit included with the Microsoft Windows 3.1 SDK, you can modify the Setup script and layout to accommodate your own product files. For example, you can create an integrated Setup program that installs your hardware specific drivers along with the Video for Windows files. To customize Setup, you must become familiar with the Setup Toolkit; for additional information, see the Setup Toolkit for Windows manual in the Windows 3.1 SDK. The SETUPSRC directory contains the following items:

Contents	Description
MKRTIME.BAT	Batch file that compresses all files and creates a series of disk images in the DISKS subdirectory. Before running MKRTIME.BAT, you must follow the procedure described later in this section.
FILES	Subdirectory containing the uncompressed Video for Windows files. Add your own files to this subdirectory.
LAYOUT	Subdirectory containing the SETUP.LYT file. This file describes the attributes of the files stored in the FILES subdirectory. The disk-layout utility included with the Setup Toolkit uses this file. For each file you add to the FILES subdirectory you must make a corresponding entry in the SETUP.LYT file. For information on layout files, see the documentation for the Setup Toolkit.
SCRIPT	Subdirectory containing the setup script file SETUP.MST. If you add files to the FILES subdirectory, use the SETUP.MST file to tell Setup when and where to copy the files on the end user's machine (using the AddSectionFilesToCopyList function). The SETUP.MST file also tells SETUP what changes (if any) to make to the WIN.INI and SYSTEM.INI files.

To customize a Setup script:

1. Obtain the Setup Toolkit and read the accompanying documentation. Make sure that both DSKLAYT.EXE and DSKLAYT2.EXE are in your path.

2. Use the MS-DOS XCOPY command to copy the SETUPSRC directory and all its subdirectories to your hard disk.

3. Copy your product files to the FILES subdirectory. The Video for Windows runtime files are already in this
subdirectory.

4. Run the DSKLAYT.EXE program and load the SETUP.LYT file from the LAYOUT subdirectory. Enter in the Source Directory dialog box the full path to location of your files. Then add entries for your product files by selecting filenames from the Source Directory list and selecting other options from lists in the dialog box.

5. If necessary, modify the setup script file stored in the SCRIPT subdirectory.

6. Run MKRTIME.BAT. This batch file compresses the files in the FILES subdirectory and creates a COMP temporary directory with the compressed files and a DISK directory containing the disk images.

Once you've finished editing your setup script, test it thoroughly, varying the installation options and system configurations. You can verify results of the setup by examining contents of the target directories for a correct set of files (with correct date and time stamps) and by checking the WIN.INI and SYSTEM.INI files for correct alterations.
After you successfully complete your tests, create your master disks by copying the disk images to floppy disks.

7. DISTRIBUTION OF THIRD-PARTY FILES

The MISC directory of the Video for Windows Development Kit contains the Panasonic laser disc driver MCIPANAS.DRV. Panasonic developed and provided this driver for use with this version of the Development Kit. For support or distribution rights to the driver, please contact:

Panasonic Technical Support:
201-392-4357
Outside of New Jersey: 1-800-524-1448
Inside of New Jersey: 1-800-624-1746

The MISC directory contains the following Gold Disk file handlers: AWMFILE.DLL, AWMFILE.REG, AWMFILE2.DLL, AWMFILE3.TSK. Gold Disk developed and provided these files for use with this version of the Development Kit. Please see the GOLDDISK.TXT for specific licensing information. For support or distribution rights to the driver, please contact:

Gold Disk Technical Support:
416-602-5292
515 Spectrum Way
Building 5
Mississauga, ON L4W5A1

8. KNOWN PROBLEMS OR ERRORS

1. If you send data through the wave mapper with a mapped format and with looping enabled, the looping is still ignored.

2. The audio quality of the implementation of the IMA ADPCM sample is seriously degraded due to improperly encoding the step index in a block header. This algorithm sounds much better if done correctly. Due to this problem and some others, please do not use this codec as sample for implementing IMA ADPCM encode or decode. This code is provided to assist the development of ACM codecs and not as an example of a proper IMA ADPCM implementation.

   If you wish sample code for a proper implementation of this algorithm, please contact the IMA or look for forthcoming releases on Compuserve.

3. In the sample code found in the ACMAPP example, there is a discrepancy between the user interface and the code to implement these features. The dialog box for the ACM Drivers command on the View menu does not implement the Filters button correctly. Also, the dialog box for Formats does not implement the Details button correctly.

4. Versions of ATI's video accelerator prior to 2.1 do not work well with Video for Windows version 1.1. Installing the accelerator software after installing Video for Windows can overwrite Video for Windows version 1.1 components with components from the previous version. Avoid installing any version of the video accelerator unless it is specifically noted to be compatible with Video for Windows version 1.1.

   The vidc.rlec=ativdacc.drv entry in the [installable compressors] section of the SYSTEM.INI file can cause problems when playing large RLE-compressed movies. Remove this entry from the SYSTEM.INI file. For update drivers, contact ATI Technologies Inc.

9. UPDATES TO THE PROGRAMMER'S GUIDE

The following information updates the information in the Video for Windows Programmer's Guide.

WORKING WITH THE DRAWDIB FUNCTIONS

1. DrawDibBegin/DrawDibEnd
   DrawDib changes the state of the DDF_ANIMATE flag only if other parameters change for the DrawDibBegin function. To make sure that DrawDib uses the state you want, reset the state of the DDF_ANIMATE flag with DrawDibEnd and use DrawDibBegin to specify the desired state.

2. DrawDibDraw
   When you call DrawDibDraw, use the biSizeImage member of the BITMAPINFOHEADER structure to specify the number of bytes contained in the image data pointed to by lpBits.

   If you open an old RLE file and get the first frame, it might be returned to you as RGB data and not RLE. This can be easily checked because the size of the frame will be equal to the width (DWORD aligned) times the height of the frame. When this happens, set the compression to 0 (RGB) before you draw it. All subsequent frames will be normal RLE frames.

   The DDF_KEYFRAME flag has not been implemented. Using the DDF_NOTKEYFRAME is still recommended for drawing nonkeyframes.

3. DrawDibChangePalette
   If you have not specified DDF_ANIMATE in DrawDibBegin or DrawDibDraw prior to calling DrawDibChangePalette, DrawDibRealize or a re-draw doesn't change the color of the bitmap. In this case, set the color table values from the values of the palette entry values used in DrawDibChangePalette. For example, if lppe is the pointer to the PALETTEENTRY

array containing the new colors, and lpbi is the LPBITMAPINFOHEADER structure used in the DrawDibBegin or DrawDibDraw, the following fragment updates the DIB color table:

```
//call to change color
DrawDibChangePalette(hDD, iStart, iLen, lppe);
// Update DIB color table now
LPRGBQUAD lpColors = (LPRGBQUAD)((LPBYTE)lpbi + lpbi->biSize) ;
for (i = iStart ; i < iStart+iLen ; i++) {
lpColors[i].rgbRed     = lppe[i].peRed ; lpColors[i].rgbGreen = lppe[i].peGreen ;
lpColors[i].rgbBlue    = lppe[i].peBlue ;
}
```

4. DrawDibGetBuffer
   The DrawDibGetBuffer function returns a pointer to a buffer if DrawDib is being used to draw compressed data. For example, if drawing RGB DIBs, this function returns NULL. Your application should always be prepared to handle NULL returns from DrawDibGetBuffer.

READING AND WRITING AVI FILES

1. AVIFile/AVIStream Functions
   The AVIFile and AVIStream functions should have HRESULT return values rather than LONG return values.

2. AVIStreamGetFrame
   The buffer referenced by the pointer returned for AVIStreamGetFrame should not be modified. This buffer is owned by AVIFile.

3. AVIStreamRead
   If the number of samples to read is specified as zero for AVIStreamRead, it reads data until the buffer is full. If the number of samples to read is specified as AVISTREAMREAD_CONVENIENT for AVIStreamRead, it reads a convenient number of data samples. For example, it reads until the end of a chunk in an interleaved file.

4. AVISTREAMINFO Flags
   The AVISF_DISABLED flag for the AVISTREAMINFO structure should be listed as AVISTREAMINFO_DISABLED. The AVISF_VIDEO_PALCHANGES flag for the AVISTREAMINFO structure should be listed as AVISTREAMINFO_FORMATCHANGES.

5. AVIFILEINFO Flags
   The AVIF_ flags for the AVIFILEINFO structure should be listed as AVIFILEINFO_ flags:
   AVIFILEINFO_HASINDEX
   AVIFILEINFO_ISINTERLEAVED
   AVIFILEINFO_MUSTUSEINDEX
   AVIFILEINFO_WASCAPTUREFILE
   AVIFILEINFO_COPYRIGHTED

WORKING WITH THE INSTALLABLE COMPRESSION MANAGER

1. ICInfo/ICGetInfo
   Use ICGetInfo to obtain complete information about a compressor. ICInfo only returns basic information about a compressor.

WORKING WITH THE AUDIO COMPRESSION MANAGER

1. acmFormatChoose/acmFilterChoose
   The following flags are not defined for acmFormatChoose and acmFilterChoose:

FORMATCHOOSE_FORMAT_ADD
FORMATCHOOSE_FORMATTAG_ADD
FILTERCHOOSE_FORMAT_ADD
FILTERCHOOSE_FORMATTAG_ADD

CONTROLLING MCI DEVICES USING THE MCIWND WINDOW CLASS

1. MCIWndCan Macros
   The MCIWndCanConfig, MCIWndCanEject, MCIWndCanPlay, MCIWndCanRecord, MCIWndCanSave, and MCIWndCanWindow return type is BOOL rather than LRESULT.

2. MCIWndCreate
   The MCIWndCreate macro also has the MCIWNDF_NOOPEN style. This style creates a popup window that excludes the open menu item.

3. MCIWndGetMode
   The MCIWndGetMode macro can return the following MCI modes: MCI_MODE_STOP, MCI_MODE_PAUSE, MCI_MODE_PLAY, MCI_MODE_OPEN, MCI_MODE_RECORD, and MCI_MODE_SEEK.

4. MCIWndOpenInterface
   The MCIWndOpenInterface macro opens a file or stream interface.
   MCIWndOpenInterface(hwnd, pUnk)
   hwnd - Specifies a window handle.
   pUnk - Specifies a handle to a file or stream interface.

   This macro is defined using the MCIWNDM_OPENINTERFACE message.

5. MCIWndPutDest/MCIWndPutSource
   The RECT argument for MCIWndPutDest and MCIWndPutSource should be an LPRECT data type.

CAPTURING AVI FILES USING THE AVICAP WINDOW CLASS

1. Optimizing AVICap Performance
   The following fragment sets parameters for optimizing AVICap performance. Capturing to extended memory provides the best performance if the entire captured sequence can fit in memory. If the sequence will not fit in memory, capturing to disk using memory below 1 megabyte gives better performance than capturing to disk using extended memory.

   ```
   CAPTUREPARMS CapParms;
   ...
   if (fCaptureToDisk) {
                   // Capture to disk
   // Writes from DOS memory give the best disk // performance. So attempt to use 10 buffers // below 1 megabyte.
   // If DOS buffers are unavailable, AVICAP
   // automatically attempts to allocate 10 buffers from // extended memory. Only if zero buffers are
   // allocated, will the capture abort. CapParms.wNumVideoBuffers = 10;
   CapParms.fUsingDOSMemory = TRUE;
   } else {
                   // Capture to Memory
   // Try to allocate 1000 buffers above 1 megabyte. // The actual number of buffers allocated will // depend on available physical memory.
   ```

// Only if zero buffers are allocated, // will the capture abort. CapParms.wNumVideoBuffers = 1000; CapParms.fUsingDOSMemory = FALSE;
}

2. CAPDRIVERCAPS
   The CAPDRIVERCAPS data structure has the following additional members:
   HVIDEO hVideoIn; // Driver In channel HVIDEO hVideoOut; // Driver Out channel HVIDEO hVideoExtIn; // Driver Ext. In channel HVIDEO hVideoExtOut; // Driver Ext. Out channel

   Existing programs do not need to be recompiled for this expanded structure.

3. CAPTUREPARMS
   The CAPTUREPARMS structure has the following additional members:
   DWORD dwAudioBufferSize; // Size of audio buffers
   // (0 = default)
   BOOL fDisableWriteCache; // Attempt to disable write
   // cache

   Existing programs do not need to be recompiled for this expanded structure. The default value for the fUsingDOSMemory member of the CAPTUREPARMS data structure has been changed to FALSE.

5. videoGetErrorText
   If the wSize parameter is zero, videoGetErrorText returns DV_ERR_SIZEFIELD and nothing is copied to the return buffer.

6. capGetMCIDevice
   The capGetMCIDevice reference in the AVI Capture Macros reference table should be capGetMCIDeviceName.

PLAYING AVI FILES USING MCI

1. Dialog Box for the Configure String Command. The Don't Buffer Offscreen checkbox has been removed from the Configure dialog box.

MCI VCR SERVICES

1. Enabling and Disabling Tracks with the VISCA Driver
   When operating in insert mode, the VISCA device driver cannot turn tracks on or off individually. It sets both the video and audio tracks in combination simultaneously. To change the setting of both the audio and video tracks in the insert mode, use the SETAUDIO command immediately followed by the SETVIDEO command. An error message will result from the first command which you can ignore. The VISCA driver changes both tracks to their new state when it receives the second command.

2. MCI VCR String Command Reference
   Change the "set pause (timeout)" command to "set pause timeout (timeout)". The references to "status clock increment rate" should be "capability clock increment rate".

MULTIMEDIA COMPRESSION, DECOMPRESSION, AND RENDERING DRIVERS

1. ICM Draw Handlers
   Returning a value other than ICERR_OK for the ICM_DRAW_WINDOW message in a draw handler prevents subsequent ICM_DRAW_WINDOW messages from being sent to a draw handler. In this case, moving a playback window will not update the window.

End.

The Microsoft Multimedia Developer Relations Group Seminar Series

In the "Miscellaneous" directory on this CD-ROM you will find our "Marketing" file containing the PowerPoint slides we used for a number of marketing oriented seminars.

Please feel free to peruse this file for interesting and informative multimedia statistics, developments, how-to's etc.

MMDRG.

View the marketing slides.

Directory of Localization Consultants

September, 1993 Edition
Prepared by the Microsoft Developer Relations Group

This directory is a listing of companies that do translation, consulting, engineering, marketing, tools, fonts, and distribution work related to localizing software and other materials into non-English languages. This directory is intended as a resource, and does not constitute recommendations or endorsements of individual companies or products by Microsoft Corporation.

If you would like to add or update information about your company, please fill out the form at the end of this file and mail or fax it to the address listed on the form.

ACA Pacific Technology (HK) Ltd.
13/F Hennessy House
313 Hennesy Rd.
Wanchai, Hong Kong
Phone: 852-838-6238
Fax: 852-838-5843
Expertise: Distributor of Microsoft Products

Acer Sertek Incorporated
3-15F, #135 Sec. 2 Chien Kuo N. Rd.
Taipei 10479
Taiwan, R.O.C.
Phone: 02-501-0055
Fax: 02-501-2521
Expertise: *Distributor of Microsoft Products*

American Electronics Association
5201 Great America Parkway, PO Box 54990
Santa Clara, CA 95056-0990
Phone: 408-987-4200
Fax: 408-987-4200
Japan Office: Yonbancho11-4, Chiyoda-ku
Tokyo, Japan 102
Phone: 03-3237-7195
Fax: 03-3237-1237
E-mail: 0003483383@mcimail.com
Contact: David M. Pollack, Director, Japan Market Development
Expertise: The AEA Japan is a non-profit, member-supported trade association with the mission of increasing business opportunities for U.S. electronics companies in the Japanese market. For software companies, the AEA Japan Office has worked in an advisory capacity with hundreds of U.S. companies wishing to locate distributors, establish offices, and otherwise expand their Japan presence. The AEA Japan also runs a Software Committee for the over one hundred U.S. package software companies with an on-going presence in Japan. Both are available through the AEA Japan Office or in the U.S. through the AEA's Publication Dept. in Santa Clara, CA by calling

408-987-4200. Add $5 per copy for shipping and handling.Soft Landing in Japan: A Market Entry Handbook for U.S. Software Companies, August 1992. Price: $95J($65 for AEA members) Software Partners: The Directory of Japanese Software Distributors, October 1992. Price: $150 ($95 for AEA members)

APTEC

Aptec House
Southbank Business Center
Ponton Rd. London SW8 5AT
England
Phone: 44-71-627-1000
Fax: 44-71-498-0496
Expertise: Distributor of Microsoft Products, including Middle East products.

ASIAmarkets

1012 South. Aldine Avenue
Park Ridge, IL 60068
Phone: 708-823-8318
Fax: 708-823-6012
E-mail: 76020.470@CompuServe.com; ericnhauser@delphi.com
Established: 1989 Employees: 10
Contact: Eric N. Hauser
Expertise: Multiple Asian market entry services, consulting and localization.

Basis Technology Corp.

304 Newbury Street
Boston, MA 02115
Phone: 617-262-2062
Fax: 617-262-4284
Japan office: Tanaka Building 3F
1-9-6 Iwamotocho, Chiyoda-ku
Tokyo 101 JAPAN
Phone: 03-3863-2997
Fax: 03-3863-2998
E-mail: basis@std.com; 76416.3365@CompuServe.com
Contact: Carl Hoffman, president
Expertise: DBCS enabling, IME interfacing, support for Kanji fonts, user interface translation, creating Japanese look and feel; contract development of device drivers, libraries, and complete applications for DOS, Windows 3.1, Windows NT, and OS/; translation of online help, printed documentation, courseware, and marketing literature into Japanese.

Experience: Eleven years of developing software products for the Japanese market; localization toolkit with C source-code library, glossary manager, Windows RC file editor, Japanese documentation browser.

Bitstream

215 First Street
Cambridge, MA 02142
Phone: 617 497-6222
Fax: 617 868-4732
Established: 1981 Employees: 150
Expertise: Kanji TrueType fonts, non-Latin fonts, Unicode development
Experience: Localized fonts and applications for France, Germany, the Netherlands (coming: Taiwan, Peoples Republic of China, Korea).

Business Software Alliance

2001 L Street, NW
Washington, DC 20046
Phone: 202-872-5500
Fax: 202-872-5501
Established: 1988
Expertise: The BSA works with government officials and national industry groups to advance the progress of software copyright protection throughout the world. This organization defines a strategy to address the global problem of software theft and works closely with members in countries around the world to implement local programs tailored for each market. It keeps track of piracy rates and laws.

Cambridge Automation, Inc.

One Kendall Square, Suite 2200
Cambridge, MA 02139
Phone: 617- 621-7086
Fax: 617-484-1299
E-mail: shafei@ai.mit.edu
Contact: Nayel El-Shafei
Expertise: Localization of English-based software, development of software in local languages (including multimedia applications), development of fonts. Languages: Arabic, Swahili, Amhari, Tigre, Turkish, Uzbeki, Turkistani, Bahasa (Maylu, Indonesian)

Cambridge Translation Resources

186 South Street
Boston, MA 02111
Phone: 617-451-1233
Fax: 617-451-2247
Contact: Margretha Lund, Vice President, Business Development

Charles View Software

1320 Centre Street
Newton Centre, MA 02159
Phone: 617-527-3655
Fax: 617-527-2975
E-mail: cbf@CharlesView.com
Established: 1991 Employees: 10
Contact: Charles Frankston, President
Expertise: Localize applications under Japanese, Chinese, or Korean Windows, including DBCS enabling of code, translation of resources and documentation, and where necessary, implementing new features required for target market.

Experience: Localization work (partial list): Ventura Publisher for Japan; Band-in-A-Box for Japan and China; 3D/Perspective Jr. for Japan. Custom programming work for (partial list): Language Engineering Corp (Windows UI for English to Japanese machine translation system); Microsoft Corporation (various); Lotus Development Corporation (mail system gateways); Toyo Information Systems (presentation graphics program).

Chase Computers

210 East Main St.

Alhambra, CA 91801
Phone: 818-300-8666
Fax: 818-300-8693
Contact: James Lin
Expertise: Distributor of Microsoft Chinese-language products.

CISD International, Inc.
110 E. 42nd St., Suite 1704
New York, NY 10017
Phone: 212-490-3030
Fax: 212-490-5840

Comjet Information Systems Corp.
2F, #262, Tun Hwa N. Road
Taipei 10484
Taiwan, R.O.C.
Phone: 02-717-4025
Fax: 02-718-4090
Expertise: Distributor of Microsoft Products

CompuFont Limited (A subsidiary of DynaLab Inc.)
Room 504, East Town Building
41 Lockhart Road
Wanchai, Hong Kong
Phone: 852-866-3560
Fax: 852-865-3308
E-mail: o-cfont@microsoft.com
Established: 1989 Employees: 9 in Hong Kong, 160 in Taiwan, 8 in Japan
Contact: Lawrence Mo
Expertise: TrueType fonts for Japan, Taiwan, China, and Korea
Experience: System font supplier to the following Windows products: Chinese Windows 3.0 for Taiwan, Chinese Windows 3.1 for Taiwan, Chinese Windows 3.1 for Peoples Republic of China

Compulingual Institute of Technology
13248 Aurora Ave. N.
Seattle, WA 98133
Phone: 206-367-9123
Fax: 206-361-5041
Established: 1992
Contact: Kim Kim
Expertise: Localization for Korea; multimedia programs in English as a Second Language;, translation

Computer Plus
3850 Wilshire Blvd. #104
Los Angeles, CA 90010
Phone: 213-480-6777
Fax: 213-400-6701
Expertise: Distributor of Microsoft Korean-language products.

Congruent Corporation
110 Greene Street
New York, NY 10012
Phone: 212 431-5100
Fax: 212 219-1532
Contact: Arthur Krietman

Counterpoint Software
Valkenburgerweg 57
6361 EB Nuth, Netherlands
Phone: 32-45-244841
Fax: 32-45-244842
Established: 1991
Contact: J.J.M. Pillaerds, Director
Expertise: Internationalization of existing software, localization for Northern European area.
Experience: Internationalization of MasterTools product line.

DaeWoo
DaeWoo Center
Namdaemonro 5
Chung-Ku, Seoul Korea
Phone: 82-2-759-2283
Fax: 82-2-778-7255
Expertise: Distributor of Microsoft Products

Darya International Software, Inc.
P.O.Box 1284
Bethesda, MD 20827
Phone: 301-299-6433
Fax: 301-299-6539
E-mail: 5965524@mcimail.com
Established: 1989 Employees: 5
Contact: Saeed Darya, President
Expertise: Localize software for Arabic and Persian; TrueType fonts for Persian; Windows software development using C, C++, and Visual Basic; Client/server applications, Database development, Electronic mail systems, Desktop publishing, communication, and educational systems.

Experience: Developed Arabic Windows terminal emulation software to Arabize mainframe applications; developed an Arabic extension to Microsoft Mail; designed and developed Persian language educational software; developed an SGML-like typesetting system for Persian.

Data-Cal
531 East Elliot Road, Suite 145
Chandler, AZ 85225-1152
Phone: 602-813-3100
Fax: 602-545-8090
E-mail: 73354.403@CompuServe.com
Established: 1982 Employees: 65
Contact: David J. Barazoto, Product Manager for WorldFont
Expertise: TrueType Fonts for over 30 languages; key mapper that allows users to re-map keyboard layout with any TrueType or Type 1 Font.

Experience: WorldFont for Windows

DBC Technology Inc.

8652 154th Avenue NE
Redmond, WA 98052
Phone: 206-861-6426
Fax: 206-861-1771
Established: 1991 Employees: 10
Contact: Yasuki Matsumoto
Expertise: Localization for the Japanese market, including software localization, documentation, testing, and porting. Specialize in Windows and Windows NT.

Experience: Staff with more than 20 years of experience in the Far East PC market. Major clients include Compaq, Intel, Microsoft, and Trans Cosmos, Inc.

Dextor Corp.

2-10-3 Kami-Itabashi
Itabashi-ku, Tokyo JAPAN
Phone: 81-3-5920-0580
Fax: 81-3-5920-0588
Expertise: Distributor of Microsoft Products

ER Associates

602 Spruce Circle
Louisville, CO 80027
Phone: 303-661-9122
E-mail: enavarro@nyx.cs.du.edu
Established: Employees:
Contact: Emilio A. Navarro
Expertise: Translation for German, French, Spanish, Italian, and Asian languages.

Experience: Localization and Translation of manuals and software for companies such as HP, IBM, and Borland.

Eten Information System Co., Ltd.

No. 59, Mei-Ning Street, Tai-Shan Hsiang
Taipei Hsien
Taiwan, R.O.C
Phone: 02-296-2900
Fax: 02-735-7956
Experise: Distributor of Microsoft products

Europhile Technology Consultants Ltd

494 Midsummer Bvd, Milton Keynes
GB - MK9 2EA, England
Phone: 44-908-690880
Fax: 44-908-670013
E-mail: aos@eurotech.demon.co.uk
Contact: Anthony St.John

EVERMAX Technology Corp.
2Fl, No.65, Chang-An E. Road Sec.2
Taipei, Taiwan
Phone: 886-2-5080542
Fax: 886-2-5071588
E-mail: 71670.1646@CompuServe.COM
Contact: David Wyatt
Expertise: Hardware/software consulting

Everson Gunn Teoranta
15, Port Chaeimhghein Íochtarach
Baile Átha Cliath 2, Éire
Phone: 353-1-478-2597
Fax: 353-1-283-9396
E-mail: everson@irlearn.ucd.ie; mgunn@irlearn.ucd.ie
Established: 1991 Employees: 2
Contact: Marion Gunn
Expertise: Comprehensive advisory service for the localization of computer software into minority and lesser-used languages, including Irish Gaelic, Welsh, Scottish Gaelic, Latvian, Lithuanian, and Estonian. Linguistic and technical support and the provision of accessories such as keyboard layouts, fonts, and sorting and code page specs. Work on behalf of larger software corporations with minority translators and local distributors. Work in coordination with Irish national organizations as well as the EC European Bureau for Lesser-Used Languages.

Experience: Localized Apple Macintosh System Software 6.0.8 and 7.0.1, Claris MacWrite II, and ClarisWorks into Irish Gaelic. Created specialized fonts for Celtic and other languages. Other packages are being prepared.

FM Towns
101 California St., Suite 1775
San Francisco, CA 94111
Phone: 415-616-9700
Fax: 415-421-3520

Futoba Tosyo Corp.
2-8-22 Kan, Non-Honmachi
Nishi-ku, Hiroshima
Hiroshima-Ken JAPAN
Phone: 81-8-2294-0184
Fax: 81-3-8-2294-0174
Expertise: Distributor of Microsoft Products

GARJAK International
5330 Carroll Canyon Rd. 100
San Diego, CA 92121
Phone: 619-625-0808
E-mail: 70313.2170@CompuServe.com
Contact: Chris Dickey or Gary Erickson

GH Soft

Nad vodovodem 65
108 00 Praha 10
Czech Republik
Phone: 42-2-776633
Fax: 42-776633
E-mail: jindrich@uochb.cas.cz
Established: 1991 Employees: 3
Contact: Jindrich Jindrich
Expertise: Czech fonts for printers (hardware, eventually download), code pages 852, 895. Localization of software for MS-Windows into Czech.

Experience: GH-Link, program for transfer and management of data from Casio digital diaries to PC compantible computers over serial port. MS-DOS and Windows 3.1 versions in Czech and English. CS895, program for automatic installation of code page 895 on computers with installable code page 852.

Global Vision Technology, Inc.

875 Mahler Road,Suite 206
Burlingame, CA 94010
Phone: 415-697-4594
Fax: 415-697-0592
E-mail: 101023.427@CompuServe.com
Established: 1990 Employees: 10
Contact: Tetsuya Chiba, Vice President, Engineering
Expertise: Localize for Japan, DBCS-enabling, multimedia adaptation, PIM Handler

Experience: PROMPT-J, DRAGNET-J, Mathcad-J, Wired for Sound-J, CASE:W-J

Glyph Systems

P.O. Box 134
Andover, MA 01810
Phone: 508-470-1317
Fax: 508-474-8087
E-mail: Glyph@mcimail.com
Established: 1988 Employees: 7
Contact: Harry Ruda
Expertise: Arabic localization, Arabic add-on products (e.g., Arabic dictionary); TrueType Fonts for most foreign languages, including Arabic, Hebrew, Cyrillic, and Thai; custom type design and engineering services

Experience: Provided fonts to Microsoft for Arabic/Hebrew Windows and Word; OEM Arabic dictionary/spell-checker for Arabic Word; localized Ventura Publisher for the Arabic market (Arabic Publis*her)*

Hightech Passport

1590 Oakland Road, Suite B208
San Jose, CA 95131
Phone: 408-453-6303
Fax: 408-453-9434
E-mail: 71644.1005@CompuServe.com
Established: 1992 Employees: 6
Contact: Magdalena Enea
Expertise: European languages Windows and Mac Applications (OCR, graphics, network,

printer drivers, multimedia); localization of software, translation and typesetting of documentation (Frame Maker); multimedia projects

Experience: Selected localization projects: Latern Services Manager for Windows, Netware Lite Utility for Windows (Novell); Omnipage Professional for Windows and Mac, Image Assistant for Windows and Mac (Caere); Various PCL and PS Printer Drivers (Xerox); Presentations and DOS/Windows Software (Mediavision); Unixware resellers Guide and Video (Univel)

I/O Software, Inc.

10970 Arrow Route, Suite 202
Rancho Cucamonga, CA 91730-4839
Phone: 909-483-5706
Fax: 909-483-5710
Japan office: Tachibana Building 5F
2-1-22 Negishi
Taito-ku, Tokyo 110 Japan
Phone: 03-3876-8243
Fax: 03-3876-8189
E-mail: 71302.42@CompuServe.com; AT&T IOS
Established: 1986 Employees: 68
Contact: William Saito, President/CEO; Yoshiaki Higashino, President (Japan)
Expertise: Japan/Japanese computing (Windows NT, Windows NT/J, Windows 3.1/J, OS/2 Japanese, DOS/V, AX, NEC-PC98 series), China (Taiwan)/Chinese; emphasis on hardware & software localization, translation, and distribution; printer and printer device driver development (PostScript, WIFE & DBCS TrueType); document translation; networks; graphics

Experience: Major clients/prior work: NEC Japan (printer driver development and misc.), Japan IBM, Hitachi, Fujitsu, Toshiba (Windows word processor development), ASCII (localization and translation of several software products); in-house development of high resolution kanji PostScript Printer with full Japanese Windows driver support; Others under non-disclosure agreement

ICL

3 Abu El-Mahassen El-Shazly Street
Agouza, Giza, Egypt
Contact: Mr. Ahmed El-Bishry, Localization Unit Manager, ICL Africa and Middle East
Expertise: Arabic and Cyrillic localization

IDOC, Inc.

10474 Santa Monica Blvd., Suite 404
Los Angeles, CA 90025
Phone: 310-446-4666
Fax: 310-446-4661
E-mail: Applelink: Inter.docs
Established: 1980 Employees: 70
Contact: Patti Tessen
Expertise: IDOC has been translating software and documentation for over 12 years. Services provided: turnkey localization of applications into European and Asian languages, including help files, samples, tutorials, and setup programs with experiences on all major operating system platforms; testing and validation of translated applications; desktop publishing of all manuals, documents, and marketing collateral.

Experience: Have localized Windows and DOS products for the following companies:

Macromedia, AST, Hewlett-Packard, Stac Electronics, Novell, Colorado Memory Systems, Fifth Generation, Hayes Microcomputer

IDS (International Documentation Services)

One Kendall Square, Suite 2200
Cambridge, MA 02139
Phone: 617-621-7011
Fax: 617-492-7370
E-mail: ids@world.std.com
Established: 1991 Employees: 10
Contact: Karin Koch
Expertise: translation and localization of technical documentation and product literature; experience in working on large-scale documentation projects which include multi-volume users guides, training manuals and installation guides. Specialize in Spanish, German, French, Italian, Swedish, Dutch, Portuguese, Russian, and Japanese.

Infotech

97-1, Cheongdam 2-Dong
Kangnam-Ku
Seoul, 135-102 Korea
Phone: 82-2-519-5700
Fax: 82-2-515-1973
Expertise: Distributor of Microsoft Products

Institute for Information Industry

8FL, 106, Hoping E. Road, Section 2
Taipei, Taiwan, ROC
Phone: 886-2-737-7289
Fax: 886-2-737-7184
E-mail: dj292@pdd.iii.org.tw
Established: 1979 Employees: 1200
Contact: Dr. Lung-lung Liu, Director, Products Development Division
Expertise: Double-byte enabling and localization for Japan, Korea, Taiwan, and China

Experience: Have localized products for Intersolv, IBM, HP, etc.; developed Japanese and Chinese versions of III-PDD products.

Interface International

319 30th Avenue South
Seattle, WA 98144
Phone: 206-325-5995
Fax: 206-322-3426
Established: 1989 Employees: 2
Contact: Chieko Gaylord
Expertise: Japanese linguistic services, DTP, graphic design manuals (will be produced with an associate graphic designer based in Tokyo, who primarily works on software manuals).

International Communications, Inc.

One Apple Hill - Suite 221
Natick, MA 01760
Phone: 508-615-9232

Fax: 508-653-9183
E-mail: intl.com; AppleLink: D4510
Established: 1984 Employees: 43
Contact: Ellen Lutvak, Manager, Marketing & Sales
Expertise: Software localization, documentation translation and layout, QA and Testing for Japan, Germany, France, Korea, Spain & Latin America, Italy, China, Brazil & Portugal, Sweden.

Experience: Have localized products for: Banyan Systems: Windows client software for VINES networks; Berkeley Systems: AfterDark for Windows; Beyond Inc.: BeyondMail, WinRules; Datastorm Technologies: Procomm Plus for Windows; Delrina: WinfaxPro, DelrinaFax; Fifth Generation: Direct Access for Windows; Inset Systems: Hijaak for Windows; Leica GmbH and Leica KK: Quantimet 500+C; Phoenix Technologies: Phoenix MicroFax; Powercore, Inc.: Network Scheduler; Sotware Ventures: Microphone II; Symantec: TimeLine for Windows.

International Language Engineering Corp. (ILE)
4875 Pearl East Circle, Suite 200
Boulder, CO 80301-6103
Phone: 303-447-2363
Fax: 303-449-2897
E-mail: admin@ILE.com; 73530.2333@CompuServe.com
Established: 1984 Employees: 50
Contact: Elizabeth Bussian, Marketing Manager
Expertise: Windows and multi-platform projects in 2-16 languages simultaneously; full localization teams working year-round on French, German, Italian, Spanish, Swedish, Dutch, Danish, Finnish, Norwegian, Portuguese, Polish and Turkish, plus Japanese, Thai, Korean, Simplified and Traditional Chinese (mainland and Taiwan); Exclusive focus on software localization for 10 years; font creation; localiX, ILEs toolkit for concurrent localized releases; ITLPack, with 30+ functions to write source code independent of character sets and locale (demo disk and licensing available).

Experience: Available upon request.

Japan Entry
65 Oak Ridge Road
West Boxford, MA 01885
Phone: 508 352-7788
Fax: 508 352-7578
Contact: Jack Plimpton

Japanese Language Services
186 Lincoln Street
Boston, MA 02111
Phone: 617-338-2211; 1-800-USA-JAPAN
Fax: 617-338-4611
E-mail: 76505.1755@CompuServe.com
Established: 1982 Employees: 8
Contact: Coleman Yeaw, marketing manager
Expertise: Japanese translation, layout, and typesetting; manuals, documentation, and marketing materials.

Jubilee Tech International
4493 Pissarro Drive

Virginia Beach, VA 23456
Phone: 804-467-1909
Fax: 804-467-3149
E-mail: 73304.3326@CompuServe.com
Established: 1989 Employees: 5
Contact: Jay K. Yoo, President
Expertise: Localization for Korea (also Chinese, Japanese); graphics; document translation
Experience: Logos Research Systems for Windows; WORDSearch; Samsung Electronics; Interlight, International, Inc.

Kappa Type Inc.

P.O. Box 1652
Palo Alto, CA 94302
Phone: 415-322-0135
Fax: 415-326-8844
E-mail: kamal@netcom.com
Established: 1993 Employees: 3
Contact: Kamal Mansour
Expertise: Provider of standard- and custom-encoded outline fonts for all Latin-script languages; Custom keyboard layouts (software) under Windows 3.1; all fonts are legally licensed.

Experience: Postscript fonts for East European languages (Latin-script); standard keyboard layouts for Polish, Czech, Slovak, Romanian, Hungarian, Croatian, Slovene,... under Windows 3.1.

Kinki System Corp.

Sumitomo-Bank
Nihon-Ichi Bldg.
1-17-17 Nippon-Bashi
Chuou-ku, Osaka
Osaka-Ken Japan
Phone: 81-6-644-6625
Fax: 81-6-644-6630
Expertise: Distributor of Microsoft Products

Knox Computer Systems, Inc.

10055 Barnes Canyon Road, Suite K
San Diego, CA 92121
Phone: 619-535-0771
Fax: 619-535-0773
Japan office: Knox Data International
Knox Building, 5-27-4 Minami-ohi
Shinagawa-ku, Tokyo 140 Japan
Phone: 03-3766-0411
Fax: 03-3766-0613
E-mail: 70651.1670@CompuServe.com
Established: 1974 (Tokyo), 1987 (U.S.) Employees: 80 (Tokyo), 5 (U.S.)
Contact: John Riley or Rick Sakai in the U.S., Mr. Oishi or Mr. Itoh in Japan
Expertise: Japanese only; engineering (i.e. 2-byte enabling, NEC 9801 porting, Unicode implementation, font integration, addition of special Japanese functions, etc,); graphics/fonts software; Windows development experience for both English and Japanese Windows 3.1; English to Japanese translation and DTP; software distribution in Japan

Experience: Japanese localization for Harvard Graphics for Windows, American Airlines SABRE Travel Software, Arts & Letters for Windows & UNIX (#1 Windows graphics software in Japan), Gerber Scientific Products (#1 sign company in the world); other clients include Quad Tech International, Symantec, NEC, Lotus, Prime/ComputerVision

Language Automation, Inc.

1806 Parkwood Drive
San Mateo, CA 94403
Phone: 415-571-7877
Fax: 415-571-6294
E-mail: localize@LAI.com
Established: 1993
Contact: David Lakritz
Expertise: Japanese localization, including translation

LE Centre, Inc.

220 D San Benancio Canyon
Salinas, CA 93908
Phone: 800-995-0975
Fax: 408-484-0940
E-mail: 73030.1472@CompuServe.com
Established: 1992
Contact: Phil Faris, President
Expertise: Translation in many languages, especially French, German, and Persian; work in conjunction with the Defense Language Institute in Monterey, CA and Altura Software; Microsoft Solutions Channel Consultant Alliance member; interactive multimedia software

Liker

192 Da-Dong, Chung-Ku
Seoul Korea
Phone: 82-2-756-2-3211
Fax: 82-2-757-2300
Expertise: Distributor of Microsoft Products

Linotype-Hell

1230 Oak Mead Pkwy
Sunnyvale, CA 94086
Phone: 408-720-9595
Expertise: *Fonts*

LOCATECH

Kurfuerstenstrasse 36
44147 Dortmund, Germany
Phone: 49-231-88 17 86
Fax: 49-231-88 16 64
E-mail: 100117,413@CompuServe.com
Established: 1993 Employees: 5-6
Contact: Anette Schachtner or Dirk Loehn
Expertise: Localization for Germany

Experience: In-house localization experience at Microsoft; localized Help for Visual Basic 3.0, for Visual Basic for Applications, and for Visual Basic for Microsoft Project; translated parts of

documentation and Help for Windows NT and Microsoft Excel 4.0; localized various marketing material

Matrix USA

University Park at MIT
26 Landsdowne Street, Suite 420
Cambridge, MA 02139
Phone: 617-577-8955
Fax: 617-621-0623
Japan Office: Matrix Corporation
King Hills Hakusan, 4th Floor
2-25-10 Hakusan
Bunkyo-ku, Tokyo 112 Japan
Phone: 81-3-5689-3535
Fax: 81-3-5689-3534@CompuServe.com
E-mail: 73067.3426@CompuServe.com (U.S.); 101004.1654@CompuServe.com, 101125.207@CompuServe.com (Japan)
Established: 1989 (U.S.), 1985 (Japan) Employees: 2 (U.S.), 25 (Japan)
Contact: Bruce Henderson
Expertise: Sotware localization for Japan: DTP, CAD/CAM, business applications, etc. on Windows, DOS, Unix; software product-support services in Japan: object-oriented languages, databases, and other development tools; custom software development for Japan: CAD/CAM, multimedia on Windows, DOS, Unix

Experience: Localized Ventura Publisher for Windows for Japan (with Charles View Software); provided Japanese localization consulting for Computervisions Design View; provided product support and training services in Japan for Borland, Microsoft, Neuron Data, UniSQL, and Versant; Developed SignGraph (sign-cutting application) for Japanese plotter manufacturer.

Mendez Translations S.A.

Avenue Franklin Roosevelt
8 - B-1050 Brussels, Belgium,
Phone: 322-647-2700
Fax: 322-647-5550
Established: 1970 Employees: 97
Contact: Joe Mendez
Expertise: Localization of Windows and Macintosh applications; languages and systems from English into French, German, Spanish, Dutch, Italian, Swedish, and Russian.

Experience: Localized products (including testing and bug fixing) include Claris FileMaker Pro for Windows and Mac, Claris Works, CA-Clipper 5.2, Compaq DOS 5.0, CA-Superproject, CA-dBFast, Aldus Personal Publisher, DEC-Query for the Macintosh. Other projects now being localized for Microsoft

Metis Technology, Inc.

530 Atlantic Avenue
Boston, MA 02210
Phone: 617-292-0550
Fax: 617-292-0552
E-mail: info@metis.com
Contact: Glenn Adams
Expertise: Custom software for supporting Unicode and ISO/IEC 10646 character sets

Experience: Object-oriented Universal Text Library that supports Unicode and ISO/IEC 10646

character data; Japanese Language Unicode Support Kit; script support libraries for display of Arabic, Indic, and SE Asian scripts

MicroBurst

9043 Shady Grove Court
Gaithersburg, MD 20877
Phone: 301-330-2995
Fax: 301-330-8609
E-mail: 71660.3416@CompuServe.com
Established: 1979 Employees: 15
Contact: Mark Frederiksen
Expertise: Localize for Japan, Korean, China, Europe, and bi-directional (Arabic, Hebrew, Farsi) for DOS, OS/2 and Windows

Experience: Localized Macromind Action!, Campbell Services On-time, Something Goods Champion, IBM Asia Pacific Take-Five

Monotype Typography

53 West Jackson Blvd, Suite 504
Chicago, IL 60604
Phone: 312-885-1440
Fax: 312 939-0378

MSI

10F, #3, Sec. 3. Ming Shen E Rd.
Taipei, Taiwan, R.O.C.
Phone: 02-506-3320
Fax: 02-506-0772
Expertise: Distributor of Microsoft Products

Multilingual Computing Magazine

111 Cedar St., Suite 5
Sandpoint, ID 83864
Phone: 208-263-8178
Fax: 208-263-6310
E-mail: 71224,1003@CompuServe.com; AppleLink: MULTILINGUAL
Established: 1988 Employees: 7
Contact: Seth Schneider
Expertise: Multilingual Computing is the leading trade publication for end users and developers of language computing products. The magazine is a quarterly publication containing editorials and a buyers guide. Useful for those wanting to follow the industry, locate products, or reach a specialized market.

Experience: Previously published under the titles of Worldwide Language Guide and SESAME Bulletin

Pacific Software Publishing, Inc

2737 77th Avenue SE, Suite 210
Mercer Island, WA 98040
Phone: 206-232-3989
Fax: 206-236-8102
Established: 1987 Employees: 18

Contact: Ken Uchikura
Expertise: Translation, localization, compatibility testing for Japan

Rank Xerox Technical Centre
Customer Business Services
Bessember Road
Welwyn Garden City, Hertfordshire AL7 1HE 44 707 383672
Phone: 44-707-383672
E-mail: john_freeman.wgc1@rx.xerox.com
Established: Employees:
Contact: John Freeman, CBS Marketing
Experience: 25 years in supporting the nationalization of Rank Xerox and Xerox products

River Systems, Inc.
50 Airport Parkway
San Jose, CA 95110
Phone: 408-437-7763
E-mail: danielho@netcom.com
Contact: Daniel Ho
Expertise: Software localization, particularly for the Asian market

Rocky Mountain Translators
5757 Central Avenue, Suite G
Boulder, CO 80301
Phone: 303-449-6954
E-mail: wsmith@rmt.com
Contact: Walter Smith

Senco Systems
1103 New Way Centre
409-415 Hennessy Rd.
Hong Kong
Phone: 852-838-9303
Fax: 852-834-4876
Expertise: Distributor of Microsoft Products

SimulTrans
145 Addison Avenue
Palo Alto, CA 94301
Phone: 415-323-1335
Fax: 415-323-3233
E-mail: bil@simultrans.com
Contact: Bill Libby
Expertise: Localization and development work into European and Asian languages.

Softrade International
19849 S.E. 123rd Street
Issaquah, WA 98027
Phone: 206-682-7050
Fax: 206-235-7943

E-mail: 76400.2113@CompuServe.com
Contact: Jim Sameth
Expertise: Tokyo-based company doing software localization into Japanese for Windows, DOS, Macintosh, and UNIX-based products; assist in sales and distribution.

Experience: Recently completed a localized Windows utilities package

Softbank Corp.
NS-Takanawa Bldg.
2-19-13 Takanawa
Minato-ku, Tokyo Japan
Phone: 81-3-5488-1135
Fax: 81-3-5488-1132
Expertise: Distributor of Microsoft Products

Soft Bank Korea
2 Fl. DaeHanJaeDang Bldg., 7-23
Shinchun-Dong, Songpa-Ku
Seoul Korea
Phone: 82-2-411-1676
Fax: 82-2-411-1699
Expertise: Distributor of Microsoft Products

Software and Services International
3 Avenue de Tervuren
B-1040 Bruxelles, Belgium
Phone: (00322) 734-6603
E-mail: ben@sap-ag.de
Contact: Ben Rockefeller or Loretta Anania
Expertise: Localization into French, German, and Dutch

Software Japan Corp.
Kyouei-Ponpian Bldg.
3-3-6 Iwamoto-chou
Chiyoda-ku, Tokyo Japan
Phone: 81-3-3862-2765
Fax: 81-3-3862-2526
Expertise: Distributor of Microsoft Products

Softwing Corp.
Komagome-General Bldg.
4-8-11 Komagome
Toshima-ku, Tokyo Japan
Phone: 81-3-3576-1137
Fax: 81-3-3576-1787
Expertise: Distributor of Microsoft Products

Stan Corporation of America
100 Pine St., Suite 1875
San Francisco, CA 94111

Phone: 415-677-0766
Fax: 415-677-9985
Japan office: Stan Software Corporation
21 Ryutsudanchi Hiraishi, Kawauchi-cho
Tokushima 771-01 Japan
Phone: 81-886-65-5527
Fax: 81-886-65-5375
E-mail: 70314.3152@CompuServe.com
Established: 1982 (Japan), 1990 (U.S.) Employees: 20 (Japan), 5 (U.S.)
Contact: Thaxter Sharp, Vice President (U.S.); S. Kondo, President (Japan)
Expertise: Full-service Japan partner for U.S. software companies; complete turnkey localizations of Windows products from English to Japanese; translation and re-design of packaging and documentation as appropriate; manufacturing of retail boxes for sale in Japan; distribute and sell products through wholesale, retail and direct channels in Japan; tech support, customer service, direct fulfillment and upgrade path service for vendors who wish to begin shipping in Japan

Experience: localized and publish exclusively GetC File Shuttle Express, On Time for Windows, Links386 Pro, Championship Golf Courses, Champion for Windows; distributor of Media Vision and Comptons New Media multimedia products in Japan

Sung Kan (Unalis) Computer Book Co., Ltd.
5F, 593 Tun-Hwa South Road
Taipei 10654
Taiwan, R.O.C.
Phone: 02-708-2125
Fax: 02-735-7956
Expertise: Distributor of Microsoft Products

System-Pro Computer Ltd. (GBS Division)
22/F Tai Yau Bldg.
181 Johnston Rd.
Hong Kong
Phone: 852-893-0303
Fax: 852-838-5872
Expertise: Distributor of Microsoft Products

SystemSoft
333 17th St., Suite L
Vero Beach, CA 32960
Phone: 407-770-3371
Fax: 407-569-1937
Contact: George H. Durr
Expertise: Distributor of Microsoft Japanese-language products

Systran (S) Pte Ltd
133 New Bridge Road #21-01
Chinatown Point
Singapore 0105
Phone: 65-5388449
Fax: 65-5388515
E-mail: sayhow@solomon.technet.sg; 74710.1350@CompuServe.com

Contact: Foo Say How

Tech Pacific
11-1F, 296 Sin-Yi Rd. Sec 4
Taipei, Taiwan, R.O.C.
Phone: 02-703-8561
Fax: 02-703-8575
Expertise: Distributor of Microsoft Products

Tech Pacific (HK) Ltd.
3/F, South East Warwick House
28 Tong Chong St.
Quarry Bay, Hong Kong
Phone: 852-564-9200
Fax: 852-561-6537
Expertise: Distributor of Microsoft Products

The Wordmill
P.O.Box 1817
Healdsburg, CA 95488
Phone: 707-431-7414
Fax: 707-431-0358
E-mail: 76547.1770@CompuServe.com
Established: 1988 Employees: 5
Contact: Lee Curtis, President
Expertise: Translation and localization for all major European and Asian languages; foreign language typesetting

The Write Stuff
5508 35th Ave. NE
Seattle, WA 98105
Phone: 206-524-4423
Fax: 206-524-9019
E-mail: v-berylg@microsoft.com
Contact: Beryl Gorbman
Expertise: Technical publications in English and in foreign languages

Experience: Currently localizing software for Microsoft and other local firms; have translated text documents for many firms, here and in Japan.

Toin America Corporation
3353 Peachtree Road, NE, Suite 1120
Atlanta, GA 30326
Phone: 404-240-4110
Fax: 404-240-4111
E-mail: TOIN@mcimail.com
Established: 1961 (Japan), 1990 (U.S.) Employees: 150 (Japan), 6 U.S.
Contact: Michael Wayne Cooper
Expertise: Japanese technical documentation writing and translation; European technical translations; machine translation

Experience: Japanese Windows Demo Manual

Trans Pacific Communications
212 Technology Dr., Suite Y
Irvine, CA 92718
Phone: 714-753-9195
Fax: 714-753-9295
Contact: John Goodale

Transcend
1138 Villaverde ln.
Davis, CA 95616
Phone: 916-756-5834
Fax: 916-756-4810
E-mail: luism@sybase.com
Contact: Luis Miguel

U-Lead Systems, Inc.
970 West 190th St., Suite 520
Torrance, CA 90502
Phone: 310-523-9393
Fax: 310-523-9399
Contact: Rosemary Bach

Unitrendix Corp.
19300 S. Hamilton Ave #165
Gardena, CA 90248
Phone: 310-329-3265
Fax: 310-329-3601
Established: 1988 Employees: 27
Contact: Hideki Nakano
Expertise: Localization for Japan; network system integration (UNIX based Client/Server); distributor for Microsoft Japanese-language products

Experience: Localized SuperFax for Windows, SuperFax Network, SuperVoice

Windows Presentation Corp.,
P.O. Box 15091
Fremont, CA 94539
Phone: 510-770-1747
Fax: 510-770-9649
Contact: JY Lin, VP of engineering
Expertise: Development, DBCS-enabling, and translation of Windows and Windows NT software between English and Chinese, including traditional and simplified Chinese characters; contract development and marketing services to software companies seeking to reach the Chinese markets in Taiwan, PRC, Hong Kong, and Singapore; licensed and localized its own line of Windows and Windows NT products for the Far East

Experience: Chinese RFFlow, Kids Karaoke

World Ready Software

123 Townsend Street, Suite 636
San Francisco, CA 94107
Phone: 415-957-1300
Fax: 415-957-1314
Contact: David Greco, President
Expertise: Localization development and consulting work

X.Net, Inc.

3452 Shangri-La Road
Lafayette, CA 94549
Phone: 510-937-2426
Fax: 510-937-2919
E-mail: 71250.1322@CompuServe.Com
Established: 1984 Employees: 1
Contact: Carl W. Brown
Expertise: Globalizing software so that it is enabled to facilitate localizing in world wide markets; provide multi-platform development from Pcs to mainframes, including MS-DOS and Windows platforms; adaptation of programs for multiple locales, active in industry to support technology to enable multi-lingual programs; Consulting and training, referrals to products and services that help in internationalization and localization efforts.

XA International

14510-A Big Basin Way, Suite 240
Saratoga, CA 95070
Phone: 408-741-5577
Fax: 408-741-0512
E-mail: jturley@netcom.com
Established: 1976 Employees: 10
Contact: Jim Turley, Director
Expertise: Translation and localization of software into Asian double-byte character languages: Japanese, Chinese Traditional, Chinese simplified, Korean, Indonesian, and Thai.

Experience: Localized WinHan v2.0 into Japanese, Chinese, and Korean on Microsoft Windows platforms

XL Soft International

15375 Barranca Pkwy, Suite A204
Irvine, CA 92718-2204
Phone: 714-453-2781
Fax: 714-453-8811
E-mail: 76350.1457@CompuServe.com
Established: 1988 Employees: 10
Contact: Paul Watson
Expertise: Japan, NEC PC-9800, Multimedia

Experience: WindowsMAKER, RoboHELP (Blue Sky Software); RFFlow; Sing-a-long Kids Karaoke (Dr. Ts Music Software); Master Track Pro (Passport Designs, Inc.), LEXIS 2000 (Mead Data Control), Applied Structure (Rasna Corporation), Icon Designer (LDC Computer Corporation)

Xyron Corporation

3080 Olcott St. #225-D
Santa Clara, CA 95054
Phone: 408-727-6336
Fax: 408-727-4548
Contact: Hiro Wakiyama

Zinc Software Inc.

405 South 100 East, 2nd Floor
Pleasant Grove, UT 84062
Phone: 801-785-8900
Fax: 801-785-8996
E-mail: chrijm@zinc.com
Established: 1990 Employees: 20
Contact: Brad Davis
Expertise: Single- and double-byte enabled, multiplatform development tools. Support for Unicode.

Microsoft Subsidiaries

Beijing Representative Office
Unit #551, 5th Flr Off Bldg
New Century Hotel, No. 6
Southern Road Capital Gym
Beijing, PRC 100046
CHINA
(86)(1)849-2148-50 phone
(86)(1)849-2148 fax

Quarry Bay HONG KONG
MS Hong Kong Limited
Far East Region
11/F., City Plaza 4
12 Taikoo Wan Road
Quarry Bay
HONG KONG
(852)804-4200 phone
(852)560-2217 fax

Tokyo JAPAN
MS Company, Limited
Sasazuka NA Bldg.
50-1, Sasazuka 1-Chome
Shibuya-ku, Tokyo 151
JAPAN
(81)(3)5454-8000 phone
(81)(3)5454-7970 fax

Tokyo JAPAN
Osaka Sales Office
Kitahama "Excel" Bldg. 5F
6-11, Kitahama 2 chome
Chuo-ku, Osaka 541
JAPAN
(81)(06)231-4470 phone

(81)(06)231-4499

Seoul KOREA
Dong-Seong Building
158-9 Samseong-Dong,
Gangnam-Gu
Seoul, 135-090
KOREA
(82)(2)552-9505 phone
(82)(2)555-1724

Kuala Lumpur MALAYSIA
9th Floor, MCB Plaza
6 Cangkat Raja Chulan
50200 Kuala Lumpur
MALAYSIA
(603)230-0299 phone
(603)230-0301

SINGAPORE
10 Anson Toad #21-15
International Plaza
SIngapore 0207
REPUBLIC OF SINGAPORE
(65)227-6833 phone
(65)227-6811 fax

Taipei TAIWAN
P.O. Box 27-16, No. 3, 10th Fl.
Min Chuan E. Road, Sec. 3
Taipei, Taiwan
REPUBLIC OF CHINA (ROC)
(886)(2)504-3122 phone
(886)(2)504-3121 fax

THAILAND
Silom Complex Building 24th Fl.
191 Silom Road, Bangrak
Bankok 10500
THAILAND
(662)231-3920 phone
(662)231-3924 fax

Sydney AUSTRALIA
Microsoft Park
65 Epping Road
North Ryde, NSW 2113 (sydney)
Mailing Address: PO Box 91
North Ryde NSW 2113
AUSTRALIA
(61)(2)870-2200 phone
(61)(2)870-2769 fax

Auckland NEW ZEALAND
Freepost 3731
P.O. Box 8624

Symopnds Street
Aukland 1
NEW ZEALAND
(64)(9)358-3724 Phone
(64)(9)358-3725 fax

Vienna AUSTRIA
Favoritentstrasse 321
A-1100 Vienna (Wien)
AUSTRIA
(43)(1)68-76-07 phone
(43)(1)68-162710 fax

Brussels BELGIUM
Rue Colonel Bourg 123-125
1140 Brussels
BELGIUM
(32)(2)730-39-11 phone
(32)(2)735-1609 fax

Prague CZECHOSLOVAKIA
MS s.r.o.
Panska 6/1
110 00 Praha 1
Prague
CZECH REPUBLIC
(42)(2)268-320 phone
(42)(2)266-020 fax

Hedenhusen DENMARK
MS Danmark ApS (A/S)
Microsoft Danmark Aps
Lautruphøj 1-3
DK - 2750 Ballerup
DENMARK
(45)(44)89-01-00 phone
(45)(44)68-55-10 fax

Espoo FINLAND
MS OY Suomi
Kutojantie 3
02630 Espoo
FINLAND
(358)(0)525-501 phone
(358)(0)522-955 fax

EUROPEAN HEADQUARTERS
MS Europe
Tour Pacific
Cedex 77
92977 Paris-La Defense
FRANCE
(33)(1)46-35-1010 phone
(33)(1)46-35-1030 fax

Paris FRANCE
MS France S.a.r.l.

18 Avenue Du Quebec
Zone de Courtaboeuf
91957 Les Ulis Cedex
FRANCE
(33)(1)69-86-46-46 phone
(33)(1)64-46-06-60 fax

Munich GERMANY
MS G.m.b.H. (Munich)
Edisonstrasse 1
W-85713 Unterschleissheim
(Munich)
GERMANY
(49)(89)3176-0 phone
(49)(89)3176-1130 fax

Athens GREECE
MS Hellas SA
64 Kifissias Avenue
Polis Centre, 15125 Maroussi
Athens
GREECE
(30)(1)689-6663 phone
(30)(1)689-6662 fax

Dublin IRELAND
MS Distribution Limited
Blackthorn Road
Sandyford Industrial Estates
Dublin 18
IRELAND
(353)(1)2955-333 phone
(353)(1)2955-363 fax

Milan ITALY
MS S.p.A.
Centro Direzionale Milano Oltre
Palazzo Tiepolo
Via Cassanese 224
20090 Segrate-Milano
ITALY
(39)(2)269-121 phone
(39)(2)2107-2020 fax

Hoofddorp NETHERLANDS
MS B.V.
Saturnusstraat 46-62
2132 BH Hoofddorp
Mailing: P.O. Box 264
2130 AJ Hoofddorp
THE NETHERLANDS
(31)25-03-89189 phone
(31)25-02-37761 fax

Oslo NORWAY
MS Norge AS

Gjerdrumsvei 21
N- 0881 Oslo
PO Box 12
Korsvoll - 0808 Oslo
NORWAY
(47)(22)02-2500 phone
(47)(22)95-0664 fax

Warsaw POLAND
MS sp. z.o.o.
ul. Grzybowska 80/82
Warszawa 00-844
POLAND
(48)(22)6615-433 phone
(48)(22)6615-434 fax

Lisbon PORTUGAL
MSFT Sotware Para Microcomputadores, Lda.
Galerias Alto da Brra
Piso 3
Av. das Descobertas
2780 Oeiras
PORTUGAL
(351)(1)441-2205 phone
(351)(1)441-2101 fax

Moscow RUSSIA
MS A/O
Ulitsa Staraya Basmannaya 14
Moscow (Moskau), 103064
RUSSIA
(007)(502)220-4215 phone
(007)(502)220-4213 fax

Madrid SPAIN
MS Iberica S.R.L.
Avda. Colmenar Viejo
Sector Foresta 2, 3, 4
28760 Tres Cantos, Madrid
SPAIN
(34)(1)804-0000 phone
(34)(1)803-8310 fax
Stockholm SWEDEN
MS Aktiebolag AB
Findlandsgatan 30
Box 27
S-164 93 Kista
SWEDEN
(46)(8)752-5600 phone
(46)(8)750-5158 fax

Zurich SWITZERLAND
MS AG
Alte Winterthurerstr. 14A
CH - 8304 Wallisellen
(Zurich)

SWITZERLAND
(41)(1)839-61-11 phone
(41)(1)831-08-69 fax

Berkshire UNITED KINGDOM
MS Limited
Microsoft Place
Wharfdale Road
Winnersh Triangle Wokingham
Berkshire RG11 5TP
UNITED KINGDOM
(44)(734)270-001 phone
(44)(734)270-002 fax

New Delhi INDIA
MS India
Paharpur Business Centre
21 Nehru Place
New Delhi - 110 019
INDIA
(91)(11)644-4457 phone
(91)(11)644-4469 fax

Tel Aviv ISRAEL
MS Israel Ltd.
Tuval St. 34
Ramat-Gan 52522 (Tel-Aviv)
ISRAEL
(972)(3)575-7034 phone
(972)(3)575-7065 fax

Buenos Aires ARGENTINA
MS de Argentina S.A.
Avenida del Libertador 602
Piso 21A
1001 Buenos Aires
ARGENTINA
(54)(1)814-0356 phone
(54)(1)814-0372 fax

Sao Paulo BRAZIL
MS Informatica Limitada
Av. Eng. Luiz Carlos Berrini
1253-13 Andar
04571 902 Sao Paulo - SP
BRAZIL
(54)(1)814-0356 phone
(54)(1)814-0372 fax

CARIBBEAN
MS Caribbean, Inc (CINC)
Metro Square Bldg
Calle 1, Lotte 11,
Metro Office Park
Guaynabo, PUERTO RICO 00920
(206) 936-8661 phone

Santiago CHILE
MS Chile S.A.
Avenida Presidente Kennedy
5146 Piso 7
Santiago CHILE
(56)(2)218-5771 phone
(56)(2)218-5747 fax

Bogota COLOMBIA
Colombian Subsidiary of MS Corp. Ltda.
Carrera 9a 99-02 Piso 2
Santafe de Bogota D.C.
COLOMBIA
(57)(1)618-22-45/92/85/94 phone
(57)(1)618-2255 fax

Quito ECUADOR
Corporacion MS del Ecuador (ESA)
Avda. Naciones Unidas 1014 y
Avda. Amazonas (esquina)
Edificio Banco de la Previsora
Torre A, Piso 10, Oficina 1001
Quito
ECUADOR
(593)(2)55-12-19 phone
(593)(2)52-90-65 fax

Mexico, D.F. MEXICO
MS Mexico, S.A. de C.V.
Blvd. M.A. Camacho 32
Piso 2
Lomas de Chapultepec, 11000
Mexico D.F.
MEXICO
(52)(5)325-09-10 phone
(52)(5)280-79-40 fax

Humacao PUERTO RICO
MS Puerto Rico, Inc.
Humacao Industrial Park
Road 3, KM 77.0 vice 77.8
Humacao 00792
PUERTO RICO
(809)852-7077 phone
(809)852-7076 fax

Caracas VENEZUELA
MS Venzuela S.A.
Centro Profesional Eurobuilding
Piso 7, Oficina 7-E
Calle La Guairita, Chuao
Caracas
VENEZUELA
(58)(2)91-47-39 phone
(58)(2)92-38-35 fax

Dubai U.A.E.
MS Middle East
P.O. Box 52244
Khalid Bin Al Waleed St.
Juma Al-Majid Bldg., 7th Fl.
Dubai
UNITED ARAB EMIRATES
(971)(4)527-444 phone
(971)(4)513-888 fax

Casablanca MOROCCO
MS Morocco
Immeuble Angle Blvd. Zerktouni
A Rue de La Haye, 7th Floor
Casablanca
MOROCCO

Toronto CANADA
MS Canada, Inc.
320 Matheson Blvd. West
Mississauga, ON L5R 3R1
CANADA
(416)568-0434 phone
(416)568-1527 fax

Johannesburg SOUTH AFRICA
MS (S.A.) (Proprietary) Limited
P.O. Box 5817
Rivonia, 2128
SOUTH AFRICA
(27)(11)444-0520 phone
(27)(11)444-0536 fax

Istanbul TURKEY
MS Turkey (Tltd)
MS Bilgisayar Yazilim Hizmetleri Ltd
Barbaros Plaza Is Merkezi, C-Blok
No: 145, Kat. 21
Emirhan Caddesi
Dikilitas, Besiktas
Istanbul
TURKEY
(90)(1)227-25-14 phone
(90)(1)258-59-58 fax

CORPORATE HEADQUARTERS
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
USA
(206)882-8080 phone
(206)936-7329 fax

Microsoft Windows Globalization Resource Kit
Directory Application Form

Please complete and fax to:
Globalization Team
Developer Relations Group
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
FAX: (206) 93MSFAX

Company name
_____

Address
_____

_____

Phone
_____

Fax
_____

Email
_____

Year established _____
Approx number of employees _____

Contact name and position: _____

Areas of expertise (i.e. localize for Sweden, truetype fonts for Japan)

_____

_____

_____

_____

_____

(optional) Prior work, i.e. "localized XYZ for Windows", etc

_____

_____

_____

_____

End.

Licensing Agreement

IMPORTANT-- READ CAREFULLY BEFORE USING SOFTWARE PACKET(S). Unless a separate multilingual license booklet is included in your product package, the following License Agreement applies to you. By opening the sealed packet(s) containing the software, you indicate your acceptance of the following Microsoft License Agreement.

MICROSOFT LICENSE AGREEMENT

(Microsoft Multimedia Pack for Windows)

This is a legal agreement between you the end user and Microsoft Corporation. By opening the sealed software packet(s) you are agreeing to be bound by the terms of this agreement. If you do not agree to the terms of this agreement, promptly return the unopened software packet(s) and the accompanying items (including written materials and binders or other containers) to the place you obtained them for a full refund.

MICROSOFT SOFTWARE LICENSE

1. GRANT OF LICENSE. This License Agreement permits you to use one copy of the enclosed Microsoft software program (the "SOFTWARE") on a single computer. The SOFTWARE is in "use" on a computer when it is loaded into temporary memory (i.e. RAM) or installed into permanent memory (e.g., hard disk, CD ROM, or other storage device) of that computer. However, installation on a network server for the sole purpose of internal distribution shall not constitute "use" for which a separate license is required, provided you have a separate license for each computer to which the SOFTWARE is distributed.

2. COPYRIGHT. The SOFTWARE is owned by Microsoft or its suppliers and is protected by United States copyright laws and international treaty provisions. Therefore, you must treat the SOFTWARE like any other copyrighted material (e.g. a book or musical recording) except that you may either (a) make one copy of the SOFTWARE solely for backup or archival purposes, or (b) transfer the SOFTWARE to a single hard disk provided you keep the original solely for backup or archival purposes. You may not copy the written materials accompanying the SOFTWARE.

3. MULTIMEDIA CONTENT LICENSE. Microsoft grants to you a royalty-free license to use, copy, reproduce, and distribute the sample multimedia files of the SOFTWARE, consisting of the audio, image, and/or video files identified in the SOFTWARE or accompanying written materials as samples (the "Samples") which you may freely use, subject only to these restrictions: (i) you may not use the Samples in any broadcast; and (ii) you may not resell or redistribute the Samples for value in the form of, or as part of, a library or clip software product consisting predominantly of audio, image and/or video materials. The video clips in the "Fun Stuff" directory may only be used for personal viewing. These clips may not be distributed commercially or non commercially.

4. OTHER RESTRICTIONS. You may not rent or lease the SOFTWARE, but you may transfer the SOFTWARE and accompanying written materials on a permanent basis provided you retain no copies and the recipient agrees to the terms of this Agreement. You may not reverse engineer, decompile or disassemble the SOFTWARE. If the SOFTWARE is an update or has been updated, any transfer must include the most recent update and all prior versions.

5. DUAL MEDIA SOFTWARE. If the SOFTWARE package contains both 3-1/2" and 5-1/4" disks, then you may use only the disks appropriate for your single-user computer. You may not use the other disks on another computer or loan, rent, lease, or transfer them to another user except as part of the permanent transfer (as provided above) of all SOFTWARE and written materials.

LIMITED WARRANTY

LIMITED WARRANTY. Microsoft warrants that (a) the SOFTWARE will perform substantially in accordance with the accompanying written materials for a period of ninety (90) days from the date of receipt; and (b) any hardware accompanying the SOFTWARE will be free from defects in materials and workmanship under normal use and service for a period of one (1) year from the date of receipt. Any implied warranties on the SOFTWARE and hardware are limited to ninety (90) days and one (1) year, respectively. Some states/jurisdictions do not allow limitations on duration of an implied warranty, so the above limitation may not apply to you.

CUSTOMER REMEDIES. Microsoft's and its suppliers' entire liability and your exclusive remedy shall be, at Microsoft's option, either (a) return of the price paid or (b) repair or replacement of the SOFTWARE or hardware that does not meet Microsoft's Limited Warranty and which is returned to Microsoft with a copy of your receipt. This Limited Warranty is void if failure of the SOFTWARE or hardware has resulted from accident, abuse, or misapplication. Any replacement SOFTWARE or hardware will be warranted for the remainder of the original warranty period or thirty (30) days, whichever is longer. **Outside the United States, neither these remedies nor any product support services offered by Microsoft are available without proof of purchase from an authorized non-U.S. source.**

NO OTHER WARRANTIES. To the maximum extent permitted by applicable law, Microsoft and its suppliers disclaim all

other warranties, either express or implied, including, but not limited to, implied warranties of merchantability and fitness for a particular purpose, with regard to the SOFTWARE and the accompanying written materials. This limited warranty gives you specific legal rights. You may have others which vary from state/jurisdiction to state/jurisdiction.

NO LIABILITY FOR CONSEQUENTIAL DAMAGES. To the maximum extent permitted by applicable law, in no event shall Microsoft or its suppliers be liable for any damages whatsoever (including without limitation, damages for loss of business profits, business interruption, loss of business information, or any other pecuniary loss) arising out of the use of or inability to use this Microsoft product, even if Microsoft has been advised of the possibility of such damages. Because some states/jurisdictions do not allow the exclusion or limitation of liability for consequential or incidental damages, the above limitation may not apply to you.

U.S. GOVERNMENT RESTRICTED RIGHTS

The SOFTWARE and documentation are provided with RESTRICTED RIGHTS. Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of The Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 or subparagraphs (c)(1) and (2) of the Commercial Computer Software -- Restricted Rights 48 CFR 52.227-19, as applicable. Manufacturer is Microsoft Corporation/One Microsoft Way/Redmond, WA 98052-6399.

If you acquired this product in the United States, this Agreement is governed by the laws of the State of Washington. If this product was acquired outside the United States, then local law may apply.

Should you have any questions concerning this Agreement, or if you desire to contact Microsoft for any reason, please contact the Microsoft subsidiary serving your country or write: Microsoft Customer Sales and Service/One Microsoft Way/Redmond, WA 98052-6399.

End.

Font UtilitiesSome Handy-Dandy Font-Handling Functions

Charlie Kindel, Microsoft Premier Support Services

Kyle Marsh, Microsoft Developer Network Technology Group

Created: January 10, 1993

Abstract

Handling fonts in Microsoft® Windows™-based applications can involve a lot of grunt work. For example, essential tasks such as creating fonts, retrieving point sizes, and calculating text metrics can take up a lot of the developer's time and effort. This article describes some useful font-handling functions that take care of most of this tedious work, thus making it easier to create and manipulate fonts in Windows.

The FONTUTIL sample application demonstrates most of the font-handling functions discussed in this technical article.

Introduction

The font utilities described in this article are primarily for applications that need the relatively simplistic font model presented in the **ChooseFont** common dialog box. **ChooseFont** deals with fonts using four primary variables: *typeface name*, *typeface style*, *point size*, and *effects*. **ChooseFont** also allows for color selection, but the font-handling functions do not currently support this feature.

The font-handling functions allow an application to create and interrogate fonts in a manner that is consistent with the **ChooseFont** model. For example, you can create a font by using the **ReallyCreateFont** function and specifying the typeface name, typeface style, and point size; you can retrieve the typeface style of a font by using the **GetTextStyle** function; and so on.

The font utility functions support Microsoft® Windows™ versions 3.0 and 3.1, and Windows NT™ version 3.1. You can include the functions either in an application or in a dynamic-link library (DLL).

Using the Font Utilities

There are two important points to remember when using the font utilities:

You must export the **fnEnumReallyEx** callback function in the application's or DLL's module definition file, or use an appropriate compiler option.

The font utilities are English-specific; to support other languages, you must use the **LoadString** function with string resources.

Exporting fnEnumReallyEx

The application or DLL containing the font utilities must export the font enumeration callback function **fnEnumReallyEx** in the module definition file. For example, the FONTUTIL sample application's FONT.DEF file exports **fnEnumReallyEx** with the following lines:

```
EXPORTS
     fnEnumReallyEx       @103
```

Alternatively, Microsoft C/C++ version 7.0 and Visual C++™ version 1.0 users can compile the utilities with the **GA** option (for applications) or the **GD** option (for DLLs). These options eliminate the need to export **fnEnumReallyEx** in the module definition file.

Supporting Different Languages

The font utilities use hard-coded English strings for font attributes. To support other languages, you must use string resources and load them into the utilities with the **LoadString** function. For example, to support German, you must translate the term "bold" into something like "Fettdruck." The variables holding the strings are currently declared as follows:

```
char szBold[]        = "Bold" ;
char szItalic[]      = "Italic" ;
char szBoldItalic[]  = "Bold Italic" ;
char szRegular[]     = "Regular" ;
```

To support other languages, the variables would be declared as:

```
extern LPSTR rglpsz[] ;

#define szBold       rglpsz[IDS_BOLD]
#define szItalic     rglpsz[IDS_ITALIC]
#define szBoldItalic rglpsz[IDS_BOLDITALIC]
#define szRegular    rglpsz[IDS_REGULAR]
```

Implementing this feature is left as an exercise for the reader.

Function Reference

The font utilities consist of seven functions:

Two of these functions create fonts: **ReallyCreateFont** creates a font for a specific device context (DC), and **CreateMatchingFont** creates a font that can be used by two device contexts.

The remaining five functions retrieve information about a selected font: **GetAverageWidth** computes the character width, **GetTextStyle** retrieves the style name, **GetTextFullName** retrieves the full typeface name, **GetTextPointSize** retrieves the point size, and **GetTextExtentABCPoint** computes the dimensions of the font based on ABC widths.

The sections below describe these functions in alphabetical order.

CreateMatchingFont

This function gets the closest matching font between two device contexts. This information is useful when producing WYSIWYG output.

Syntax

```
HFONT WINAPI CreateMatchingFont
   (HDC hdcScr,
    HDC hdcPrn,
    LPSTR lpszFace,
    LPSTR lpszStyle,
    LPSTR nPointSize,
    UINT uiFlags);
```

Parameters

hdcScr
Specifies the device context to which the specified font will be mapped. If this parameter is NULL, the screen DC is used.

hdcPrn
Identifies a printer DC. The typeface, style, and point-size parameters specify a font that is already selected into the printer DC.

lpszFace
Points to a null-terminated string that identifies the typeface name of the font to be created (for example, "MS Sans Serif").

lpszStyle
Points to a null-terminated string that identifies the font style to be used (for example, "Bold Italic"). If *lpszStyle* is NULL, the style indicated by *uiFlags* is used.

nPointSize
Specifies the point size of the font to be created (1 point = 1/72 inch).

uiFlags
Specifies flags that modify the behavior of the function. You may use any combination of the following:

Value	Meaning
RCF_NORMAL	Creates a font with the regular font style.
RCF_ITALIC	If *lpszStyle* is NULL, creates an italic font.
RCF_UNDERLINE	Creates an underlined font.
RCF_STRIKEOUT	Creates a font with strikeout.
RCF_BOLD	If *lpszStyle* is NULL, creates a bold font.
RCF_NODEFAULT	Specifies that a system font should not be returned. Normally,

CreateMatchingFont always attempts to create a font. If the specified parameters do not match an installed font, it creates a system font unless this flag is set.

Return value

The **CreateMatchingFont** function returns the handle to the created font. If a font could not be created, the function returns NULL.

GetAverageWidth

This function retrieves the text metrics of the font currently selected into the device context and returns the average character width. It computes the average character width by using **GetTextExtent** on the "abc...xyzABC...XYZ" string. This computation method works much better than the **tmAveCharWidth** value in the **TEXTMETRIC** structure returned by the **GetTextMetrics** function for alphanumeric, Latin-based proportional fonts, especially when used for dialog-box unit calculations.

Syntax

```
int WINAPI GetAverageWidth
   (HDC hdc,
    LPTEXTMETRIC lptm);
```

Parameters

hdc
Identifies the device context for which the text metrics information should be returned.

lptm
Points to the **TEXTMETRIC** structure that receives the metrics.

Return value

The **GetAverageWidth** function returns the average character width of the currently selected font, computed as described below (the *rgchAlphabet* parameter contains the string "abc...xyzABC...XYZ").

If the code detects it is running on Windows 3.0, it uses:

```
nAveWidth = LOWORD( GetTextExtent( hDC,
                    (LPSTR)rgchAlphabet, 52 ) ) / 52 ;
```

If the code detects it is running on Windows 3.1, it uses:

```
nAveWidth = LOWORD( GetTextExtent( hDC,
                    (LPSTR)rgchAlphabet, 52 ) ) / 26 ;
// Round up
nAveWidth = (nAveWidth + 1) / 2 ;
```

If the code detects neither Windows 3.0 or 3.1, it assumes a later version of Windows (for example, Windows NT) and uses:

```
GetTextExtentPoint( hDC, (LPSTR)rgchAlphabet, 52, &size ) ;
nAveWidth = size.cx / 26 ;
nAveWidth = (nAveWidth + 1) / 2 ;
```

Comments

The calculations above are identical to those that USER uses to calculate character widths for dialog-box units.

GetTextExtentABCPoint

The **GetTextExtentABCPoint** function computes the dimensions (advance width) of the specified text string, taking ABC widths into consideration. **GetTextExtentABCPoint** uses the currently selected font to compute the width and height of the string in logical units, without considering any clipping.

The "A" width of a character is the distance added to the current position before drawing the character glyph. A negative "A" width indicates an underhang. The "B" width of a character is the width of the drawn portion of the character glyph. The "C" width of a character is the distance added to the current position to provide white space to the right of the character glyph. A negative "C" width indicates an overhang. The *advance width* of a character is the sum of the A, B, and C widths.

Syntax

```
UINT WINAPI GetTextExtentABCPoint
   (HDC hdc,
    LPSTR lpszString,
    int cbString,
    LPSIZE lpSize);
```

Parameters

hdc
Identifies the device context.

lpszString
Points to a text string.

cbString
Specifies the number of bytes in the text string.

lpSize
Points to a **SIZE** structure that will receive the dimensions of the string.

Returns

The **GetTextExtentABCPoint** function returns the "A" width of the first character in the string. If the "A" width is negative, **GetTextExtentABCPoint** returns its absolute value. In the case of an error, the function returns zero.

Comments

Unlike **GetTextExtentPoint**, this function uses the **GetCharABCWidths** function to calculate the extent of the specified string. This function is very useful if you want to calculate the full extent of a string.

For example, if you use the standard Windows **GetTextExtentPoint** function to calculate the string extent, the string will appear clipped on both the left and the right (Figure 1).

Figure 1. Using GetTextExtentPoint

If you calculate the string extent with the **GetTextExtentABCPoint** function, but do not use the return value to offset the first character, the string will appear clipped on the left (Figure 2).

Figure 2. Using GetTextExtentABCPoint (return value ignored)

Figure 3 illustrates the string when you use the **GetTextExtentABCPoint** function, and offset the first character by the return value. The string appears in its entirety, with no clipping.

Figure 3. Using GetTextExtentABCPoint (return value considered)

Under Windows 3.0 or when a non-TrueType® font is selected, the **GetTextExtentABCPoint** function is simply a wrapper around **GetTextExtent** or **GetTextExtentPoint**.

GetTextExtentABCPoint also works around a bug in Windows 3.1: The **GetCharABCWidths** function calculates the ABC spacing incorrectly for fonts that simulate boldface. For example, Windows 3.1 does not include the TrueType Wingdings™ Bold font. If the user selects a font that was created with the Wingdings typeface name and a weight greater than FW_NORMAL into a DC, the graphical device interface (GDI) simulates bold by overstriking the characters. When an application uses **GetCharABCWidths** to determine the advance width of this font, the ABC widths returned are off by one for each character. To work around this bug, the **GetTextExtentABCPoint** function adds one logical unit to the "B" width of each character.

GetTextFullName

This function retrieves the full name of the selected font, and copies it into the buffer as a null-terminated string.

The full typeface name is found only in TrueType fonts. It usually contains a version number and other identifying information. For example, the full name of the Windows Times New Roman Bold Italic typeface is *Monotype:Times New Roman Bold Italic:Version 1 (Microsoft)*.

Syntax

```
UINT WINAPI GetTextFullName
   (HDC hdc,
    UINT cbBuffer,
    LPSTR lpszStyle);
```

Parameters

hdc
Identifies the device context.

cbBuffer
Specifies the buffer size in bytes. If the full name of the typeface is longer than the number of bytes specified by this parameter, the name is truncated.

lpszFace
Points to the buffer that contains the full typeface name.

Return value

If the **GetTextFullName** function is successful, its return value specifies the number of bytes copied to the buffer, not including the terminating null character. Otherwise, the return value is zero.

Comments

This function uses the **GetOutlineTextMetrics** function in Windows 3.1 and later, and returns a null string in Windows 3.0.

GetTextPointSize

This function calculates the point size of the selected font using the following code:

```
nPtSize = MulDiv( tm.tmHeight - tm.tmInternalLeading,
                  72,
                  GetDeviceCaps( hDC, LOGPIXELSY ) );
```

Syntax

```
UINT WINAPI GetTextPointSize
```

```
   (HDC hdc);
```

Parameter

hdc
Handle to the device context.

Return value

The **GetTextPointSize** function returns the point size of the currently selected font.

GetTextStyle

This function retrieves the style name of the selected font, and copies it into the buffer as a null-terminated string.

Syntax

```
UINT WINAPI GetTextStyle
   (HDC hdc,
    UINT cbBuffer,
    LPSTR lpszStyle);
```

Parameters

hdc
Identifies the device context.

cbBuffer
Specifies the buffer size in bytes. If the style name is longer than the number of bytes specified by this parameter, the name is truncated.

lpszStyle
Points to the buffer that contains the typeface style name.

Returns

If the **GetTextStyle** function is successful, its return value specifies the number of bytes copied to the buffer, not including the terminating null character. Otherwise, the return value is zero.

Comments

This function uses the **GetOutlineTextMetrics** function in Windows 3.1 and later, and the **GetTextMetrics** in Windows 3.0.

ReallyCreateFont

This function creates a font based on typeface name, typeface style, and point size.

Syntax

```
HFONT WINAPI ReallyCreateFont
   (HDC hdc,
    LPSTR lpszFace,
    LPSTR lpszStyle,
    int nPointSize,
    UINT uiFlags);
```

Parameters

hdc
Identifies the device context to use (NULL for the screen DC).

lpszFace
Points to a null-terminated string that identifies the typeface name of the font to be created (for example, "MS Sans Serif").

lpszStyle
Points to a null-terminated string that identifies the typeface style of the font to be created (for example, "Bold Italic"). If *lpszStyle* is NULL, the style indicated by *uiFlags* is used.

nPointSize
Specifies the point size of the font to be created (1 point = 1/72 inch).

uiFlags
Specifies flags that modify the behavior of the function. You may use any combination of the following:

Value	Meaning
RCF_NORMAL	Creates a font with the regular font style.
RCF_ITALIC	If *lpszStyle* is NULL, creates an italic font.
RCF_UNDERLINE	Creates an underlined font.
RCF_STRIKEOUT	Creates a font with strikeout.
RCF_BOLD	If *lpszStyle* is NULL, creates a bold font.
RCF_NODEFAULT	Specifies that a system font should not be returned. Normally, **ReallyCreateFont** always attempts to create a font. If the specified parameters do not match an installed font, it creates a system font unless this flag is set.

Return value

The **ReallyCreateFont** function returns the handle to the created font. If a font could not be created, the function returns NULL.

Supporting Different Versions of Windows

You can include the font utilities in an application or a DLL. In an application, the font utilities call **MakeProcInstance** in the font enumeration procedure; in a DLL, they do not call this function.

You can compile the font utilities to run under Windows 3.0, Windows 3.1, or Windows NT version 3.1. The font utilities are based on the Windows 3.1 application programming interface (API); some extra steps were taken to support the Windows 3.0 and Win32™ APIs.

The font utilities use two techniques to get information about the available fonts:

- To search for a font, the font utilities call **EnumFonts** when running under Windows 3.0, and **EnumFontFamilies** when running under Windows 3.1.
- To get information about the current TrueType font, the font utilities call **GetOutlineTextMetrics**. The functions do not call **GetOutlineTextMetrics** for non-TrueType fonts or when running under Windows 3.0.

Supporting Windows 3.0

The font utilities use three functions that do not exist in Windows 3.0: **GetOutlineTextMetrics**, **GetCharABCWidths**, and **GetTextExtentPoint**. These functions operate on TrueType fonts, which were introduced in Windows 3.1. The font utility functions operate correctly if called under Windows 3.0, or for non-TrueType fonts. However, if an application links calls to **GetOutlineTextMetrics**, **GetCharABCWidths**, and **GetTextExtentPoint**, the Windows 3.0 kernel will not load the application. To avoid this problem, you can compile the font utilities so that they use the internal wrapper functions **MyGetOutlineTextMetrics**, **MyGetCharABCWidths**, and **MyGetTextExtentPoint** instead of calling the Windows 3.1 functions directly. These wrapper functions call **GetProcAddress** to dynamically link in the functions at run time so the Windows 3.0 kernel can load the application.

In addition, the font utilities call the **MyGetOutlineTextMetrics** and **MyGetCharABCWidths** wrapper functions only when running under Windows 3.1. The font utilities do call **MyGetTextExtentPoint** when running under Windows 3.0, so this wrapper function includes logic to call the Windows 3.0 **GetTextExtent** function instead of the Windows 3.1 **GetTextExtentPoint** function. You can also compile the font utilities so that they do not run under Windows 3.0. In this case, the Windows 3.1 functions are called directly.

Here is how the wrapper functions are set up:

```
#ifdef WORK_IN_WIN30
/* Wrapper functions we use so we can run in 3.0 and 3.1 since
 * GetOutlineTextMetrics, GetCharABCWidths, and GetTextExtentPoint
 * do not exist in 3.0, and the application will not load if
 * kernel 3.0 sees that we're importing externals that don't
 * exist in GDI.
```

```
 */

UINT NEAR PASCAL MyGetOutlineTextMetrics(HDC hdc, UINT ui,
                        OUTLINETEXTMETRIC FAR* lpOTM);
BOOL NEAR PASCAL MyGetCharABCWidths(HDC hdc, UINT ui1, UINT ui2,
                        ABC FAR* lpABC) ;
BOOL NEAR PASCAL MyGetTextExtentPoint(HDC, LPCSTR, int, SIZE FAR*);

/*
 * You probably want to make this a global in your app. There is,
 * after all, no reason to call GetVersion all the time.
 */

#define fWin30 ((BOOL)(LOWORD( GetVersion() ) == 0x0003))

#else

#define fWin30 FALSE
/*
 * If we are a 3.1-only app, just call the 3.1 functions directly!
 */
#define MyGetOutlineTextMetrics GetOutlineTextMetrics
#define MyGetCharABCWidths      GetCharABCWidths
#define MyGetTextExtentPoint    GetTextExtentPoint

#endif
```

Another step the font utilities take for Windows 3.0 compatibility is to call **EnumFonts** when running under Windows 3.0 and **EnumFontFamilies** when running under Windows 3.1. The font utilities do this through run-time dynamic linkingby calling **GetProcAddress** and using the Windows function that corresponds to the correct version of Windows. Here is how the font utilities call either **EnumFonts** or **EnumFontFamilies**:

```
   int  (WINAPI *lpfnEnumFont)(HDC,LPSTR,FONTENUMPROC,LPARAM) ;
   /*
    * On 3.0 call EnumFonts, on 3.1 call EnumFontFamilies.
    *
    * EnumFonts is exported from GDI at ordinal #70.
    * EnumFontFamilies is exported from GDI at ordinal #330
    * (3.1 and later).
    */

   (FARPROC)lpfnEnumFont = GetProcAddress( GetModuleHandle( "GDI" ),
      (LPCSTR)(fWin30 ? MAKEINTRESOURCE( 70 )
                      : MAKEINTRESOURCE( 330 )) ) ;

   (*lpfnEnumFont)( hdcCur, lpszFace, lpfn, (LPARAM)(LPVOID)&elf ) ;
```

Supporting the Win32 API

Supporting the Win32 API is much simpler than supporting Windows 3.0. The font utilities do not need Windows 3.0 support for Win32, so they can call the **GetOutlineTextMetrics**, **GetCharABCWidths**, **GetTextExtentPoint**, and **EnumFontFamilies** functions in Windows 3.1

directly without using the wrapper functions. For example:

```
#ifdef WIN32
EnumFontFamilies( hdcCur, lpszFace, lpfn,(LPARAM)(LPVOID)&elf ) ;
#else
(*lpfnEnumFont)( hdcCur, lpszFace, lpfn, (LPARAM)(LPVOID)&elf ) ;
#endif
```

End.

The Types of Fonts

Windows supports two broad categories of fonts, called "GDI fonts" and "device fonts."
The GDI fonts are stored in files on your hard disk; those that come with Windows are stored in the SYSTEM subdirectory of your Windows directory.

Many of these GDI font files, which are sometimes called "font resource files," have the extension .FON. Each file contains one or more complete fonts. These files are in the New Executable format, which you can verify by running Microsoft's EXEHDR or Borland's TDUMP on them. They are library modules, although somewhat unusual ones in that they contain no code or data. All they contain are two types of resources: a font directory and the fonts themselves.

TrueType fonts are also GDI fonts. They are stored in files with the extension .TTF. These files are not in the New Executable format. However, for each .TTF file there is a small .FOT file, which is a resource-only New Executable file that contains a font directory that references the corresponding .TTF file.

The second broad category of fonts, the device fonts, are internal to particular graphics output devices. For video display adapters, device fonts are currently rare. Windows uses the video adapter in graphics mode, so it must use the GDI fonts and write the pixels directly to the video display.

For printers, however, device fonts are very common. For instance, Windows can write text to a dot-matrix printer using either the printer's normal text mode or the printer's graphics mode. In text mode, Windows uses a device font and needs to send only the ASCII numbers of the characters out to the printer. In graphics mode, Windows can use a GDI font and must send the pixel patterns to the printer. For laser printers, device fonts can be stored in ROM within the printer or in ROM cartridges. If the printer requires a downloadable font that originates from a disk file, this font is also classified as a device font because it is specific to the particular device.

GDI fonts come in three flavorsraster fonts, stroke fonts, and TrueType fonts.
A raster font is sometimes called a bitmap font, because in a raster font file, each character is stored as a bitmap pixel pattern. Figure 14-1 shows a character from a GDI raster font, enlarged so that you can see the individual pixels.

Each raster font is designed for a specific aspect ratio and character size. Windows can create larger character sizes from GDI raster fonts by simply duplicating rows or columns of pixels. However, this can be done only in integral multiples and within certain limits. Right away, you can probably perceive one major difference between drawing othe
r graphics on the display and writing text to the display using the GDI raster fonts. Although you can draw a rectangle of virtually any size, GDI raster fonts are available only in discrete sizes. (You can't write text using a raster font smaller than the smallest raster font. If you want a GDI raster font of a specific size, that size may not be
available.) For this reason, GDI raster fonts are termed "nonscalable" fonts. They cannot be expanded or compressed to an arbitrary size. The primary advantages of raster fonts are performance (because they are very fast to display) and readability (because they have been hand-designed to be as legible as possible).

Prior to Windows 3.1, the only other GDI fonts supplied with Windows were the GDI stroke fonts. The stroke fonts are defined as a series of line segments in a "connect-the-dots" format. Stroke fonts are continuously scalable, which means that the same font can be used for graphics output devices of any resolution, and the fonts can be increased o
r decreased to any size. However, performance is poor, legibility suffers greatly at small sizes, and at large sizes the characters look decidedly weak because their strokes are single lines. Figure 14-2 shows a character from a blown-up GDI stroke font.

Stroke fonts are now sometimes called "plotter fonts" because they are particularly suitable for plotters but not for anything else.

For both GDI raster fonts and GDI stroke fonts, Windows can "synthesize" boldface, italics, underlining, and strikethroughs without storing separate fonts for each attribute. For italics, for instance, Windows simply shifts the upper part of the character to the right.

With Windows 3.1, modern outline font technology comes to Windows in the form of TrueType. A TrueType font uses straight lines and curves to define the outline of each character. These lines and curves are stored based on an arbitrary coordinate system. By simply scaling these coordinates, TrueType fonts can be continuously scaled and even rotated
.
When your program wants to use a TrueType font of a particular size, Windows "rasterizes" the font. This means that Windows scales the coordinates connecting the lines and curves of each character using "hints" that are included in the TrueType font file.

These hints compensate for rounding errors that would otherwise cause a resultant character to be unsightly. (For example, in some fonts the two legs of a capital H should be the same width. A blind scaling of the font could result in one leg being a pixel wider than the other. The hints prevent this from happening.) The resultant outline of each
character is then used to create a bitmap of the character. These bitmaps are cached in memory for future use.

Obviously, the rasterization of a TrueType font takes some time. However, once a TrueType font of a particular size has been rasterized and cached in memory, performance is as fast as using the GDI raster fonts.

Although the legibility of TrueType fonts is very good, they are not quite as readable as the hand-tuned raster fonts. However, italic and boldface versions of TrueType fonts are not synthesized as they are in the GDI raster fonts and GDI stroke fonts. Instead, separate TrueType font files provide italic and boldface versions. This is preferable b
ecause the italic versions of some fonts are quite different from the nonitalic versions.
The following table summarizes the characteristics of the GDI raster, TrueType, and stroke fonts:

GDI Font	**Performance**	**Appearance**	**Scalable**
Raster	The best	The best	Hardly
TrueType	Almost as good as raster	Almost as good as raster	Yes
Stroke	Poor	Poor	Yes

The GDI stroke fonts should be avoided except for use on plotters. For all but the most rudimentary applications involving fonts, the scalability of TrueType fonts greatly outweighs any disadvantages when compared with the GDI raster fonts. Because the TrueType fonts can be used on any type of graphics output device (except plotters), true WYSIWYG is possible. In the past, programs had to approximate printer output by using the GDI raster fonts. Now, TrueType fonts can be used on both devices.
Because device fonts are stored in printers and used in a device-specific manner, it is impossible to discuss them in the same detail as GDI fonts. Many modern laser printers (in particular, PostScript printers) support their own outline font technologies. Sometimes the device can italicize or boldface a device font, and sometimes it can't. You ca
n obtain such information from the GetDeviceCaps function using the TEXTCAPS index. If you want to obtain this information for particular printers, you can use the GetDeviceCaps function as shown in the DEVCAPS1 program in Chapter 11.

End.

How WaveMix Works

1. Mixing Sounds

The low-level multimedia wave out put function (waveOutWrite) does not support multiple simultaneous channels of output. So in order to accomplish this we must fake out the output device. This is done by premixing the wave output and then sending this new wave to the wave output device. The core concept which wavemix uses to mix files at run time is very simple. What it does is take a small slice off of each input wave file (each input is referred to as an "channel"), mix them together into an extra buffer which is the same length of the slice and then submit this wave buffer to the multimedia function waveOutWrite.

A digital wave is essentially an array of wave samples. At 11Khz 8bit Mono (which is used by the Wave Mixer) a wave array (or buffer) will contain 11025 byte elements per sec. of the wave duration. (i.e. a one second wave will contain 11025 data samples, a 1 minute data wave will contain 60*11025=661500 samples). To mix the waves in real time the input waves are summed together into another destination buffer. The destination buffer is typically a small size so that we can achieve real time results. A typical length is 1/8 th of a second = 11025/8=1378 samples.

e.g. Assuming that we have 4 waves playing and the slice length = 1378 samples. we could call a mixing routine: mixit(lpDest,rgpCDdata,4,1378);

```
void mixit(LPSAMPLE lpDest, LPSAMPLE rgWaveSrc[], int iNumWaves, WORD wLen)
{
    int i,iSum;
    WORD ctr;
    ctr = 0;
    while (wLen)
    {
        iSum=128; /* 128 is the "normal" value (silence) for 8 bit 11KHz */
        for (i=0;i<iNumWaves;i++)
            iSum = iSum + *(rgWaveSrc[i]+ctr) -128;
        PEG((int)0,iSum,(int)255);
        *lpDest++=iSum;
        ctr++;
        wLen--;
    }
}
```

The above routine will visit the i th element in each source array, sum them into a destination variable (iSum) that can accommodate any possible overflow (i.e. 8 bit elements are summed into a 16 bit destination) and then the destination variable is converted back to the original magnitude (saturation is done after the additions to minimize distortion).

The destination buffer can now be submitted to the wave output device (using waveOutWrite) and the user will hear all four sounds played simultaneously. When the wave output device completes playing the buffer it will return it to the client (WaveMix.DLL) with a request for more output data.

Since the wave output device can only play data that has been submitted to it and it takes a finite amount of time to mix the wave data, as soon as we submit the destination wave to the output device we must mix another buffer with the next slice of data and submit it to the wave output device to avoid an interruption in the audio output. This buffer will get queued by the device and

it will then play the data in it when it finishes playing the data in the first buffer.

While the output device is playing this second buffer we can mix the third slice into the first buffer and then submit it back to the device. We continue this "ping-pong" procedure of mixing the data and submitting the buffer to the wave output device until we have submitted all the wave data, at that point since we no longer submit data to the device it will stop playing. Note: In practice we often use more than two buffers (i.e. 3 or 4 is common) We then replace the ping-pong action with a "juggling" system:

The above text and diagrams describe the mixing process. There are, however, other situations which occur that greatly complicate the mixing process: Playing multiple files that are of uneven length, A request to start a new wave playing while others are already playing, A request to terminate a specific wave that is simultaneously playing with other wave files, A request to play multiple wave files and start them together. A request to pause the wave output, but not lose the current location.

Playing multiple files of uneven length:

The Wave Mixer has to take into account the possibility that the current slice will be shorter than the wave slice which it is attempting to fill. That is: one of the waves which it is mixing does not contain sufficient data to have a sample mixed into all the elements of the destination buffer. To handle the situation the wave mixer must first determine if a wave like this exists. If it does then it must determine how many samples it can mix from that wave. It will then mix from all the waves for that many samples. Then it will stop mixing that wave and mix the remaining waves for the second part of the destination buffer. Since it is possible that the same situation can occur again while mixing the remaining waves into the remaining part of the buffer it must repeat the above process again. This process will continue until the destination buffer gets completely filled up or their is no more wave data remaining to be mixed. At this point the destination buffer is submitted to the wave output device.

A request to start a new wave playing while others are already playing: "remixing"

A complex situation that the real time wave mixer must handle is a request to play a new wave file while there is currently other waves being played. The reason that this is difficult is that the wave mixer is actually mixing wave data a finite amount of time before the wave output that is currently being played by the wave driver. The wave mixer must determine the position at which the wave output device is actually playing wave data, "remix" the wave data to include the new wave, notify the output device that the data it is currently playing is no longer valid (waveOutReset( )) and submit new wave data to it that contains the new wave too. All of this must be done very quickly so that the user does not hear an interruption in the audio.

A second algorithm is also employed to achieve the above effect on hardware configurations where doing a waveOutReset can cause an audible click or cause the hardware to slow down. In these situations the wave mixer queues the new wave but does not interrupt the wave output device. When it mixes the next slice of data it will include the new wave also. This method can have a small delay which is the amount of time from which the wave data is submitted to when it is played: The driver must finish playing the current buffer, play all the previously queued buffers, and then play the new buffer. If there are three buffers being juggled with the wave output device the delay can be the amount of time required to play two to three buffers before the new wave data can be heard. If the wave buffers are very short then it is difficult but not impossible to hear the delay.

A request to terminate a specific wave that is simultaneously playing with other wave files:

Occasionally while playing multiple files the WaveMix.DLL will be requested to stop playing a wave on a particular channel before that wave has completed playing, i.e. "flush" a channel. This is handled in a similar manner to starting a new wave while others are playing. The wave mixer first determines the wave output position. It then removes the desired wave from the specified channel and "remixes" the remaining waves from the obtained output position.

End.

THE DEVELOPMENT
of
"FLAIR" for WINDOWS
A CASE STUDY

Overview

This paper outlines the processes which were used in the creation of this Multimedia CD-ROM. It examines the problems and discusses some interesting issues which were encountered at Microsoft. The lessons learned created a better understanding of some of the design and development issues facing a Windows multimedia developer today.

What is *FLAIR?*

Flair is the internal code name for the CD-ROM element of a Multimedia Publishers program. It's goal is to provide the MM developer community with valuable information on all aspects of Multimedia development under Microsoft Windows. Flair is bundled in the MM Publishers kit and is also available outside the program as a separate item. Flair was designed to show what could be achieved in the Windows environment by offering the user a new dimension in animated front-end design. The techniques to provide audio, video, morphing (a branch of video and graphics) and animating text with random special effects are intended to offer the user a different learning experience at every execution of the program.

One goal was to demonstrate that (given a suitably fast machine) Windows was indeed capable of offering a new and exciting environment for Multimedia development. This case study is only avenue employed to design and develop a CD-ROM and is not intended as a standard method. On the contrary, most developers will have their own in-house methods and practices. It highlights the steps within a Multimedia project which would be required, giving an idea of time scales for each of the elements along the way. This is valuable to someone starting out or for someone using new or familiar techniques.

The application, since it is not a product and not supported by Microsoft, is written to be as generic as possible, meaning each of the screen elements can be manipulated and customized through ".INI" text files by anyone. The source code and graphic designs can also be used to explore the techniques which are graphically displayed.

Flair consists of two basic windows, written and controlled by a C program shell, which launches different applications, such as Viewer applications, third party applications and video applications. Using the basic model, a large number of simple screens for both third parties and Microsoft were created in order to demonstrate other associated products and applications.

Rather than starting from the beginning at each stage, products and applications were already available with the added advantage that most of the products used from internal and third party sources were stable or in final beta-testing stage. The range of high quality multimedia tools for Windows development further eased the development process coupled with the rise in Interactive Design Houses offering their unique talents meant that a great deal of work which would have been required even 6 months earlier was either available or easy to find.

Planning, Designing, and Staffing

Initially there was one person assigned to this project. The project evolved from a series of test

programs which were used to try a development package from Pixar called Typestry. This allowed the user to create animated sequences of text. After a number of tests, theses sequences could be transformed from flick file format to the AVI format. Once in AVI, one could pull and stretch the animation as it ran. Once a number of these buttons were created, it was proposed that a control program be written to allow people interactivity through these AVI sequences. In other words, the buttons could be made to look interactive. This was the beginning of the project, and another member of the team, Steve Molstad, was assigned as programming lead. His goal was to wrap these AVI files into a usable program -- easier said than done. Initially, it was decided that the program's structure would be created under Viewer and have Viewer buttons call the AVI files that it generated. This, in fact, was impossible to achieve with any degree of usability. Perceived performance to the user was essential in keeping their interest. Any latency in the response of interactive buttons would pose an obstacle to the success of the program. Unfortunately, rather than having planned the program completely from the onset, it evolved through many different stages and further along in the project, any changes which were necessary became overwhelming obstacles.

Important Note! Plan all stages before beginning a project. The project was also a learning experience for all those involved in understanding the technology and defining a strategy to undertake the project and complete it in a timely manner. The aim of the interface was to show how AVI files could be used interactively. Another important area was its flexibility and almost every aspect of the C shell can be customized. .INI files have been provided to change buttons, associated audio files, morphs for those buttons, positioning, randomness, toggling between disabling and enabling the buttons and a host of other functions. Towards the end of the project, it was decided to provide a means of returning from one screen back to the previous screen. This proved impossible technically using the methods that had been employed in the overall design to that point. Had this function been defined earlier in the project, it may have been possible to integrate this function. Yet another reason for clearly specifying at each and every stage what function is desired.

The applications that were used also involved a steep learning curve and their idiosyncratic nature of the applications used made the project more of a challenge. For example, Photomorph was used in the creation of the morphing AVIs. When a button was pressed, a random morph appears which may change the button to a pattern or fade the button to the black background. In order to do this, several stages between using the program and integrating it into the design were necessary. The following steps, which were learned through stages of trial and error were subsequently taken to achieve the desired result. Using Typestry, the animated button was created. Before creating the animation, the first rendered version of that button was saved as a bit map to disk. Following this stage, the animated continued to be created. The animation had its own color palette which was an interesting problem in itself.

One of the fundamental problems that most developers face will be in the limitations of the handling of color maps. Hardware is limited by the number of colors that can be displayed by the accelerator cards which vary between 16 and 16,000,000 depending on what type of platform is used. This requires a fundamental design decision in which a choice must be made between what is desired and what is capable of being designed for a target market. The color map handling generally requires that the user allocate a certain number of system colors which are reserved for the system itself such as Window drawing menus, dialogue boxes, etc. Within this framework, the rest of the colors can be used as desired, however, when multiple programs or applications are running which also require the color map, there will be a conflict and color map flushing can occur. Designers need to try and avoid this at all costs. Because the sample front end required the simultaneous display of up to six buttons, which were all animated, it was necessary to paste the palette into the color map for each of the six buttons individually in order for them all to share a common palette. One of the buttons was left without having the palette applied so that this problem is easily demonstrated.

The front end was never intended to be anything more than a sample demonstration to determine

the limitations of Windows in the area of user interfaces. However, the sample code evolved into an attempt to use it as a navigator through the information on the CD. Since the program had been developed on the hard disk of a 486/66Mb PC, the real development did not begin in earnest until the first test disc was cut and the performance dropped to such a degree that a total restructure of the design was necessary. It was also necessary to run the demo disc from a system with a single speed CD-ROM drive with no software installed except for a basic Windows system. It was soon apparent that a redesign had to take place in order to increase the usability of the program. The reason for this was that the audio track could be associated with a video file. The problem while running from the CD-ROM was that there were two separate seeks, one for the audio track and one for the video file, which slowed the performance down significantly. By disabling the audio playback, performance increased to an acceptable level once again. However, since the program did not have a method of returning from the secondary screen after making an initial choice, it was decided that it was not suitable for use as a navigational tool for the CD-ROM. Therefore, it was left as a sample application which could be used to demonstrate various techniques of mixing audio, video, graphics, morphing and text. It is hoped that the program will help developers in creating interesting titles. If anything, it illustrates some of the basic techniques employed in creating nearly every title.

End.

Case Study of PLAYVFW.EXE

Planning ahead and not changing the spec

It is very important to fully spec out the program before you decide to proceed with the coding. It is even more important to stick to the spec once the coding has started. Makingchanges to the spec can lead to many unforeseen problems. It is important to have a good solid idea of what the program looks like and what the requirements are before starting in on coding. If possible a prototype of what you are doing is very helpful. Use a simple language (VB) or tool to mock up what the program will look like before going ahead with the actual coding. In this manner you can look at the design and determine if it is really what you want and if it is feasible. It is also important to think about the internals of the program. What structures should be used, what file formats, should I use child windows or owner draw buttons, etc. A well thought out program can help avoid the last moment rush of feature adds and unforeseen bugs caused by those feature adds.

Owner Draw Buttons

Looking back on the decision to use owner draw buttons, I would say, it wasn't such a good decision. The main problem is that you do not have control over the messages that are sent to the window. Therefore this causes problems when you want to receive a mouse click or mouse move message. To trap these messages (without subclassing) I used two messages that are sent from the child window to the parent window.

WM_PARENTNOTIFY
WM_SETCURSOR

WM_PARENTNOTIFY works fine as long as the only messages you want to see are WM_LBUTTONDOWN, WM_RBUTTONDOWN, and WM_MBUTTONDOWN. The WM_SETCURSOR message is sent to the parent window when the cursor is moved across thescreen. The WM_SETCURSOR message sends the handle to the child window that it iscurrently over to the parent window. In this manner you can tell if you are over a owner draw button or not.

Exec'ing Viewer

Originally I had used the winexec command to execute a Viewer application. This approach was very limited since I had no control over the application. I then decided to use the VwrCommand function to exec Viewer. This allowed me to be able to quit Viewer using the VwrQuit command and load another Viewer topic without loading another instance of Viewer.

MCI device ID storage in a structure

To avoid loading the AVI files each time I needed to play them back I pre-loaded the AVI files before the page was loaded. I also pre-loaded the AVI device driver since this causes some overhead when it is loaded for each AVI file. This increased the speed at which the AVI files would start to playback. Using the MCI commands for playback avoided the overhead of loading all the AVI files into memory. The MCI command interface streams the AVI data from the disk. The AVI device ID and all the individual AVI file device ID's were stored in a series of structures that represented each device. Since the amount of information for each device was small I stored these structures as global variables. To make the amount of ID's dynamic a linked list stored in global memory could have been used.

The structures were also re-used for the second page. When the second page was displayed

the old device ID was destroyed and a new device ID was created. This was done to conserve resources. Window handles were also re-used in the same manner. In this manner you limited the amount of windows and devices open at any given time.

Page information storage in a structure

To allow for a more dynamic way to display a page, the information could have been stored in a structure. This structure would allow for a certain amount of windows, devices and attributes of those devices. The structures could then be referenced in a linked list which represented the entire title. This linked list would grow or shrink depending on how many pages were in the title. The actual information contained in the list (pages) could be stored in a file. This page information could be read ahead and maybe some caching of sound files and AVI files could be done.

Using INI files

For the purposes of this program storing information in INI files worked fine. It is conceivable that you could have created a binary file format for storing the information. Along with this step a editor would be necessary so the user could change the information in the files. Although storing the information in ini files is very convenient since there are API's to take care of most of the work for you. Another step could be to create an editor which read the ini files and organized the data in some fashion as to make the editing of the files easier.

Displaying the image for the first two screens (RC vs reading from file)

The initial two screens display static bitmaps read from the resource file. This limits who can change these bitmaps to someone who could resource compile the file with different bitmaps. Changing the names of the bitmaps has to be done by actually editing the RC file of the project. With some additional coding this could be changed to allow for loading of bitmaps dynamically. I will leave this as an exercise to a future coder.

LOWRES vs HIRES

Since the user will be running this on different resolution monitors it is important to be able to determine what video mode you are running in. Something designed for 1024 X 768 will look considerably different on 640X480. It is possible that some of your images may not be on the screen at all. Therefore it is important to detect what video mode you are in and change the look appropriately. I took the approach of having two separate ini files that had lowres and highres information. When I detected the user was running in 640X480 mode using GetDeviceCaps, I read the ini file from the lowres directory. If they were running in 1024X768 I read the ini file from the program directory. I also did this for the other AVI files and controls by creating a lowres directory in the ini directory.

The Palette, how to playback AVI files simultaneously.

The secret is in using the videdit program. The idea is to combine all of your AVI clips (that appear on the same page) into one and then create an optimal palette. This palette you preserve as a file and paste into every AVI file that will appear on the same page.

How this is done. . .

First you load one of the AVI files into videdit. Run another copy of videdit and load a second AVI file. Mark in at the beginning and out at the end of the second AVI file. Select copy from the edit menu. Mark in and out at the end of the AVI file. Paste the second AVI file at the end of the first AVI file by first Marking in and out at the end of the first AVI file. Then choose paste from the edit menu. Repeat this process for all AVI files you want to display on the screen at the same time.

Now run the program paledit. In videdit go to the Video menu and select Create Palette. Click OK without changing any of the attributes in the dialog. If the amount of AVI information is very big you may want to scale size down by selecting convert frame rate from the Video menu. Set the frame rate to 1. This will considerably size down the information you are creating a palette for.

Once the palette is created a dialog box will appear, press OK. Activate the paledit program. Select paste from the edit menu. Save the palette from the file menu in paledit. Now you have a optimal palette which you can paste into each and every AVI video. To do this first run paledit. Open the optimal palette you created by choosing open from the file menu. Choose select all from the edit menu. Choose copy from the edit menu. Now run the Videdit program. Choose paste palette from the edit menu. Click OK, the Remap palette to best video colors should be checked and the all frames radio button should be selected.

End.

A Cool Rotating Parrot and Soldiers

The parrot3.exe and rotate3.exe files demonstrate a rotation effect on an image. This effect was done through special photography and the use of Video for Windows to seek to special

Run the Parrot program
Run the Soldiers program

README for new AVI Hotspot Stuff

Wayne Radinsky, Microsoft MMDRG, 2 November 1993

What's New?

The major new feature is the ability to see hotspots while the video is playing back while using Video for Windows 1.1. This is accomplished by means of a drawproc that draws red rectangles on top of the AVI. If you don't have VFW 1.1, you should get a MessageBox telling you it can't install its drawproc,but everything else should work the same as before.

The other new item is a hotspot VBX, which can fire VB events when hotspots are clicked. It also has the capability of making the hotspots visible with the drawproc, and your VB app can changethat on the fly. A simple VB app ("hottst") is included with the VBX.

The viewer DLL has also been modified to show hotspots. This can't be changed at run-time, however; if you don't want the hotspots toshow, use the old DLL.

Files

There are 3 subdirectories: Editor, Viewer, and VBX. If you're using Visual C++ you can just go into each dir and load the .MAK files (it will tell you that it's in a new directory). If you're using C7, go into the .MAK file and change the directory to your directory, then make a "Makefile" that includes the .MAK file. After that, you can use NMAKE to make the project, as usual.

There's one AVI file, ANIMAT.AVI, in the VBX directory. Copy it into the Viewer directory if you want to see the viewer demo.

BUGS

Video for Windows gives the drawproc a "relative" frame number, which the drawproc interprets as absolute -- what this means is that sometimes the hotspots don't go with the frame being shown, but with the beginning of the video instead. I originally thought this was an AVI bug, but when last I talked with the AVI folks they said it was intentional, so I need to add code that remembers what frame was seeked to every time a seek occurs, and makes that number available to the drawproc, which will add it to the frame number it gets from VFW.

End.

Visual Basic Calling Viewer Sample

Contained in the directory \develpmt\source\roadshow are the files used in the Multimedia Roadshow presentation. For those of you who didn't attend all is not lost. This example demonstrates how to call Viewer from a Visual Basic application. In addition to this it also demonstrates how to use the Motion Works utilities (AVI hotspot edtior and playback engine, Animation editor and playback engine) from Visual Basic. Excluded from this directory are the .\ bmp, .\midi, .\wav, .\avi directories used in the Viewer project and for the AVI button in the Visual Basic project. The files can be easily replaced by editing the commands in the Viewer and Visual Basic files.

About The Transparent BLT Routine

Draws an image with transparent portions. The transparent areas are a color in the source image.

This demonstrates the 'old' way and the new MM way to do transparent drawings.

You can find this sample application and source code in the \develpmt\source\transblt.

The Wavemix DLL Utility

The wavemix DLL is a utility that allows multiple wav files to be played simultaneously. It is designed to be as simple to use as possilbe but still have the power to do what is required by games. The DLL supports 8 channels of simultaneous wave play, the ability to queue up waves along the same channel and wave completion notification.

The wavemix DLL currently supports 11.025 Khz, 22.05 Khz and 44.1 Khz 8-bit Mono. Wave conversion from the file format is done automatically to the output sampling rate when the file is opened. The sampling rate is set in wavemix.ini. See wavemix.ini for details. Stereo files and 16 bit files are converted to 8-bit mono when the wave is loaded.

The API consists of the following functions:
HANDLE WINAPI WaveMixInit(void);
HANDLE WINAPI WaveMixActivate(HANDLE hMixSession, BOOL fActivate);
LPMIXWAVE WINAPI WaveMixOpenWave(HANDLE hMixSession, LPSTR szWaveFilename, HINSTANCE hInst, DWORD dwFlags); UINT WINAPI WaveMixOpenChannel(HANDLE hMixSession, int iChannel, DWORD dwFlags);
UINT WINAPI WaveMixPlay(LPMIXPLAYPARAMS lpMixPlayParams);
UINT WINAPI WaveMixFlushChannel(HANDLE hMixSession, int iChannel, DWORD dwFlags);
UINT WINAPI WaveMixCloseChannel(HANDLE hMixSession, int iChannel, DWORD dwFlags);
UINT WINAPI WaveMixFreeWave(LPMIXWAVE lpMixWave);
UINT WINAPI WaveMixCloseSession(HANDLE hMixSession); void WINAPI WaveMixPump(void);
WORD WINAPI WaveMixGetInfo(LPWAVEMIXINFO lpWaveMixInfo);

In order to play a file a program must:
1. Initialize the DLL with WaveMixInit() 2. Open a wave file with WaveMixOpenFile() 3. Open a channel with WaveMixOpenChannel() 4. Play the file using WaveMixPlay()

A given channel can be silenced by calling WaveMixFlushChannel() at any time.
When the program is done with playing sounds it should
1. Close the open channels using WaveMixCloseChannel()
2. Free the memmory for the waves using WaveMixFreeWave()
3. End the session with WaveMixCloseSession()

Note 1:
An appliction should call WaveMixActivate(FALSE) when it loses the focus so that the waveform device can be freed for other applications. When focus is regained the application can call WaveMixActivate(TRUE);

Note 2:
An application that does not release the processor can cause the sound to "skip" To avoid this it can call WaveMixPump() frequently.
WORD WINAPI WaveMixGetInfo(LPWAVEMIXINFO lpWaveMixInfo);
parameters
lpMixWaveMixInfo
a pointer to a WAVEMIXINFO structure that will get filled in by the WaveMix DLL:

```
typedef struct
{
        WORD wSize;
        BYTE bVersionMajor;
        BYTE bVersionMinor;
        char szDate[12]; /* Mmm dd yyyy */
DWORD dwFormats; /* see waveOutGetDevCaps (wavemix requires synchronous device)
```

*/ }
WAVEMIXINFO, *PWAVEMIXINFO, FAR * LPWAVEMIXINFO;

wSize: should be set to sizeof(WAVEMIXINFO) before calling WaveMixGetInfo.
bVersionMajor: the major version of the DLL.
bVersionMinor: the minor version of the DLL.
szDate: The compilation date of the DLL, Null terminated string in form Mmm dd yyyy
where Mmm is one of the following: "Jan","Feb","Mar","Apr","May","Jun","Jul","Aug", "Sep","Oct","Nov","Dec".
dwFormats: Specifies which standard formats are supported. The supported formats are specified with a logical OR of the flags listed in WAVEOUTCAPS documentation:
for version 1.0: the following wave formats are supported:

```
WAVE_FORMAT_1M08   /*  11.025 kHz, Mono,     8-bit   */
WAVE_FORMAT_1S08   /*  11.025 kHz, Stereo, 8-bit   */
WAVE_FORMAT_1M16   /*  11.025 kHz, Mono,     16-bit */
WAVE_FORMAT_1S16   /*  11.025 kHz, Stereo, 16-bit */
WAVE_FORMAT_2M08   /*  22.05   kHz, Mono,     8-bit   */
WAVE_FORMAT_2S08   /*  22.05   kHz, Stereo, 8-bit   */
WAVE_FORMAT_2M16   /*  22.05   kHz, Mono,     16-bit */
WAVE_FORMAT_2S16   /*  22.05   kHz, Stereo, 16-bit */
WAVE_FORMAT_4M08   /*  44.1     kHz, Mono,     8-bit   */
WAVE_FORMAT_4S08   /*  44.1     kHz, Stereo, 8-bit   */
WAVE_FORMAT_4M16   /*  44.1     kHz, Mono,     16-bit */
WAVE_FORMAT_4S16   /*  44.1     kHz, Stereo, 16-bit */
```

this is done by converting the wave format of the input wave to the current output format when the wave is opened.
return value
zero will be returned if the size fields match and the structure will be filled in.
if the size fields do not match the DLL will return the size that it expected. None of the fields in the structure will be filled in.

HANDLE WINAPI WaveMixInit(void);
This function should be called before any of the other functions. It will return a handle that should be used for subsequent API calls.
Parameters
none
Return Value
NULL will be returned if an error occurs ----------
UINT WINAPI WaveMixActivate(HANDLE hMixSession, BOOL fActivate);
This function should be called when an application loses the focus or otherwise becomes inactive. This will permit other applications to aquire the wave output device. Calling this function keeps all the channels open.

Note:

1. The wavemix DLL will keep sounds that were queued while the application was active in the queue. When wavemix play is subsequently reactivated the sounds will continue from where they were. (there will be a small amount of data loss equivalent to the amount of data that was buffered in the wave driver).

2. Calls to WaveMixPlay that are made while the application is not active will not be queued.

3. The Application can call WaveMixFlush if it does not wish the data to be pending while it is inactive.

Parameters
HANDLE hMixSession

handle that was returned by WaveMixInit()
BOOL fActivate
TRUE: The application is being activated and wishes to regain the wave output device
FALSE: The application is not active and wishes to be a good Windows Citizen and allow other applications to use the wave output device. This call will cause all the channels to be flushed.
Return Value
Returns zero if the function was successful. Otherwise, it returns an error number. Possible error returns are:
MMSYSERR_ALLOCATED
Specified resource is already allocated to a process. try again later.
MMSYSERR_NOMEM
Unable to allocate or lock memory.
MMSYSERR_ERROR
an internal error ----------
LPMIXWAVE WINAPI WaveMixOpenWave(HANDLE hMixSession, LPSTR szWaveFilename, HINSTANCE hInst, DWORD dwFlags); Parameters
HANDLE hMixSession
handle that was returned by WaveMixInit()
LPSTR szWaveFileName
szWaveFileName can be the name of a wave file to open, or the name of a WAVE resource to open, or an integer ID of a WAVE resource to open. See the dwFlags parameter for details
HINSTANCE hInst
Identifies the instance of the module whose executable file contains the resource. See the dwFlags parameter for details.
DWORD dwFlags
WMIX_FILE: if this bit is set then szWaveFileName specifies a far pointer to a string containing the filename of the file to open. hInst is ignored.
The MS-DOS filename should not be longer than 128 bytes, including the terminating NULL.
Currently the DLL will only permit 11Khz 8-bit mono files to be opened.
This flag should not be set with WMIX_RESOURCE
WMIX_RESOURCE: if this bit is set then szWaveFileName specifies a WAVE resource to be opened in hInst.

If the high-order word of the szWaveFileName is zero, the low-order word specifies the integer identifier of the name or type of the given resource. Otherwise, the parameter is a long pointers to a null-terminated string. If the first character of the string is a pound sign (#), the remaining characters represent a decimal number that specifies the integer identifier of the resource's name or type. For example, the string #258 represents the integer ID 258.

To embed a wave file in a resource use the following format in your .rc file:
GameSound WAVE gamesnd.wav

Note: to reduce the amount of memory required for the resources used by an application, the application should refer to the resources by integer identifier instead of by name.
Return Value
NULL will be returned if the dll was unable to open the file or resource. ----------
UINT WINAPI WaveMixOpenChannel(HANDLE hMixSession, int iChannel, DWORD dwFlags);
Parameters
HANDLE hMixSession
handle that was returned by WaveMixInit()
int iChannel
Specifies a number which indicates which channel should be opened. Currently the DLL supports channels 0 though 7.
It is not necessary to open or close them in any particular order.
DWORD dwFlags

WMIX_OPENSINGLE: i Channel specifies the single channel to be opened.
WMIX_ALL: all the available channels will be opened. iChannel is ignored.
WMIX_OPENCOUNT: i Channel specifies the number of channels to open.
eg. if iChannel==4 then channels 0 through 3 will be opend.
Return Value
Returns zero if the function was successful. Otherwise, it returns an error
number. Possible error returns are:
MMSYSERR_INVALHANDLE
if an invalid channel (ie > 7) was specified
MMSYSERR_BADDEVICEID
Specified device ID is out of range.

MMSYSERR_ALLOCATED
Specified resource is already allocated. Or the channel has already been opened.
MMSYSERR_NOMEM
Unable to allocate or lock memory. WAVERR_BADFORMAT
Attempted to open with an unsupported wave format.
WAVERR_SYNC
Attempted to open a synchronous driver without specifying the WAVE_ALLOWSYNC flag.
MMSYSERR_ERROR
an internal error ----------
UINT WINAPI WaveMixPlay(LPMIXPLAYPARAMS lpMixPlayParams);
parameters
lpMixPlayParams
a pointer to a MIXPLAYPARAMS structure:

```
typedef struct
{
        WORD wSize;
        HANDLE hMixSession;
        int iChannel;
        LPMIXWAVE lpMixWave;
        HWND hWndNotify;
        DWORD dwFlags;
        WORD wLoops;   /* 0xFFFF means loop forever */
}
```

MIXPLAYPARAMS, * PMIXPLAYPARAM, FAR * LPMIXPLAYPARAMS;
wSize: should be set to sizeof(MIXPLAYPARAMS)
hMixSession: the handle that was returned by WaveMixInit() iChannel: the channel on which the wave should be played. lpMixWave: a wave which was opened using WaveMixOpenWave()
hWndNotify: a window handle to receive the MM_WOM_DONE message
when the wave completes. If this value is set to NULL then
the message will not be posted.
dwFlags: WMIX_QUEUEWAVE: the wave will be placed on the specified channel
and played after all waves which are currently waiting to play on that channel.
WMIX_CLEARQUEUE:this wave will preempt all waves currently playing
on the specified channel. Notification messages will not be sent for any waves that get dumped.
This message should not be combined with WMIX_QUEUEWAVE
WMIX_HIGHPRIORITY: Play this way immediately. This flag will
interrupt
the data buffered in the wave driver and remix the sound. If this flag is not set you could
experience up to a half
second delay before sound is played.
Note: if WMIX_QUEUEWAVE is set with this flag then a sound playing on the channel will not be prempted, but it will begin immediately after the previous sound finishes. If no sound is currently playing on this channel then a remix will occur.
WMIX_USELRUCHANNEL: the wave should be played on any available channel

or played on the channel that was least recently used. This flag should be combined with WMIX_QUEUEWAVE or WMIX_CLEARQUEUE.
WMIX_WAIT: setting this flag cause waveMixPlay to put the play information on a "waiting list" to play. when waveMixPlay is called without this flag set it will process the calls in the order they were received. This is useful if you want to play multiple sounds simultaneously.
Note 1: This flag is not a 'pause'. Waves that are playing will continue to play regardless of how many files are on the wait list.
Note 2: Since the waves that are submitted with this flag set are not checked until waveMixPlay is called without the flag set you should
be careful about using other API calls before playing the sounds. eg. WaveMixFlushChannel and WaveMixCloseChannel do not process the wait list.
wLoops: The number of times the wave should be repeated. If dwLoops is set to 0xFFFF the wave will loop until the channel it is on is flushed, preempted,
or closed.
Return Value
Returns zero if the function was successful. Otherwise, it returns an error number. Possible error returns are:
MMSYSERR_INVALHANDLE
The specified channel has not been opened or invalid lpMixWave was passed in
MMSYSERR_NOMEM
The dll has run out of internal memory to queue up waves. Wait until some sounds complete and then try again.

UINT WINAPI WaveMixFlushChannel(HANDLE hMixSession, int iChannel, DWORD dwFlags);
This function will empty the queue of any waves waiting to play on this channel.
This function can be called to stop a wave that is playing on the channel without affecting any of the other channels.
Parameters
HANDLE hMixSession
The handle that was returned by WaveMixInit()
int iChannel
an integer which specifies a previously opened channel
DWORD dwFlags
WMIX_ALL: causes all the channels to be flushed. iChannel is ignored.
WMIX_NOREMIX: prevents WaveMixFlushChannel from causing the data to be immediately remixed. This is useful if you want to flush more than one channel, or if you will be calling WaveMixPlay with a HIPRIORITY wave following WaveMixFlushChannel
Return Value
Returns zero if the function was successful. Otherwise, it returns an error number. Possible error returns are:
MMSYSERR_INVALHANDLE
The specified channel was not open
MMSYSERR_INVALPARAM
One of the parameters passed to the function is not valid ---
UINT WINAPI WaveMixCloseChannel(HANDLE hMixSession, int iChannel, DWORD dwFlags);
This function will flush and close the specified channel.
Parameters
HANDLE hMixSession
The handle that was returned by WaveMixInit()
int iChannel
an integer which specifies a previously opened channel
DWORD dwFlags
WMIX_ALL: causes all the channels to be flushed. iChannel is ignored.
Return Value
Returns zero if the function was successful. Otherwise, it returns an error number. Possible error returns are:

MMSYSERR_INVALHANDLE
The specified channel was not open ----------
UINT WINAPI WaveMixFreeWave(HANDLE hMixSession, LPMIXWAVE lpMixWave);
Parameters
HANDLE hMixSession
The handle that was returned by WaveMixInit()
lpMixWave
A pointer that was created by WaveMixOpenWave()
Return Value
Returns zero if the function was successful. Otherwise, it returns an error
number. Possible error returns are:
MMSYSERR_INVALHANDLE
The given pointer was not valid

UINT WINAPI WaveMixCloseSession(HANDLE hMixSession);
This function pairs up with WaveMixInit(). It should be called before the application terminates.
Parameters
HANDLE hMixSession
The handle that was returned by WaveMixInit()
Return Value
Returns zero if the function was successful. Otherwise, it returns an error number. Possible error returns are:
MMSYSERR_INVALHANDLE
The given handle was not a valid session ----------void
WINAPI WaveMixPump(void);
Calling this function causes the WaveMix DLL to mix the next slice of wave data and submit it to the wave drivers. This function can be called if the application does a lot of processing and fails to allow messages to reach the WaveMix DLL in time to avoid drying up the wave queue.
Parameters
none
Return Value
none ----------

End.

The MSCDEX Document Files

The MSCDEX document files are contained in the jumpstart js2.mvb file. Run go.exe and search for the name of the article. You can also find a copy of the document in \development\tech_art\ cdrom. The following listing of articles is provided so that you know what articles were included with the MSCDEX disks.

CDROMIFY.DOC/TXT
Contains information on creating CD-ROM media for the MS-DOS CD-ROM file systems. It also has information on creating MS-DOS CD-ROM media readable on CD-ROM file systems provided for other operating systems.

CONTENTS.DOC/TXT
Contains infomation on the contents of the documentation and disk files.

COVER.DOC/TXT
Title page for documentation.

DEVICE.DOC/TXT
Describes the interface for a MSCDEX device driver.

EXAMPLE.DOC/TXT
Describes the example HITACHI device driver.

INSTALL.DOC/TXT
Describes how to install MSCDEX.EXE and the device driver without the SETUP program.

KANJI.DOC/TXT
Describes the Kanji CD-ROM disc support in MSCDEX.EXE.

MCTRL.DOC/TXT
Describes the programming interface for applications using MSCDEX.EXE.

MSDOSIFY.DOC/TXT
Contains information on creating CD-ROM media for other CD-ROM file systems so that it will also be readable on the MS-DOS CD-ROM file system.

NETINFO.DOC/TXT
Information about sharing CD-ROM devices on a network.

OVERVIEW.DOC/TXT
Describes the version 2.21 product release.

QNA.DOC/TXT
Contains answers to common questions about device drivers and MSCDEX.

CDSPEED.DOC/TXT
Describes the transfer rate test programs.

TESTDRV.DOC/TXT
Describes the automated device driver test program. It also gives information on testing CD-ROM drives for Windows compatibility.

End.

The Convert Program

The Convert program allows you to convert from numerous data types to numerous other data types. It also allows you to batch up a number of files to be converted and then converts them into the selected file type.

The Convert program is located in the \develpmt\tools\convert directory.

UpFront Overview

UpFront is a program that floats ontop of all other programs that allows you to quickly access other files and utilities without having to minimize or toggle away from your current program.

It is recommended that you place UpFront in your startup group, so it will always be available when are running Windows.

UpFront is actually two floating programs in one. The first part of the program has ten buttons that are always on the screen. These are called top level buttons. You can configure up to eight of these buttons for your own use. The second part of the program has 21 icons that you can assign programs to. To access the second part of the program choose the control menu (the minus sign.) For more information see Adding Your Own Programs and Buttons

UpFront also comes with a Create Directory utility that allows you to quickly create a directory without having to use File manager or shelling out to the MS-DOS prompt.
To use UpFront simply single click the button you wish to run with the mouse.
To find out how to configure UpFront with your own programs see Adding Your Own Program and Buttons.

For information on other UpFront features choose the Contents button and select the appropriate topic.

TouchSend Consulting Services
904-668-6180 Compuserve 76064,3410

TouchSend specializes in the following:

1. Training developers/authors in innovative ways to use Viewer 2.0 to create satisfying titles, including electronic publishing, textbooks, catalogues, interactive demonstrations, tutorials, corporate policies & procedures, statutes & regulations, and educational material.

2. Developing migration and conversion strategies for existing hard copy and electronic media. For example, converting existing textbooks from desktop publishing format to Viewer format on an automated cost effective basis or converting banking/insurance corporate policy/procedure manuals to electronic format.

3. Designing databases that will automatically create, maintain and update material as well as autogenerate and update hypertext links and jumps.

4. Custom programming of Microsoft Multimedia Viewer extensions (such as the TouchSend Tools) and interfacing of Viewer data with other software (both Windows and non-Windows).

5. Developing corporate strategies for electronic publishing, electronic sales and presentation tools, as well as kiosks, electronic catalogues and CD Rom Development.

To run TouchSend's excellent demo, you can access it off the Jumpstart CD directory in \ develpmt\s_devtls. Have fun with it!

Graphic Evidence

A Division of LSI

You can find samples of Victors work in \samp_med\graphics\v-osaka. Run the dispbmp.exe program to see all his work in a full screen slide show.

Please peruse the most excellent bitmap images created by Victor Osaka. Victor's talent is self-explanatory once you view these images.

His services may be contracted out by contacting him at:

Graphic Evidence
200 Corporate Point, Suite 300
Culver City, CA 90230
310 568 1831
310 568 1861 fax

LENNY's MusicToons

Demo Version 1.0

Please Note !!
This is a demo version only !
Most games have been disabled !

To run Lenny's demo exit this application and run it from the CDROM. It is located in the \ Samp_Aps\musicpn directory.

Index to Topics :

HARDWARE & SOFTWARE REQUIREMENTS

Platform:

- MPC Multimedia Computer

Computer:

- IBM 386, 486 or higher, and other 100% compatible
- PC systems
- Minimum 20 MHz CPU, 33 MHz & higher recommended

Memory:

- 4 megs or more

Video:

- VGA 256 colors 640 x 480, fast video recommended

Operating System:

- Dos 3.3 or later

- Windows 3.1 or later

Sound:

- Any MPC compatible sound card, MIDI & WAV file capable, such as:
- Sound Blaster SB16 Sound Card. (Use optional 'Midikit' for external synths.)
- Sound Blaster Pro Sound Card. (Use optional 'Midikit' for external synths.)
- External synth such as the Roland Sound Canvas
- synthesizer for better sound quality. All MIDI
- files use General MIDI patching for instrument
- selection.

Other:

- Stereo sound amplification or headphones
- Mouse

MORE ON SOUND

Volumes: Wav & MIDI
The Windows Midi Mapper

VOLUMES: WAV & MIDI

Note: Volume between the 2 different types of sound sources (i.e. MIDI & WAV) are controlled by a mixer utility that comes with your sound card. Slider controls will adjust the volumes. Check your manual for the mixer utility and how it is used.

The Windows MIDI MAPPER

Other Synths & the Windows MIDI Mapper Program:

- Check your MIDI Mapper in Windows (use the Control Panel icon) to check the patch mapping in your system. Since the midi files in Lenny are General MIDI patches, and if your synth is set up for General MIDI patching, set the patch route for each channel to "None" using the Midi Mapper.

  On the other hand, if your synth is not set up to receive general MIDI, you will need to reroute instrument patches with a new MIDI map file. If "wrong pitches" occur it is likely that channel 10 should have it's destination set to channel 16 in the MIDI Mapper. See below for more information.

TROUBLE SHOOTING

RAM MEMORY

Lenny's MusicToons requires a minimum of 4 megabytes of random access memory (RAM) to

operate. Check that available memory is adequate. Although Windows allows you to run more than one application simultaneously, it is highly recommended that you close everything else to release all available memory when you run Lenny.

VIDEO DISPLAY DRIVERS

Lenny requires 256 colors in a screen resolution of 640 x 480. When you install Lenny's MusicToons, Lenny will tell you if you have the correct video driver or not. This mode is available in VGA cards which have 512K video RAM or higher. To install the required video driver use the "Windows Setup" icon (setup.exe) and take the following steps (note: setup.exe can also be accessed from DOS when in the windows directory) :

1. Locate your video card manufacturer's display drivers disk.
2. Find your Windows disks.
3. Find the Windows Setup Icon (setup.exe).
4. Select Change Settings from the Options menu.
5. Tn the pop-up menu, locate the Other Drivers selection.
6. Specify the drive and/or directory of the video display drivers disk.
7. Select the driver which supports 640 x 480 in 256 colors and click OK.
8. You may need to insert some of the Windows disks for fonts, etc.
9. To change to any other video mode, simply follow steps 1 through 8 for whatever mode required.
10. Once the above steps are completed, restart Windows.

Note that current drivers may be obtained directly from the video card manufacturer or from bulletin boards that are often setup by the video board manufacturers. Bulletin boards are databases that are accessible via a modem.

SOUND DRIVERS for your sound card

Sound drivers are not people who deliver musical equipment to you from your local music store. Hardware/sound drivers are special pieces of software which allow Windows to "communicate" and control specific pieces of hardware in your computer. These specific pieces of hardware are the cards in your computer which give the computer its sound making & listening capabilities. There are a great number of manufacturers of MPC sound cards of varying qualities. There are 2 major sound drivers that must work i.e. FM/MIDI and WAV. If Lenny reports that any of your FM/MIDI or WAV drivers are not correct, then try the following:

Locate your manufacturer's Driver disk for whatever card needs software installation. Also check your manual for any information regarding installation. Usually there is a set procedure for installation and most of the steps are done for you. Follow the manufacturer's procedures for installation which usually works.

However if the manufacturer's installation doesn't work or is incomplete, the following describes a more "manual" installation process:

1. Keep your Windows 3.1 disks handy. They're often needed for hardware installations.
2. Find the Windows "Control Panel".
3. Find the Drivers Icon.
4. Find the Add button and click on it.
5. Find and select "Other".
6. Insert your manufacturer's Driver disk.
7. If necessary, Delete drivers to clear out any unecessary drivers once the above steps are completed, restart Windows.
9. Try installing Lenny's MusicToons again or try the Sound Test to see if there is any improvement.

THE MIDI MAPPER

The MIDI Mapper utility is a program which translates MIDI data in specific ways. The program is called a mapper because it literally remaps different classes of midi data into new values. The types of MIDI data remapped or reassigned new values are Patch, Channel, and Key information.

For the most part, the MIDI data that Lenny provides for the sound card's on board synthesizer should require little or no adjustments with the Midi Mapper. The "Name" should have the card's synth map file for the "All FM". The most common adjustment needed is the reassignment of channel 10 which holds the percussion part due to variations in sound card synths. The symptom of this error is when a MIDI file sounds as if it is playing many "wrong notes". These "wrong notes" are supposed to be percussion sounds and if the synthesizer doesn't understand these sounds in channel 10, it should understand them in channel 16. If correct, then the "wrong notes" will now sound roughly like percussion instruments. To make this adjustment do the following:

1. Go to the Control Panel.
2. Click on the Midi Mapper icon.
3. Click on Setups.
4. Click on Edit. The 16 channel display of Source, Destination, Port Name, and Patch Map Name will appear.
5. Find channel Source 10.
6. In the Destination column, change the value to 16.
7. Return to Lenny's MusicToons or Lenny's Sound Test to play different MIDI files.

TECH SUPPORT

Have information about your computer ready. Try to be next to your computer when making the call to tech support.

Paramount Product Support Service : (317) 843-5111

ACKNOWLEDGEMENTS & TRADEMARKS

Windows 3.1, and DOS are products of Microsoft Corp.
Sound Blaster Pro, SB16 are products of Creative Labs Inc.
Roland Sound Canvas is a product of Roland Corp US.

End.

The Microsoft
Multimedia Developer Relations Group
Fall '93 JumpStart
Road Show

If you missed it all is not lost! In this document you will find a reference to our PowerPoint slides. Just click on the hotspot and you will be able to browse through all the slides we used on the Road Show.

What was the Road Show you ask?
The Road Show provided information regarding how to take information or an idea you have for a Multimedia title and bring it to market. We provided information regarding media types, tools to manipulate the media and actual code in Visual Basic and Viewer to pull all the media together into a great title.

View the Slides

About the Animation Directory

The animation directory contains a series of animations in flc and awa format. You will need the appropriate playback engine to play these files. To open
the flc file use the VidEdit program contained in the \development\sdk\vfw11.win.

About the Audio Directory

The \Sample_Aps\audio directory contains a sampling of audio clips ranging from 16 khz, 44 bit, stereo, to 22 khz, 8 bit, mono.

They can be accessed directly from the CDROM.

About the Video Directory

The video directory contains video clips that you can load after installing video for windows. To install video for windows go to the \development\ SDK directory for the setup files.

Interactive Multimedia Development Guide

by
Frédéric De Wulf

December 2, 1993

Introduction

The development process of a multimedia program parallels the building construction model in many ways. To a large degree most multimedia applications are a structure in which your pore content. Because the ultimate user will be free to wonder through this structure as he/she pleases, one must take particular care in creating an environment that is easy to use, encourages exploration, makes information available in an effortless intelligent way, engages one aesthetically and emotionally.

As in construction, costs vary greatly depending of the type of the project. The cost of building a house is insignificant compared to the costs of building a warehouse, which in turn is dwarfed by the cost of constructing a 40 story skyscraper. The budgeting and scheduling pitfalls are the same: the later your change the design in the development process, the more costly the changes will be. Sure you can add an elevator shaft, but after building five of the first 20 floors, its gonna cost ya!

Labor often represents 90% of your costs. Time is of the essence! Overall, one of the most important decision in the development process is the careful selection of a good team. You need a team of people working in a very strong collaborative spirit to accomplish this. This team will work very closely from the very first concept meeting to the final delivery of the finished products. An author approach to multimedia can often lead to disaster.

Like construction, you need an architect to design the structure, an engineer to consult on and implement the technical aspects of the project, an interior decorator to create an engaging environment, a project director to watch over the budget and time frame as well as insure smooth communications between members of the team. In addition you will need a number of specialists depending of the demands of the project.

You will start with a sketch of the project, then create a model. You will then elaborate very detailed blue prints identifying everything single aspects of the project. Then, the production process executes these plans and blue prints to create the final product. Finally you put this product through the most rigorous testing procedure to insure its survival in the hand of the toughest judge of all: the consumer.

Process

Probably the most important aspect of the process is the team's self discipline in respecting the approval process. After approving something, DO NOT CHANGE IT, unless a focus group test indicates a desperate need for change. Once of the evils of multimedia is what is know as 'creeping elegance', it can double or triple a projects budget and time frame if left to rampage unchecked. For this purpose, every time you see the word Ö **APPROVE** in this document, it indicates a point where all team members agree and physically sign off on a deliverable.

Phase I: Analysis

The analysis phase is critical and, for the most part, very revealing. This is when your put your

concept through the fire by making sure it is viable from a business and marketing perspective. It also helps you focus your concept, budget and time frame and flags up-front critical issues.

Title & audience definition

What is the general concept and purpose of the title?

Who is going to buy it and why?

What do you want the audience to do/feel/experience while going through the title?

What is the life of your title: timeless, seasonal or time-sensitive?

Market analysis & business case

How large is the audience segment you seek to reach?

How many of them own multimedia playback capable machines now, a year from now, two years from now?

Which are competing titles? Have you looked at them?

What are typical retail and wholesale prices for title in the same genre as yours?

What is the typical sell through you can conservatively expect from a title in your market segment?

How much will product marketing cost you?

From the above, estimate a conservative break-even figure and profit margin you would be comfortable living with and identify a production budget range target. (see budget section for more details)

Investigate critical issues

Rights acquisition

Who owns the rights to the material you are likely to use?

What potential costs / royalty negotiations issues are you going to run into?

Distribution

How is it going to be packaged and distributed? Through which channels?

Platforms

What platforms is it going to be distributed for?

In how many different mediums and formats?

Cachet

Does your title incorporate brand name or brand product with an established national recognition?

Team Selection & Kickoff

Carefully select your team (see team section for more details)

Hold a kickoff meeting outlining key objectives with clear priority and . For example:

Budget not to exceed $300,000

Must be done within six months

Must be the best entertaining first then educational children's interactive storybook ever produced

Must run completely off the CD-ROM under Windows with a minimum of 16 colors and 8 bit audio

Must include animation of at least 15 fps

Set target milestone and budget targets for each subsequent phases

Phase II: Design

This is one of the more important aspects of title development. Thorough design and planning are critical to the successful production of a multimedia title. This is the time to be creative and try out ideas, designs and technical ideas. Its also the time to plan to the highest degree of detail and to check, double-check, triple-check and check again these plans.

High Level design

Define high level flow chart of the title

Identify the major areas of the title

Define to basic architecture

Draw it out on a on page flow chart

Ö **APPROVE** high level flowchart

Define the look of the title

Sketch pencil thumbnails options of each major screens defined in the high level flow chart

Sketch options for buttons, icons and other navigational graphics

Select best of each and 'flesh out' color version in actual target computer medium with correct aspect ratio and pixel resolution

Ö **APPROVE** final key frames

Preliminary disc space estimate

This must be done early to insure that the project will not unexpectedly require more than a single disc

Roughly estimate the amount of space which the content will require.

Prototype

The prototype is the only time before beta release that everyone will get a feel for the interactive interface and the look and feel of the title. It is critical to the success of a good interactive title. It takes the high level design and makes it come to life on the target platform. It emulates everything major pathways of the interactive design, going down each branch once (not adding content depth). It plays the same role as a building model from both n evaluation standpoint and a marketing one. It also keeps top management off your back while you produce the title.

Produce prototype audio

Its important to produce the audio professionally rather than have an assistant record a voice. Your want the prototype to be as close to the final project as possible.

Narration
write rough narration track
audition and select professional talents (try male and female voices)
Record script in professional studio with a variety of intonations

Music
Select a number of stock pieces of music and/or sound effects
If using professional composer, have him/her produce a scratch track of several music themes

Encode the audio as you would in the production phase

Prepare sample stills / video / animation

Identify existing footage or animation which subject represents as closely as possible your ultimate content

For animation, animatics (rough key frames of an animation) are often sufficient

Encode the video / animation as you would in the production phase

Prepare text and graphics

Your graphics should have been done in the KEY FRAMES process

Scan or type sample text content

Program prototype

Use highest level programming tools available to avoid time consuming development or duplication of efforts in the production phase

Programs like Viewer, Visual Basic, Authorware and MacroMind director are ideal

Even though the prototype may not emulate the final program speed, its important that it emulate as closely as possible the navigation and interface feel of the final product.

Conduct a focus group testing of the prototype

Select representative small sample of your target audience and let them play with the prototype. 8-10 people are sufficient (qualifying its produced state beforehand of course)

Videotape their experience. Have the design team watch them experience the product in real time. (its a rare opportunity for you to see first hand reaction to the design)

If feedback suggests dramatic re-design, go back to the high level design drawing board. Its much cheaper and easier to do it now than once the product is on the market.

Ö **APPROVE** prototype

Detail Design Development

Its now time to take the high level design and develop it to its most minute detail. Part of this process is also to test technical assumptions and production pathways, and most importantly, plan the production phase to its most minute detail.

Content research

Engage a researcher(s) to dig out all content options for the title

organize and categorize the results according to the high level design. Developing a research database at this time is strongly advised

Identify 60-80% if not all content items. (not actual images, but at least descriptions and whereabouts

Build detail flowchart and detail design document

Develop complete flowcharts drawings.
Expand high level flowchart into a detailed flowchart from research outline and script draft

Be sure to develop complete drawn out copy of flowchart identifying every single pathways, branching, content description and script draft.

Produce Detail Design Document
Identify all branch coding and labeling standards (should be done with the core team)
Develop an asset and production database. Its very desirable to develop a series of related databases which encompass the entire production process from beginning to end: research, acquisition, creation and tracking of all digital assets through each phase of development, etc...

Populate database with filenames and location for each assets, making sure to identify forward and backwards branching and good description of the assets.

Script draft

From treatment outlines, draft a rough script. Its very desirable to treat each script component as an individual asset and input it directly into the assets database.

Fact proof and spell check flowchart & detail design document

Do it once

Do it again

And once more

and be prepared to have this proofing process continue throughout the project

Ö APPROVE **detailed design flowchart & document**

Rights negotiations and footage acquisition begins

Initiate right negotiations for large content source

This process can take forever and if not initiated immediately can jeopardize the delivery of the project and weaken your negotiating strength as you near completion of project. Be sure to get any verbal agreements in writing.

Develop rights / content acquisition relational database

Identify major source for your content (images, sound, text)

Clearly identify the kind of right you need and in what order you are willing to give up depending on price. i.e.:

Start with all interactive rights in perpetuity, in any medium whether know or unknown, throughout the universe (very tough to get)

Middle of the road: Interactive rights in perpetuity for a specific platform, a specific title and specific market and territory(very possible to get)

Lowest: right for initial press run with locked in amount for subsequent press runs

Engage in rights negotiations for bulk acquisitions with tiered bench prices depending on quantity acquired.

Acquisition of screening materials

You usually first order screening tape (usually low quality or copy protected)

Before you buys any footage, one must sift through the footage and identify likely options. Hold off final purchase until product is completed. You usually are not completely sure of the final footage count until the final product master is shipping.

Screening and logging of all acquired footage

Its critical to keep excellent records on all incoming elements. A lost original such as a simple slide cost alone $2,000 in fees

Technical planning & testing

Identify, test, and document production pathways

There are many ways to produce content for multimedia titles. Just the list of tools and various file formats alone can boggle the mind. The technical producer must select and thoroughly test the pathways to produce:

Graphics: from slides or print to computer images, icon templates, button highlights, partial screen animation, color palette production standards, etc..

Sound: final analog master tape format, digitizing tools, audio quality standards for the product, etc.

Video: playback frame rate & image size, capture and encoding pathway, etc.

Text: scanning or typing, system font use, anti-aliasing issues, etc.

File conversion: Converting and transporting asset files from one platform to the other, from one format to other, from one computer to the other (network or drive)

Backup and archiving: how often, how, on what medium

Future technology or platform provisions: Scan all assets at highest possible resolution to archive them for future potentially higher quality platforms

Document each pathways: Step by step instructions for each pathway should be written and available to any project member. (on-line access would be ideal)

Design, develop and implement tracking databases

As mentioned above, the Technical producer and programmers will work very closely with other development staff to create a database system to track:

Assets Acquisition

Title detailed flowchart

Tracking of assets creating and versioning

Final assets filename, directory pathway, branching parameters, screen pixel coordinates, etc.

Audio & video digitizing logs

Define file naming convention and directory structure

Multimedia usually implies an enormous amount of media store in thousands of individual files incorporated into the final project, and many more representing building blocks to arrive to the final assets files. One must establish strict guidelines for naming these files to be able to identify them easily.

Establish file extension standards to be applied from the time of file creation

Establish directory structure which will easily enable programmers to locate assets according to the flowchart structure. (sometimes the directory structure is a mirror image of the detail flowcharts)

Develop and test uncharted Software Engineering issues

Most multimedia titles push the Software Engineering envelope by requiring unique features for which code has never been written or is not readily available. To verify that these unique design features can be executed, these features must be written and tested at least in principle.

Playback or platform related technical issues: fastest animation playback, tricky audio interleaving techniques

A special text scrolling technique, or unique search engine

Final Parts count

The detail design outlines every potential assets which will be included in the title. Every assets, (graphics, icons, buttons, audio minutes, video minutes, text characters) must be counted

If the database is set-up properly, this can be an easy task.

Final disc space estimate

CD-ROM applications require careful space planning. Even though a single disc can store in upwards of 650 Megabytes of information, multimedia assets are notorious for being space mongers.

The complete parts count must translated into bytes according to the appropriate compression algorithms which are going to be used.
A rough application code size must be allocated
Audio interleaving techniques must be taken into account
One must allow 50 to 60 megabytes of room on the disc as contingency

Ö APPROVE **pathways, databases, file naming convention, Parts count and disc space**

estimate

Create project bible

Its a good idea to create a bible outlining all project standards for anyone to consult at their leisure

Again a well design relational database system pays off

Budget and schedule development

This is an important phase of the entire project and plenty of time must be allowed to get it accomplished. Each team member must carefully breakdown the detail design document and parts count into individual tasks at the lowest reasonable level. They must then figure out critical task dependencies and draw out a schedule with the help of the project manager. From this schedule, the final production budget can be derived.

Assets Acquisition

They must estimate acquisition times frames based on their research, rights and dubbing costs as well as original duplication and transfer costs

Production: graphics, text, icons, buttons, etc.

Each specialist must take the final parts count and identify the assets he/she is going to produce.

They need to carefully estimate how much time its going to take them to produce each and identify where they will be able to streamline the production process with automation (macros, batch processing, etc.)

They need to identify which items depend on one another to be created and string the tasks together to come up with a schedule and a final man-hour count.

They need to meet with Software Engineering to see identify critical assets hand-offs, tests and revision contingencies

They also need to meet with Assets acquisition to coordinate steady production flow

Production: sound and video

Sound and video specialists need to estimate production, digital capture and encoding costs.

Key issues to watch out for are
Talent costs
Crews and equipment
Travel and pier diem
off-line and on-line editing
stock and dubbing
footage transfers
Producer and other key personnel
location fees etc.

Software Engineering

The lead software engineering must design the software architecture

It then needs to be broken down into modules and subroutines

As in production, the Engineers must carefully

Estimate how much time its going to take them to produce each and identify where they will be able to build code writing efficiencies

They need to identify which module depend on one another to be created and string the tasks together to come up with a schedule and a final man-hour count.

Items to not overlook:
One-off disc pressings for in progress testing

mastering / compiling time
mastering and replication costs at end of project
archival materials (exabyte tapes, etc..)
meeting time: engineering, project team, etc.

They must develop a plan for memory management, coding and documentation standards

They need to include time for documentation, archiving, engineering testing and debugging

They need to meet with Production to see identify critical assets hand-offs, tests and revision contingencies

Testing and quality administration

One must put together a testing plan throughout the production development

Testing must cover: code, assets accuracy, script proofing, fact checking, focus group testing of completed working modules, interface functionality, rights clearance verification, etc.

Several cycles need to be anticipated towards the end of the project

A testing database for bug reports, bug report convention needs to be established

Master schedule
This is where it all comes together. Once every team members have done their part, the Project Manager consolidates the information.

The Project Manager gets the information from each team after carefully reviewing it with them.

He/she consolidates it into a program like Microsoft Project in order to output a PERT of GANT chart outlining the entire production process

The team gets together and reviews the schedule (sometimes take a day) to identify any errors or further efficiencies.

Ö **APPROVE** master schedule

Budget

The project manager computes the master schedule into a final budget

Ö **APPROVE** final budget

Phase III: Production

If the plan is solid, the production phase has great chances of being completed successfully on time, on budget and according to the design specs. However, a plan requires close supervision, great collaboration, constant deadline driven objectives and above all, great communications.

Management critical issues

tangible key milestones every month

weekly objective for each production member

weekly team meeting to review progress and objectives

weekly status reports

monthly financial reports (see budget section)

conflict resolution skills

clear sense of priorities

Phase IV: Testing and validation

Consumer CD-ROM multimedia titles must be absolutely crash proof. The distribution channels

and the retail prices are such that versioning products release of the same title/ upgrade is not realistic. It is critical that a very rigorous testing cycle be put in place once the product is in beta. Testing is much more than debugging, it includes many aspects that standard software application do not normally have to deal with.

Testing critical issues

Test every pathways of the product forward and backwards

Establish excellent bug report strategy, forms, consolidation and cycling

Perform real time observed focus group testing at key point of the project: prototype, alpha and beta.

Videotape testing operations to be able to backtrack bug pathway

Test for:

Functionality bugs

Performance bugs

Assets accuracy bugs (is this the right asset playing here?)

Graphics alignment bugs

and many more

Engage testers who are good at giving directions

Look at testing programs like Microsoft Test to test crash bugs

Staffing

Core Project Team

Project Manager (Producer /Project Director

Interactive designer

Writer/Editor/content expert

Software Engineer

Technical Producer

Quality Administrator

Free-lance / project staff

Art Director

Video Producer

Audio Producer

Acquisitions Manager

Graphic Artists

Programmers / Authors

Researchers

Production Assistants

Assets Librarian / Database Administrator

Technical Coordinator

Testers

Proof-readers

Fact-checkers

Project Accountant

Freelance or Staff?

How to balance people you hire full time on site, contractors working on site, contractors working off site, and service companies? There are many factors involved:

Free-lance talent pool in your area

Graphic artists are usually abundant but require training to deal with pixel based computer imagery rather than vector based graphics. They need significant training and supervision during the first weeks of production

Interactive Designer are hard to find. They are a unique breed and its hard to cipher the good ones from the fakes. If you find a good one, you should probably hang on to them.

Software Engineers and programmers are usually available. On a freelance basis though, their rates is going to be significantly more than if they were on staff. Maybe short term contract are most appropriate here.

Project Managers will multimedia project experience are as rare as Interactive designers. However, a good project manager for a video production or for a print project can just as effective if they have a strong affinity for computers and if they are educated for the first few weeks.

Budget

Typical costs 200-400K

Most titles on the consumer market fall in this budget range. This includes all production costs specific to the project from concept to release master. It does not include initial start-up costs such as capital investment in equipment, facilities and senior management overhead.

Typical breakdown

Every title is different but as a general rule of thumb one can assume the following general breakdown of costs:

1/3 design

1/3 content

1/3 software engineering

On some project Software Engineering can absorb 40-50% of the costs particularly if they are very rich in interaction. On others, the rights to acquire content and/or the cost of producing high quality original material can drive the cost of content up. This is especially true or high quality animation projects.

Series strategy

An attractive aspect of interactive multimedia, is the ability to create a good design and software engineering template which will then be used to build many titles similar in interface but with different content.

The first title can cost $300-500K but then subsequent titles can cost $100K-$300K depending

The subsequent titles mostly require content creation with a little bit of redesign and a little bit of re-engineering. But for the most part, you can shave from 1/3 to 2/3 of the first project cost.

You also build production process efficiencies as you cookie cut titles from the same mold which

save you time and money.

Contingencies

Its very important to build in 5-15% contingency in all budgets. Its especially important on your first project.

Many, many small unforeseen things will surface, and increase costs

Tracking: key to survival

You need a very solid way to track budgets

Leverage template costs (shell) which is 2/3 of cost (design & engineering)

Amortize template over at least 3 similar titles

Schedule

Typical time-frames 6-12 months

Microsoft Project a good tool for setting up and tracking schedule

Weekly progress reports

Monthly Actual versus plan

Allow 2-3 month set-up time for first title (studio, equipment, training)

Project Management

The driving force

Not top down but the center of the wheel

Conflict resolution is a critical skill

channeling team communications

Tips

Start with simple project (not an encyclopedia)

Staff only what you want to retain tight control over, freelance or contract rest until build-up significant production outfit

Two areas which always go over: design and graphics (subjective)

Give priority to budget and schedule over design and content quantity

Watch out for 'creeping elegance': everyone has their two cents to put in

Respect deadlines and approvals: Empower the Project Manager to enforce them.

Details and boring stuff is what ultimately gets an interactive project production in trouble if not managed properly: pay attention to details! pay attention to details! pay attention to details!

Once you select a pathway or a tool stick to them for the duration of the project. Many new tools and better pathways will tempt you along the way every two to four weeks. New ways in the middle of production could throw a monkey wrench in the process and do much more harm than good.

Pitfalls

Set-up studio

Hardware and software selection is critical to the success of the production process. Take time to evaluate both carefully and test your assumptions before committing to either.

Take the time to properly install hardware and software, as well as tracking tools, back-up and archival systems before you begin.

Avoid mid-stream change of personnel or hardware or physical location.

It can usually cost more in both equipment and time to add hardware and software as they become needed during the production process that doing it all up-front. During the production process. people's mind should be on the title not on the tools they need.

There is no such thing as a plug and play tool in multi-media application now matter how hard the hardware sales person tries to convince you. Allow 1-3 days for any machine to be installed and tested.

In most cases its a good idea to dedicate an individual full time to maintain systems, software, back-ups, file transfers, network, etc...

Training and ramp-up time

It always seem to take longer to educate people on the specifics of the project, but its far offsets the costs of 'learning by trial and error'

Every multimedia project has its specifics and everyone on the team needs to be in sync on the common issues, weekly communication is key. When any standards has been approved or set, make sure everyone on the team understands it and knows how to apply it in their particular area.

Most common over budget areas

Design and graphics always tend to go over budget

Design

Multimedia is a very exiting medium and its very easy to grow from a simple project to a very elaborate and complicated one.

Over-designing is a classic first timer problem when in doubt, go for simplicity, remember at all times who your audience is.

Watch out for creeping elegance

Most consumers don't give a hoot about multimedia. Al they care to be able to do things easier, faster, cheaper and have fun. (lots of buttons, 'neat' effects and fancy transitions wear out very quickly and often get in the way of the content)

Set clear priorities and objectives remember what's important: the content and how to get to it effortlessly in an engaging manner.

Graphics

Art Direction here is sometimes the biggest culprit.

Once Keyframes are approved, stick to them

Changing the look and feel of the graphic mid-stream usually affects thousands of elements which have to be modified or re-done

Assets acquisitions delays can also cause significant problems: start early.

Every Artist has their own pace. Make sure the individuals you plan to hire actually estimate the work they have to do for you and include those estimate in the master schedule

Software Engineering

When dealing with a title requiring significant research and development, Software Engineering can become the biggest culprit since they are at the end and represent sometimes as much as 40% of the budget.

Be sure to test any ambitious software engineering concepts before you buy off the final design.

Object oriented programming is probably the most efficient way short term and long term to build your application

Avoid hardwiring assets into the code as you'll find that most of the time spend by the engineers will be typing assets names (expensive assistants)

Do not ever hesitate to call upon other's existing experience, chances are someone already has written a subroutine that accomplishes your objective. And if not free, that code can sometime be negotiated to your advantage. Programming in a vacuum in this industry will quickly put you out of the multimedia production business.

Expectations

Managing expectations is challenging in multimedia since after the prototype, the product is not likely to come together again until the Beta (or next to the end of the project

One must be realistic about goals and objectives, always pad estimates by 10 to 20% before locking in final schedule

Always allow more time for approvals and meeting time

End.

Jumpstart Help File

Welcome to Jumpstart!

Jumpstart has been re-designed to allow you easier navigation through a stylized front-end. Information about the following topics are available:

{ewl MVBMP2, ViewerBmp2, orgcht5.shg}

Root

Program that decides which video mode you are running in and runs the appropriate version of the program. This is the program you should run to browse thorugh the information on the disk.

go.exe

The main Visual Basic Program designed for either 640 X 480 or 1024 X 768.

js2.exe
js2640.exe

Required Visual Basic DLL's and VBX's.

msajt110.dll
msaes110.dll
vbrun300.dll
vbdb300.dll
mci.vbx

Bitmaps used by the programs.

mssans.bmp
firstbmp.bmp
earth.avi

Access database which contains context string information as well as group and browse sequence info.

context.mdb
context.ldb

The Viewer title which is run from the Visual Basic application.

js.mvb

Development

This directory contains all the sample source code, Software Development Kits, Systems Software, Technical Articles, and Tools on the CDROM. This directory is where you will find in depth information and material on developing Multimedia Applications. You will also find a full verion of Video for Windows 1.1 including the Software Development Kit.

Sample Applications

This directory contains various sample applications from companies that have successfully developed Multimedia Titles. In these directories you will find disabled versions of their applications. These applications are very usefull in giving the developer an idea of what Multimedia capabilities exist under Windows.

This is a list of all the files in the Samp_Aps Directory

samp_aps\broderbd*.*
01.bmp 03.bmp* 05.bmp* 07.bmp* 09.bmp* kpcdemo.exe*
02.bmp 04.bmp* 06.bmp* 08.bmp* 10.bmp*

samp_aps\crawfd*.*
*crawford.avi**

samp_aps\foundat*.*
*kimage.avi**

samp_aps\hilbers*.*
hilbers.avi hilbers2.avi**

samp_aps\intermed*.*
aaplay.dll aawin.exe* chklist.ms* faucet.flc* toggle.flc**

samp_aps\jasmine*.*
autmlvs2.avi flowrbud.avi* humbird.avi* pondrip2.avi**

samp_aps\mondo*.*
aaplay.dll aawin.exe* page1.flc* page2.flc* page3.flc**

samp_aps\musicpn*.*
[0] [19] [bd0midi] banddll.dll lenny.exe**
[16] [1] [vwr] bandutil.dll sched.dll**
[17] [4] band.ini bd6.msf**
[18] [6] band1.ini go.exe**

samp_aps\musicpn\0*.*
bd0.pag bd2.exe* bd4.exe* bd57.pag**
bd1.exe bd3.exe* bd5.msf* bd6.msf**

samp_aps\musicpn\16*.*
bd1660.msf bd1667.msf* bd1674.msf* bd1681.msf* bd1688.msf* bd1695.msf**
bd1661.msf bd1668.msf* bd1675.msf* bd1682.msf* bd1689.msf* bd1696.msf**
bd1662.msf bd1669.msf* bd1676.msf* bd1683.msf* bd1690.msf* bd1697.msf**
bd1663.msf bd1670.msf* bd1677.msf* bd1684.msf* bd1691.msf* bd1698.msf**
bd1664.msf bd1671.msf* bd1678.msf* bd1685.msf* bd1692.msf* bd1699.msf**
bd1665.msf bd1672.msf* bd1679.msf* bd1686.msf* bd1693.msf**
bd1666.msf bd1673.msf* bd1680.msf* bd1687.msf* bd1694.msf**

samp_aps\musicpn\17*.*
bd1700.msf bd1706.msf* bd1712.msf* bd1718.msf* bd1724.msf* bd1730.msf**
bd1701.msf bd1707.msf* bd1713.msf* bd1719.msf* bd1725.msf* bd1731.msf**
bd1702.msf bd1708.msf* bd1714.msf* bd1720.msf* bd1726.msf* bd1732.msf**
bd1703.msf bd1709.msf* bd1715.msf* bd1721.msf* bd1727.msf* bd1733.msf**
bd1704.msf bd1710.msf* bd1716.msf* bd1722.msf* bd1728.msf* bd1734.msf**
bd1705.msf bd1711.msf* bd1717.msf* bd1723.msf* bd1729.msf**

samp_aps\musicpn\18*.*
bd1800.msf bd1805.msf* bd1810.msf* bd1815.msf* bd1820.msf* bd1825.msf**
bd1801.msf bd1806.msf* bd1811.msf* bd1816.msf* bd1821.msf* bd1826.msf**
bd1802.msf bd1807.msf* bd1812.msf* bd1817.msf* bd1822.msf* bd1827.msf**
bd1803.msf bd1808.msf* bd1813.msf* bd1818.msf* bd1823.msf* bd1828.msf**

bd1804.msf bd1809.msf* bd1814.msf* bd1819.msf* bd1824.msf* bd1829.msf**

samp_aps\musicpn\19*.*
bd1910.wav bd1912.wav* bd1914.wav* bd1916.wav* bd1918.wav* bd1920.wav**
bd1911.wav bd1913.wav* bd1915.wav* bd1917.wav* bd1919.wav**

samp_aps\musicpn\1*.*
bd150.msf bd152.pag* bd154.msf* bd156.msf* bd158.wav* bd161.msf* bd163.wav**
bd150.pag bd152.wav* bd154.pag* bd156.wav* bd159.wav* bd161.wav* bd165.msf**
bd151.msf bd153.msf* bd154.wav* bd157.msf* bd160.msf* bd162.msf**
bd151.pag bd153.pag* bd155.msf* bd157.wav* bd160.pag* bd162.wav**
bd152.msf bd153.wav* bd155.wav* bd158.msf* bd160.wav* bd163.msf**

samp_aps\musicpn\4*.*
bd400.msf bd401.msf* bd402.wav* bd410.msf* bd412.wav* bd421.pag* bd424.pag**
bd400.pag bd401.wav* bd403.msf* bd410.wav* bd413.wav* bd422.pag**
bd400.wav bd402.msf* bd403.wav* bd411.wav* bd420.pag* bd423.pag**

samp_aps\musicpn\6*.*
bd600.msf bd603.msf* bd606.msf* bd612.msf* bd653.msf* bd659.msf**
bd600.pag bd603.pag* bd607.msf* bd613.msf* bd654.msf* bd660.msf**
bd601.msf bd604.msf* bd608.msf* bd614.msf* bd655.msf* bd661.msf**
bd601.pag bd604.pag* bd609.msf* bd650.msf* bd656.msf* bd662.msf**
bd602.msf bd605.msf* bd610.msf* bd651.msf* bd657.msf* bd663.msf**
bd602.pag bd605.pag* bd611.msf* bd652.msf* bd658.msf* bd664.msf**

samp_aps\musicpn\bd0midi*.*
*bd0midi.pag**

samp_aps\musicpn\vwr*.*
[rtf] musicpen.log musicpen.mvb* musicpen.mvp* readme.doc**

samp_aps\musicpn\vwr\rtf*.*
*readme.rtf**

samp_aps\pixar*.*
pixar.avi pixar2.avi**

samp_aps\pmi*.*
6x4goin2.flc aqua1e.mid* bid_d4.flc* ltcback.dll* readme.doc**
a_demo1c.wav aucgoing.wav* bid_d5.flc* ltcbackd.dll* readme.txt**
a_demo2c.wav aucpre1.wav* bid_d6.flc* mciaap.drv* tdossi06.wav**
a_demo31.wav auctit01.flc* bidders.exe* nur_regs.flc* tninfov.wav**
a_demo32.wav bid_a_in.flc* copyrit2.bmp* nurawel2.flc* tninfov2.wav**
a_demo33.wav bid_d1.flc* dossie.flc* nurglad.wav* warning.bmp**
a_demo34.wav bid_d2.flc* eshelm4.mid* nurglad1.flc* z33_5_2.wav**
a_demo3b.wav bid_d31.flc* gone.exe* nuria.exe* zeekd021.flc**
a_demo4d.wav bid_d32.flc* logos5.mid* nurwel.wav* zeekd022.flc**
a_demo5d.wav bid_d33.flc* lpmires.dll* profile.doc* zeke.exe**
aaplay.dll bid_d34.flc* lpmiresd.dll* r33_5_1.wav**

samp_aps\walksoft*.*
[nim]

samp_aps\walksoft\nim*.*
[index] nim.tbk qtnotify.exe* readme.txt* tbkutil.dll**

[letters] nimprint.dll qtole.dll* sparky.ico* tbkwin.dll*
[movs] qcmc.qtc* qtraw.qtc* tbkbase.dll* tbook.exe*
[papers] qtcvid.qtc* qtrle.qtc* tbkcomp.dll* xword.fon*
bwcc.dll* qthndlr.dll* qtrpza.qtc* tbkdb3.dll*
comm_dll.dll* qtim.dll* qtrt21.qtc* tbkdlg.dll*
newsqtw.dll* qtimcmgr.dll* qtsmc.qtc* tbkfile.dll*
nim.dll* qtjpeg.qtc* qtvhdw.dll* tbknet.exe**

samp_aps\walksoft\nim\index*.*
africa.bok catherwo.bok* headline.bok* oped.bok* world.bok*
americas.bok* comics.bok* japan.bok* photos.bok*
animatio.bok* easterne.bok* letters.bok* russia.bok*
asia.bok* ec.bok* middleea.bok* sports.bok*
business.bok* gourmet.bok* nim.idx* unitedst.bok**

samp_aps\walksoft\nim\letters*.*
*sample.txt**

samp_aps\walksoft\nim\movs*.*
cather.mov letters.mov* popcorn.mov* typing.mov*
horosc.mov* moon.mov* rabbit.mov* wipe.mov**

samp_aps\walksoft\nim\papers*.*
[nm930731] [nm930911] nm930828.nim
[nm930828] nm930731.nim* nm930911.nim**

samp_aps\walksoft\nim\papers\nm930731*.*
aborted.txt cuba.txt* iranrebs.txt* nafta.txt* spieg2.txt*
aboutnim.txt* disable.txt* iraqiso.txt* neonazi.txt* stadiums.txt*
africa.txt* express.pic* isanal.txt* newsday.txt* subic.txt*
afrscore.txt* farmer.pic* israel.pic* nicarag.txt* sudanter.txt*
albania.txt* flood.mov* israelop.txt* nolumber.txt* syria.pic*
animals.txt* flood.pic* isrtort.txt* oldman.pic* thu.xwa*
asiaimp.txt* floodsci.txt* italsuic.txt* ramos.txt* thu.xwd*
asylum.txt* fri.xwa* italycor.txt* recipe.txt* tsars.txt*
bars.pic* fri.xwd* jap3.txt* reveries.mov* tue.xwa*
blast.pic* garbage.pic* japanus.txt* reveries.txt* tue.xwd*
boers.txt* garbage.txt* japedit2.txt* russiaop.txt* ukraine.txt*
bosanal.txt* gay.mov* japvoter.txt* russtrav.txt* usdraft.txt*
bosedit.txt* gayaclu.txt* japwom.txt* safcap.txt* ussuic.txt*
cat.pic* germany.txt* major.txt* sat.xwa* vampire.mov*
cather.txt* ginsburg.txt* mandela.txt* sat.xwd* vigileng.txt*
china.txt* guard.pic* march.pic* shilts.txt* wash.pic*
clinted.txt* haiti.txt* milart.txt* skinhead.txt* wed.xwa*
cover.pic* head_01.pic* mirrors.txt* smith.txt* wed.xwd*
credits1.txt* horoscop.txt* mon.xwa* somaliun.txt* westbank.txt*
credits2.txt* ibmop.txt* mon.xwd* spieg1.txt**

samp_aps\walksoft\nim\papers\nm930828*.*
altmed.txt credits2.txt* internet.mov* mostar.txt* schiz.txt*
andes.txt* doctors.txt* internet.txt* namibia.txt* shoespt.txt*
andrew.txt* eeur_01.pic* isrpamp.txt* navajo.txt* smudge.mov*
apology.txt* egyptact.txt* japsuit.txt* nicarag.txt* sovrepub.txt*
asia_01.pic* elecarsd.txt* kenwacky.txt* oped_01.pic* sovsci.txt*
asia_02.pic* eleccar.txt* kidnaped.txt* panther.txt* spts_01.pic*
bingaman.txt* estonia.txt* kids_01.pic* plofin.txt* statehea.txt**

blues.txt eur_01.pic* king.mov* privacy.txt* taiwan.txt*
bonmeet.txt* exxoned.txt* king.pic* rabinanl.txt* teaequal.txt*
bosanal.txt* eyecut.txt* king.txt* readers.txt* thu.xwa*
brazilmd.txt* fri.xwa* korea.txt* rec.txt* thur.xwd*
bullies.txt* fri.xwd* kuwait.txt* recipe.txt* tues.xwa*
catfish.txt* gator.txt* lemonde.txt* roblou.txt* tues.xwd*
cather.txt* gehealth.txt* march.txt* ruchurch.txt* turtle.txt*
china.txt* germen.txt* mars.txt* rus_01.pic* wed.xwa*
clintpol.txt* germprop.txt* mide_01.pic* ruscrime.txt* wed.xwd*
commies.txt* haiti.txt* mirrors.txt* safeduc.txt* woodstck.txt*
cover.pic* horoscop.txt* mon.xwa* sat.xwa* woodsto.mov*
credits1.txt* hosokawa.txt* mon.xwd* sat.xwd**

walksoft\nim\papers\nm930911*.*
abortdoc.txt chess.txt* heaanal.txt* mahbued.txt* somalia.txt*
afr_01.pic* china.mov* head_01.pic* midanal.txt* spts_01.pic*
afr_02.pic* chinett.txt* health.txt* mided.txt* stats.txt*
amer_01.pic* clinpoor.txt* horoscop.txt* mon.xwa* stocks.txt*
amer_02.pic* cover.pic* horthy.txt* mon.xwd* tennis.txt*
antinaf.txt* credits1.txt* immig.txt* msdrug.txt* textown.txt*
arafbond.txt* credits2.txt* indblind.txt* oped_01.pic* thu.xwd*
armenia.txt* domjapan.txt* interv.txt* rec1.txt* thur.xwa*
aspin.txt* ec_01.pic* islam.txt* recipe.txt* travel.txt*
bizop.txt* fire.mov* issplit.txt* rusdef.txt* tue.xwd*
bosfam.txt* fire.txt* kids_01.pic* russex.txt* tues.xwa*
boshair.txt* fri.xwa* labor.txt* s&s.pic* union.txt*
brazil.txt* fri.xwd* lat1.txt* saf.txt* wed.xwa*
burma.txt* genx.txt* lat2.txt* safviol.txt* wed.xwd*
cambodia.txt* gloria.mov* lat3.txt* sat.xwa* workers.txt*
canporn.txt* goreed.txt* lawmaker.txt* sat.xwd*
cather.txt* hair.mov* lesmil.txt* sexed.txt*
centam.txt* haiti.txt* letter.txt* shalied.txt*

samp_aps\zone*.*
apdgv.vdr mark2.avi* mmdemo.exe* opening.avi*
dblbu1.avi* mmdemo.apr* newden2.avi**

Sample Media

This directory contains various Audio, Video, Animation, and Graphics clips. These media elements were provided to give the developer an idea of what type of media exists for Multimedia Windows. It also gives the developer some sample media to test with their applications. This media cannot be shipped with your application since it is copyrighted material.

Miscellaneous

This directory contains information on Marketing, General Viewer, and other miscellaneous material.

This is a list of all the files in the Develpmt\Drivers Directory:

mscdex\bin*.*
audio.exe hitachia.sys* mspl11.bin* testdrv.exe*
books.bin* mscdex.exe* sbc.bin* testdrv.pro*
chklist.ms* mspl.bin* svd.exe*

mscdex\cdspeed*.*
blktest.bat cdtest.bat* chartcd2.xlm* readme.txt* speed.txt*
cdspeed.exe* chartcd1.xlm* chklist.ms* speed.rtf* speed.xlc*

mscdex\hitachi*.*
cd.asm cdread.asm* cdread.doc* macros.mac* mscdex.asm* tracer.asm*
cd.inc* cdread.at* ecc.asm* makefile* mscdex.inc*

mscdex\inst_cd\icsample*.*
drvproc.c icsample.def* icsample.rc* libinit.asm*
icsample.c* icsample.h* icsample.rcv* makefile*

mscdex\inst_cd\msrlec*.*
df.asm libinit.asm* msrlec.c* msrlec.h* msrlec.rcv* rle.h*
drvproc.c* makefile* msrlec.def* msrlec.rc* playrle.c* rlea.asm*

mscdex\iscdrom*.*
*iscdrom.asm**

mscdex\svd*.*
makefile makefile.rel* svd.c*

mscdex\testdrv*.*
books.bin mspl.bin* request.c* send.asm* testdrv.cvh* testdrv.rtf*
chklist.ms* mspl11.bin* request.h* setup.c* testdrv.exe* testdrv.txt*
cmacros.inc* prepare.c* rstruct.h* support.c* testdrv.log* tests.c*
makefile* proto.h* sbc.bin* test.h* testdrv.map*
message.h* readme.txt* script.hex* testdrv.c* testdrv.pro*

This is a list of all the files in the Develpmt\Examples Directory:

infobrws\avi*.*
[lowres] demo.avi help1.avi* textinfo.avi*
audio.avi* diamond.bmp help2.avi* video.avi*
audio.bmp* dim2stev.avi* help3.avi* video.bmp*
audio1.avi* dim2will.avi* earth.avi* helpno.avi* video1.avi*
audio2.avi* graphic1.avi* media.avi* video2.avi*
audio3.avi* graphic2.avi* media1.avi* video3.avi* cloudsgr.avi*
graphic3.avi* mplayer.avi* vidinfo.avi*
defindeo.avi* graphics.avi* question.avi*
defmsv16.avi* graphics.bmp* refer.avi*
defmsv8.avi* help.avi* refer1.avi*
defrle.avi* help.bmp* sunwatr2.avi*

infobrws\avi\lowres*.*
audio.avi defmsv16.avi* graphic3.avi* help3.avi* textinfo.avi*
audio.bmp* defmsv8.avi* graphics.avi* media.avi* video.avi*
audio1.avi* defrle.avi* graphics.bmp* media1.avi* video.bmp*
audio2.avi* demo.avi* help.avi* mplayer.avi* video1.avi*
audio3.avi* earth.avi* help.bmp* question.avi* video2.avi*
cloudsgr.avi* graphic1.avi* help1.avi* refer.avi* video3.avi*
defindeo.avi* graphic2.avi* help2.avi* sunwatr2.avi* vidinfo.avi*

infobrws\ini*.*
[lowres] button2.ini button4.ini* button6.ini*
button1.ini* button3.ini* button5.ini* morph.ini*

infobrws\ini\lowres*.*
button1.ini button3.ini* button5.ini* morph.ini*
button2.ini* button4.ini* button6.ini*

infobrws\lowres*.*
playvfw.ini

infobrws\src*.*
ani.cur dialogs.sbr* mciutils.c* playvfw.ini* proto.h*
animate.cur* graphics.cur* mciutils.obj* playvfw.mak* refer.cur*
audio.cur* help.cur* mciutils.sbr* playvfw.map* segentry.dat*
button1.cur* help.hlp* media.cur* playvfw.obj* style.h*
button2.cur* init.c* msvc.pdb* playvfw.pdb* test.c*
button3.cur* init.obj* newlogo.bmp* playvfw.rc* text.cur*
button4.cur* init.sbr* palette.pal* playvfw.res* utils.c*
button5.cur* libentry.asm* playvfw.bak* playvfw.sbr* utils.obj*
button6.cur* libentry.obj* playvfw.bpt* playvfw.vcw* utils.sbr*
chklist.ms* makefile* playvfw.bsc* playvfw.wsp* vgalogo4.bmp*
dialogs.c* makefile.bak* playvfw.c* profile.c* video.cur*
dialogs.dlg* mci.c* playvfw.def* profile.obj*
dialogs.obj* mci.obj* playvfw.h* profile.sbr*
dialogs.res* mci.sbr* playvfw.ico* proto.bak*

infobrws\wav*.*
ahhh.wav facttoot.wav* help.wav* text.wav* water.wav*
anim.wav* graf.wav* laser.wav* tron120a.wav*
chimes.wav* guitblus.wav* laugh.wav* tron133.wav*
clock.wav* harp1.wav* oooh.wav* video.wav*

examples\rotate*.*
bmfx.vbx parrot.avi* rotate3.exe* threed.vbx*
mciwndx.vbx* parrot3.exe* soldiers.avi* vbrun200.dll*

This is a list of all the files in the Developmt\S_Devtls Directory

s_devtls**.
tshelp.hlp tshelp1.hlp tsread.txt tstools.dll tstools.mvb tstools1.dll

This is a list of all the files in the Developmt\SDK directory:

sdk\mwme*.*
[avisetup] [editors] [vbx] readme.wri
[demo] [install] promot.hlp vbtoolkt.hlp**

sdk\mwme\avisetup*.*
*_mssetup.ex_* mplayer.ex_* mscuistf.dl_* msvidc.dr_* setup.ini**
*_mstest.ex_* mplayer.hl_* msdetect.in_* msvideo.dl_* setup.lst**
*dispdib.dl_* mplayer.re_* msdetstf.dl_* profdisp.ex_* setup.mst**
indeo.dr_* mscomstf.dl_* msinsstf.dl_* setup.exe setupapi.in_**
*mciavi.dr_* mscpydis.dl_* msshlstf.dl_* setup.in_* ver.dl_**
*mciole.dl_* mscpydis.in_* msuilstf.dl_* setup.inf**

sdk\mwme\demo*.*
[bitmaps] ianimscr.frm ipictbut.frm* mwmedemo.exe* toolmenu.frm**
[mwf] iavi_lt.vbx isndansc.frm* order.txt* vbrun300.dll**
[sound] intro.avi ivideosc.frm* pictbtnl.vbx* vbtoolkt.bas**
house1.avi introscr.frm* mainscr.frm* proeng.dll* vbtoolkt.hlp**
house1.ivd ipict.frm* mci.vbx* proeng.ini* vbtoolkt.mak**
ianim_lt.vbx ipict_lt.vbx* mw_me.ico* sndanlt.vbx**

sdk\mwme\demo\bitmaps*.*
ani_dn.bmp ftrkdn.dib* ip_up.bmp* mwlogo.dib* stardn.dib**
ani_up.bmp ftrkup.dib* macscr1.dib* playdn.dib* starup.dib**
btrkdn.dib fwddn.dib* macscr1.hsp* playup.dib* stkbtn2.dib**
btrkup.dib fwdup.dib* macscr2.dib* rewdn.dib* stockbtn.dib**
cb_dn.bmp icon.dib* macscr2.hsp* rewup.dib* stopdn.dib**
cb_up.bmp im_dn.bmp* macscr3.dib* sac_dn.bmp* stopup.dib**
computer.dib im_up.bmp* macscr3.hsp* sac_up.bmp**
computer.hsp ip_dn.bmp* mtitle.dib* star.dib**

sdk\mwme\demo\mwf*.*
*test.mwf**

sdk\mwme\demo\sound*.*
sandesc.san sandesc2.wav* startsnd.wav**
sandesc.wav singdart.wav* switch.wav**

sdk\mwme\editors*.*
bumble.mwf impexp.dll* perfcars.mwf* pro_dlg.dll* sanedit.exe**
ctrlpad.dll io_ctrl.dll* pictbtnl.vbx* proeng.dll* sndanlt.vbx**
dib.drv link.bmp* play.bmp* proeng.ini* timeline.dll**
iavi_lt.vbx paint.dll* player.exe* promot.exe* unlink.bmp**
iaviedit.exe path_seq.dll* preview.dll* promot.hlp* waveobj.vbx**

sdk\mwme\install*.*
install.ex$ install.ins* setup.bmp* setup.exe**

sdk\mwme\vbx*.*
ianim_lt.vbx iavi_lt.vbx* ipict_lt.vbx* pictbtnl.vbx* sndanlt.vbx**

sdk\vfw11.win*.*
[multilng] [runtime] [vfwdk] [winvideo] dev_kit.txt readme.txt

sdk\vfw11.win\multilng*.*

multilng.avi

sdk\vfw11.win\runtime*.*
[disks] [setupsrc]

sdk\vfw11.win\runtime\disks*.*
[frn] [ger] [usa]

sdk\vfw11.win\runtime\disks\frn*.*
[disk1]

sdk\vfw11.win\runtime\disks\frn\disk1*.*
*_mssetup.ex_ dva.38_ ir30.dl_ msadpcm.ac_ msshlstf.dl_ ole2prox.dl_*
*_mstest.ex_ iccvid.dr_ map_win.hl_ mscomstf.dl_ msuilstf.dl_ profdisp.ex_*
*acmcmprs.dl_ imaadpcm.ac_ mciavi.dr_ mscpydis.dl_ msvidc.dr_ setup.in_*
avicap.dl_ indeo.in_ mciole.dl_ mscpydis.in_ msvideo.dl_ setup.ini
avifile.dl_ indeov.dr_ mplayer.ex_ mscuistf.dl_ ole2.dl_ setup.mst
*cleanup.re_ iniupd.dl_ mplayer.hl_ msdetect.in_ ole2.re_ setupapi.in_*
*compobj.dl_ install.exe mplayer.re_ msdetstf.dl_ ole2conv.dl_ storage.dl_*
*ctl3d.dl_ install.lst msacm.dl_ msinsstf.dl_ ole2disp.dl_ typelib.dl_*
*dispdib.dl_ ir21_r.dl_ msacm.dr_ msrle.dr_ ole2nls.dl_*

sdk\vfw11.win\runtime\disks\ger*.*
[disk1]

sdk\vfw11.win\runtime\disks\ger\disk1*.*

sdk\vfw11.win\runtime\disks\usa*.*
[disk1]

sdk\vfw11.win\runtime\disks\usa\disk1*.*

sdk\vfw11.win\runtime\setupsrc*.*
[files] [layout] [script] mkrtime.bat

sdk\vfw11.win\runtime\setupsrc\files*.*

sdk\vfw11.win\runtime\setupsrc\layout*.*
setup.lyt

sdk\vfw11.win\runtime\setupsrc\script*.*
setup.mst

sdk\vfw11.win\vfwdk*.*

sdk\vfw11.win\winvideo*.*

This is a list of all the files in the Develpmt\Source Directory

source\cropdib*.*
chklist.ms cropdib.exe* cropdib.map* cropdib.res* dib.obj*
cropdib.c* cropdib.h* cropdib.obj* dib.c* makefile*
cropdib.def* cropdib.ico* cropdib.rc* dib.h**

source\dialogs*.*
color.c dialogs.def* dialogs.ico* font.c* readme.txt*
color.dlg* dialogs.exe* dialogs.rc* makefile* search.c*
dialogs.c* dialogs.h* file.c* print.c* select.dlg**

source\hotspot*.*
[editor] [exe] [include] [vbx] [viewer] readme.new

source\hotspot\editor*.*
*avi.c avihed.rc drawhot.c hotspot.sbr pattern.bmp sbcls.c
avihed.def avihede.h file.c icon1.ico puzzproc.obj util.c
avihed.h avihot.doc hotspot.c main.c readme.txt
avihed.mak dlg.c hotspot.h movie.c resource.h*

source\hotspot\exe*.*
*animat.avi avihed.exe hotspot.dll sample.avi test.mvb
animat.hot avihot.vbx hottst.exe sample.hot*

source\hotspot\include*.*
digitalv.h

source\hotspot\vbx*.*
*avi.c avihapp.def avivbx.h hotspot.c hottst.mak sample.avi
avicd.bmp avihot.def debug.c hotspot.h main.c sample.hot
avicu.bmp avihot.mak drawhot.c hotspot.rc movie.c sbcls.c
avieu.bmp avimu.bmp drawhot.obj hottst.frm resource.h util.c*

source\hotspot\viewer*.*
*animat.hot drawhot.c hotspot.mak mvbtask.c test.mvp
avi.c hotspot.c hspplay.c readme.txt test.rtf
avihapp.h hotspot.def main.c sbcls.c util.c
debug.c hotspot.h movie.c test.mvb*

source\lava*.*
chklist.ms lava.exe* lava.map* lava.res* lv.asm* sqrt.asm*
lava.c* lava.h* lava.obj* lava.sym* lv.obj* sqrt.obj*
lava.def* lava.ico* lava.rc* lava3.rtf makefile**

source\mergedib*.*
chess1.dib file.c* mergedib.c* mergedib.rc* mtnrain1.dib*
chess2.dib* file.obj* mergedib.def* mergedib.rtf mtnrain2.dib*
dib.c* gmem.h* mergedib.h* mtnlk1.dib* nocrap.h*
dib.h* makefile* mergedib.ico* mtnlk2.dib* rare.c*
dib.obj* mem.asm* mergedib.obj* mtnlk3.dib* rare.obj**

source\midikeyb*.*
makefile midikeyb.h* testkeyb.c* testkeyb.ico*
midikeyb.c* midikeyb.rc* testkeyb.def* testkeyb.lnk*
midikeyb.cur* midikybd.rtf testkeyb.h* testkeyb.rc**

source\palfx*.*
dib.c makefile* palfx.c* palfx.ico* sampver.h*
dib.h* mtnrain1.dib* palfx.def* palfx.rc* sampver.ver*
gmem.h* mtnrain2.dib* palfx.h* palfx.rcv*

source\rleapp*.*
chklist.ms dog.rl0* readme.txt* rlea.asm* rleapp.h* rlefile.c*
df.asm* gmem.h* rle.c* rleapp.c* rleapp.ico*
dib.c* makefile* rle.exe* rleapp.def* rleapp.rc*
dib.h* mem.asm* rle.h* rleapp.dlg* rledlg.c*

source\roadshow*.*
agenda.rtf eplist.dll people.rtf roadshow.rtf video.frm winapi31.bas
animate.frm eplist.rtf popcmds.lst roadshw.exe video.txt
animate.txt global.bas reallife.rtf sites.rtf viewer.bas
bumble.mwf global.txt roadshow.mak startup.frm viewer.txt
control.frm listitem.lst roadshow.mvb startup.txt winapi.bas
control.txt media.rtf roadshow.mvp vbroad.exe winapi30.bas

source\snapshot*.*
chklist.ms snapshot.def* snapshot.ico* snapshot.rc*
makefile snapshot.exe* snapshot.map* snapshot.res*
snapshot.c snapshot.h* snapshot.obj* snapshot.sym*

source\sprites*.*
[dibs] cmacro32.inc dib.drv* help.hpj* sprite.c*
[scenes] d.bat draw.c* init.c* sprites.exe*
app.def debug.c* fast16.asm* main.c* sprites.hlp*
app.ico dialogs.dlg* fast32.asm* makefile* sprites.ini*
app.rc dialogs.h* file.c* palette.c*
bkgnd.c dialogs.res* global.h* ppdbg.asm*
chklist.ms dib.c* help.c* print.c*

source\sprites\dibs*.*
bigbalon.dib donut.dib* fence.dib* olivia.dib* smnell.dib*
bigcloud.dib duck1.dib* grass.dib* pnkflwr.dib* smolivia.dib*
bkgnd.dib duck2.dib* herman.dib* redflwr.dib* sun.dib*
cloud16.dib duck3.dib* moon.dib* rlefence.dib*
computer.dib duck4.dib* nell.dib* rlegrass.dib*

source\sprites\scenes*.*
cloud.ini clouds.ini* dog.ini* garden.ini*

source\transblt*.*
cat.bmp pawsmask.bmp* tblt.obj* transblt.ico* transblt.res*
catmask.bmp sampver.h* transblt.c* transblt.map* transblt.sym*
chklist.ms sampver.ver* transblt.def* transblt.obj*
makefile tblt.c* transblt.exe* transblt.rc*
paws.bmp tblt.h* transblt.h* transblt.rcv*

source\triq*.*
[4d] gmem.h tri.exe* triq.dlg* triqrc.h*
ball.3d makefile* tri.ico* triq.exe* view.h*
chklist.ms parse.c* tri.rc* triq.h*
dib.c parse.obj* triangle.asm* triq.ico*

dib.h tri.c* triq.c* triq.obj*
dib.obj* tri.def* triq.def* triq.rc*

source\triq\4d*.*
4d.c 4d.obj* math.h* transfor.h*
4d.h* makefile* math.obj* transfor.obj*
4d.lib* math.asm* transfor.c* vttable.inc*

source\waveconv*.*
chklist.ms muldiv32.asm* waveconv.c* waveconv.rc* wavesup.h*
makefile* pcm.c* waveconv.def* waveconv.rcv* wavever.h*
msadpcm.c* pcm.h* waveconv.exe* waveio.c* wavever.ver*
msadpcm.h* riffsup.c* waveconv.h* waveio.h*
msvc.pdb* riffsup.h* waveconv.ico* wavesup.c*

source\wavemix*.*
1.wav 4.wav* 7.wav* mixtest.def* mixtest.rc* wavemix.ini*
2.wav* 5.wav* makefile* mixtest.exe* wavemix.dll* wavemix.lib*
3.wav* 6.wav* mixtest.c* mixtest.h* wavemix.h* wavemix.txt*

This is a list of all the files in the Develpmt\Systems Directory:

systems\viewer2*.*
ctl3d.dll mvbmp2.dll* mvfs2.dll* mviewer2.exe* mvsrch2.dll*
mvapi2.dll* mvbrkr2.dll* mvftsui2.dll* mvmci2.dll* mvtitle2.dll*

This is a list of all the files in the Develpmt\Tech_Art Directory

tech_art\adobe*.*
setup.doc

tech_art\animat*.*
[rleapp] [sprites]

tech_art\animat\rleapp*.*
rle_samp.doc

tech_art\animat\sprites*.*
sprites.doc

tech_art\audio*.*
[waveconv]

tech_art\audio\waveconv*.*
wavecon.doc

tech_art\bitmap*.*
bitmap.doc

tech_art\cdrom*.*
*[docs] optcdrom.doc quirks.doc readme2.txt**

tech_art\cdrom\docs*.*
cd_ext1.doc device.doc kanji1.doc netinfo.doc speed.doc
cdromify.doc example.doc mctrl.doc overview.doc testdrv.doc
contents.doc install.doc msdosify.doc qna.doc

tech_art\device*.*
pfm.doc print.doc

tech_art\mapmodes*.*
fontmap.doc rghmap.doc

tech_art\postscri*.*
descrip.doc epspfm.doc print.doc

tech_art\scan*.*
layout.doc scan_inf.doc scan_iss.doc scan_tls.doc

tech_art\truetype*.*
dig_typ.doc truetype.doc

This is a list of all the files in the Develpmt\Tools Directory:

tools\bin*.*
*riffwalk.exe**

tools\convert*.*
convert.exe mediaman.dll readme.txt wrkbench.dll*
convert.hlp mmtools.ini wincom.dll

tools\upfront*.*
buttons.dll ctl3d.dll* upfront.dat* upfront.exe* upfront.hlp**

INFO

*mmag.mvb**

MTKG

nytime2.ppt readme.txt

PPT

mmroad2.ppt readme.txt

PPT_VIEW

pptview.exe

TOUR

tour\media*.*
cdtour.ico sample.avi sample.awa sample.mid sample.wav

source\docs*.*
cdtour.rtf

source\graphics*.*
allpics.pal boxvis.bmp bull.bmp control.shg icovfw.bmp titencar.bmp
bkgapp.bmp boxwss.bmp butjump.bmp control1.bmp icovis.bmp tithand.bmp
bkgintro.bmp bug1bs.bmp butminus.bmp control2.bmp icovis2.bmp tithype.bmp
bkgmain.bmp bug1bs.shg butplus.bmp control3.bmp icovis3.bmp titmmcd.bmp
boxbook.bmp bug1main.bmp butsbs.bmp control3.shg icowss.bmp titmmpc.bmp
boxcine.bmp bug1main.shg butsbs.shg icobook.bmp main.bmp titmsdn.bmp
boxencar.bmp bug1min1.bmp butsmain.bmp icocine.bmp mainbuts.bmp titvfw.bmp
boxhand.bmp bug1min1.shg butsmain.shg icoencar.bmp mainbuts.shg titvis.bmp
boxhype.bmp bug2main.bmp butsmin1.bmp icohand.bmp mainrend.bmp titwss.bmp
boxmmcd.bmp bug2main.shg butsmin1.shg icohype.bmp maintxt.bmp vicon.bmp
boxmmpc.bmp bug2min1.bmp butsnav.bmp icommcd.bmp titbits.bmp
boxmsdn.bmp bug2min1.shg butsnav.shg icommpc.bmp titbook.bmp
boxvfw.bmp bugmin2.bmp control.bmp icomsdn.bmp titcine.bmp

ANIM
card16m.awa companyn.flc* endfire.awa* jungle.awa**

AUDIO
16_44_s*.*
jtfly_44.wav lstcl_44.wav**

8_22_m*.*
goodv2m3.wav guitblus.wav* spring.wav**

8_22_s*.*
*bach4th.wav**

8_44_m*.*
glass_44.wav sonar_44.wav**

8_44_s*.*
sail_44.wav space_44.wav**

GRAPHICS
v-osaka*.*
bike2.bmp car3.bmp* comptr.bmp* dinning.bmp* drag.bmp* spiral1.bmp*
bike3.bmp* chopper.bmp* cover.bmp* dispbmp.exe* lobby.bmp* stand.bmp*
bottle.bmp* coffee.bmp* daisy.bmp* dispbmp.ini* speaker.bmp* wheels2.bmp*

VIDEO
funstuff*.*
*wtrski.avi**

misc*.*
cvlmvt.avi *earth.avi* *funplay.avi**

pubsgr*.*
t040370a.avi t040372a.avi* t040373a.avi**

End.

Welcome to the JumpStart CD-ROM!

Hello fellow programmers, developers and multimedia enthusiasts! All of us at Microsoft's Multimedia Developer Relations Group hope you find this CD both informative and useful as a multimedia development aid.

In JumpStart, you will find useful development tools like the Video For Windows 1.1. SDK, sample applications, sample media and a host of technical articles full of useful information and sample code.

Navigating around JumpStart is easy. Simply click on the following buttons on the controller for the following information:

WELCOME:	Brings up the screen you are viewing now
DEVELOPMENT	This button lists all the development articles on the CD. Simply scroll down the list you see in the controller window and click on anything that interests you.
SAMPLE APPS	The Sample Apps button references many of the sample applications on the CD. Many other sample applications can be accessed by browsing through the Sample Apps directory directly off the CD.
SAMPLE MEDIA	This button references some of the sample media clips available on JumpStart. Again, make sure you browse through the CD directory as well to find more clips to sample.
MISC.	We added the Miscellaneous button to place things like marketing material and slide presentations, JumpStart credits, licensing agreements etc.
<<<< AND **>>>>**	These buttons allow you to browse through related topics you see in the window of the controller.
SEARCH	If you can't find what your looking for, try the full text search feature of JumpStart. This useful feature is available when you author titles with Microsoft Multimedia Viewer.
EXIT	Exits the application.
HELP	The help button will provide a map of the directories on JumpStart. Click on any of the directories to find information on what they contain.
ALWAYS IN FRONT ON/OFF	This toggles between the controller being always in front or, if you wish it not to be viewed, send it behind. Pressing Alt-Tab will bring the controller back to the foreground.

Well, we hope this gets you started. Please send us your comments via the Compuserve WINMM forum and let us know what else you would like to see on our next version of JumpStart due in early 1994.

End.

{ewl MVBMP2, ViewerBmp2, firstbmp.bmp}

Jumpstart 1.1 Credits

Steve Molstad	*Programming Lead*
Donna Scott	*Project Manager*
William Vablais	*Program Manager*
Rick Segal	*Person In Charge*
Steve Banfield	*Critical Analysis*
Chris Lye	*Critical Analysis*
David Feinlieb	*Hotspots & Sprites*
Nadine Kano	*Internationalisation Docs and Lists*
Melanie Schacht	*Administrative support*
Dennis Crain	*Consultancy (graphics and software gotchas)*
Chris Rimple	*CDROM Mastering*
John Sayler **Steve Holgrove**	*Video Guys (filmed the hotspot scenes)*
Wayne Radinsky	*Hotspot modifications and development*
Alex St. John	*Loan of Spike the "Green Monster"*
The guys at WesDesign	*Look and feel of the Publishers program design*

End.