1.

2.Unicode

3.

4.

5.

……

6.

……
……
……
……
……

7.

……
……
……
……
……
……

8.

……
……
……
……

9.

……
……
……
……
……
……

10.

……

……

……

11.

……

……

……

12.

……

……

……

13.

……

……

14. Bitblt

……

……

……

…… GDI

15.

…… DIB

……

…… DIBDDB

16.

……
……
……
…… DIB

17.

……
……
……
……
……
……

18. Metafile

…… MetaFile
…… MetaFile

19.

…… MDI
…… MDI

20.

……
…… WINDOWS
……
……
…… TLS

21.

……
…… DLL

22.

…… WINDOWS
……
…… MIDI

23. Internet

…… Windows Sockets
…… WININETFTP

Microsoft Windows 98Microsoft Windows NT 4.0Windows NT 5
Windows Application Programming InterfaceAPIWindows
Windows API

Windows 98Intel 32486PentiumIBMWindows
RISCWindows

Windows 98WindowsWindo

CWindowsCWindows
CC

Windows32CMicrosoft

Windows

WINDOWS

WindowsWindowsWindows

Windows

1981IBM PCMS-DOSPCMS-DOSMicrosoft
MS-DOSMS-DOSDIRTYPEI/O

Lisa1983119841MacintoshMacintoshMacintosh
WindowsXerox Palo Alto Research CenterPARC70

Windows198311LisaMacintosh198511Microsoft
Windows1.0

Windows2.0198711Windows

WindowsIntel 808680881MBWindows/386Windows
2.0Intel 3868086MS-DOSWindows2.1Windows/286

Windows 3.01990522Windows/286Windows/386Windows
286386486WindowsWindows16MBWindowsWindows

WindowsOS/2OS/2DOSWindowsMicrosoftIBMOS/21.0Intel
2861987198810OS/21.1PMPresentation
WindowsAPI

19909IBMMicrosoftIBMOS/2MicrosoftWindowsOS/2
Windows

Microsoft Windows3.119924TrueTypeWindows))OLE
Object Linking and EmbeddingOS/2Windows 3.11MB28(

19937Windows NTIntel 386486Pentium32WindowsWind
)Windows NTIntelRISC

Windows 9519958Windows NTWindows 95Intel 38
NTRISCWindows 95

Windows 9819986WWW

Windows

Windows 98Windows NT32preemptive multitaskingV
GUI70AltoStarSmallTalkXerox
MicrosoftGUIMicrosoftCharles

GUIWYSIWYGwhat

WindowsWindowsWindows

WindowsWindowsWindowsWindows

WindowsWindows

WindowsWindowsWindows

Windowsnon-preemptiveWindowsWindows
98

8088Windows 1.0Windows
Windows 2.0WindowsEMSWindows 3.0Windows1(
NTWindows 9832

WindowsWindowsWindows

WindowsWindows

WindowsWindowsGDIWindowsWindowsWindows

WindowsIBM PCPCPCPCMS-DOSMS·
WindowsWindows

WindowsWindowsDynamic
.DLL.EXEWindows 98\WINDOWS\SYSTEMWindo
\WINNT\SYSTEM\WINNT\SYSTEM32

WindowsWindowsKernelUserGDIWindowsWindows
Kernel16KRNL386.EXE32KERNEL32.DLLI/OUser16
USER.EXE32USER32.DLLGDI16GDI.EXE32GDI32.DLL

Windows 98CreateWindowWindows

WindowsWindowsstrlenCCWindowsDLL

WindowsWindowsWindows.EXEWindowsDLL
DLL

Windowsimport
Windows

WINDOWS

WindowsCWindows
Windows

API

APIAPIWindowsAPI

Windows APIWindows 1.0Windows
APIWindows 1.0450

Windows API1632Windows1.03.116Intel
Intel16Cint1616segment16offsetlongfar
shortnear

Windows NTWindows 95WindowsIntel 386486Pentium3232Ci
32Windows32

16WindowsAPIWindows 1.0Windows 3.1Win1632Windows
APIWindows 95Windows 98Windows NTWin32Win16Win32
Win1616Win3232Win1632Win32

32WindowsWin16 APIWin32 API
95Windows 98Windows NTWin16Win32Windows
98Win32Win16

Windows APIWin32s ssubsetAPIWindo
APIWin1632Windows 95 APIWin32cccor

Windows NTWindows 98Win32 API

CAPIWindows 98WindowsWindows

WindowsAPIWindows

WindowsWindowsCAPIWindowsMicrosoft
DelphiPascalVisual
Halvorson1996Learn Visual Basic Now

Microsoft Visual C++M
C++WindowsJeff ProsiseProgramming V
Microsoft Press,1999MFC

InternetWorld Wide WebSun MicrosystemsJavaC++Micr
PressMicrosoft J++MicrosoftJavaProgramming Visual J++
1998Stephen R. Davis

WindowsWindows
API

Windows APIVisual

MFCOLEDocument/ViewMFCWindows
MFCMFCWindows

Microsoft Visual C++ 6.0Visual

Microsoft Visual C++ CWindowsVisual

Visual C++ 5.0Windows 98Windows NT 5.0Microsoft
http://www.microsoft.com/msdn/**Downloads** **Platform SDK**
Microsoft Developer Studio **Tool** **OptionsDirect**

MicrosoftmsdnMicrosoft Developer NetworkMicrosoftCD-RO
CD-ROMWindowsMSDNMicrosoft

API

Windows APICD-ROMInternet

Visual C++ 6.0APIMSDNMicrosoft
http://www.microsoft.com/msdn/**MSDN Library Online**

Visual C++ 6.0 **HelpContents**MSDNAPI
Platform SDK Platform SDKM
/ Platform SDK / User Interface Services / User Input / Mouse Input

WindowsKernelUserGDIkernel/ Platform SI
User / Platform SDK / User Interface ServicesGDI / Platform SDK
Graphics and Multimedia Services / GDI

WINDOWS

Windows

Character-Mode

The C Programming LanguagePrentice Hall19781988
KernighanDennis M. RitchieK&Rhello,

The C Programming Language6

```
main ()

{

        printf ("hello, world\n") ;

}
```

CprintfC90K&R

```
#include <stdio.h>

main ()
```

```
{
        printf ("hello, world\n") ;
}
```

mainANSI

```
#include <stdio.h>
int main ()
{
        printf ("hello, world\n") ;
        return 0 ;
}
```

mainincludereturn

Windows

Windowshello, worldincludereturn

```
/*------------------------------------------------------------
HelloMsg.c -- Displays "Hello, Windows 98!" in a message box
   (c) Charles Petzold, 1998
------------------------------------------------------------*/
#include <windows.h>
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
```

```
                PSTR szCmdLine, int iCmdShow)
{
MessageBox (NULL, TEXT ("Hello, Windows 98!"), TEXT ("HelloMs
return 0 ;
}
```

Visual C++ Developer Studio

FileNew New Projects Win32 ApplicationLocation Name HelloMsg Location Create New Workspace Platfo Win32OK

Win32 Application - Step 1 Of 1 Empty ProjectFinish

FileNew New Files C++ Source FileAdd To Project HelloM NameHelloMsg.cOK

HELLOMSG.C **Insert File As Text**CD

HELLOMSG.CK&Rhello,worldSTDIO.HWINDOWS.Hmain WinMainCprintfWindows APIMessageBox

HELLOMSG.CCWindows

```
#include <windows.h>
```

WINDOWS.HWindows

- WINDEF.H

- WINNT.H Unicode
- WINBASE.H Kernel
- WINUSER.H
- WINGDI.H

WindowsWindowsVisual
FilesDeveloper Studio

CmainWindowsWinMain

```
int WINAPI WinMain       HINSTANCE hInstance,HINSTANCE hPrev

        PSTR szCmdLine,int iCmdShow
```

/ Platform SDK / User Interface Services / Windowing / Windows / Wir
Reference / Window FunctionsWINBASE.H

```
int

WINAPI

WinMain

   HINSTANCE hInstance,

   HINSTANCE hPrevInstance,

   LPSTR lpCmdLine,

   int nShowCmd
```

```
;
```

HELLOMSG.CWINBASE.HLPSTRPSTRWINNT.HLP
16Windows

WinMainWindowsiintsz

WinMainintWINAPIWINDEF.H

```
#define WINAPI __stdcall
```

WindowsWINAPI

WinMainWindowsWindowsWindows
multiple instanceshPrevInstance

32WindowsWinMainNULL0

WinMainWindowsWinMain

MessageBox

MessageBoxMessageBox

MessageBoxHELLMSG.CTEXTTEXTUnicode

MessageBoxWINUSER.HMB_

```
#define    MB_OK                    0x00000000L
#define    MB_OKCANCEL                  0x0000000
#define    MB_ABORTRETRYIGNORE              0x000
#define    MB_YESNOCANCEL                0x000000
```

```
#define    MB_YESNO                    0x00000004L
#define    MB_RETRYCANCEL              0x000000
```

HELLOMSG0 **OK**COR|

```
#define    MB_DEFBUTTON1               0x000000
#define    MB_DEFBUTTON2               0x000001
#define    MB_DEFBUTTON3               0x000002
#define    MB_DEFBUTTON4               0x000003
```

```
#define    MB_ICONHAND                 0x00000010L
#define    MB_ICONQUESTION             0x00000020L
#define    MB_ICONEXCLAMATION          0x00000030L
#define    MB_ICONASTERISK             0x00000040L
```

```
#define    MB_ICONWARNING              MB_ICONEXCLAMATION
#define    MB_ICONERROR                MB_ICONHAND
#define    MB_ICONINFORMATION          MB_ICONASTERISK
#define    MB_ICONSTOP                 MB_ICONHAND
```

MB_ / Platform SDK / User Interfa
Boxes / Dialog Box Reference / Dialog Box Functions

MessageBox1IDOKIDOKWINUSER.H1MessageBox
IDYESIDNOIDCANCELIDABORT IDRETRYIDIGNORE

WindowsK&Rhello, worldMessageBoxhello,
printfMessageBox

HELLOMSG **Build** **Build Hellomsg.exe** **F**
Build **ToolsCustomize** **Toolbars**

BuildExecute Hellomsg.exe **Ctrl+F5** **BuildExe**

C.OBJ.OBJ.LIB.EXE
Settings **Link**KERNEL32.LIBUSER32.LIBGDI32.LIBWindows
.EXEWindowsKERNEL32.DLLUSER32.DLLGDI32.DLL

Visual C++ Developer StudioDebugReleaseDebug
.EXE

CD-ROM.MAKmake
StudiomakeDeveloper StudioBI
HELLOMSG

```
NMAKE /f HelloMsg.mak CFG="HelloMsg - Win32 Debug"
```

```
NMAKE /f HelloMsg.mak CFG="HelloMsg - Win32 Release"
```

```
DEBUG\HELLOMSG
```

RELEASE\HELLOMSG

.EXE

CD-ROMDebug **Project**
UNICODE

Unicode

CMicrosoft WindowsUnicode

UnicodeASCIIASCII78Unicode16Unicode
UnicodeASCIIASCII

UnicodeWindows
ANSICUnicode

UnicodeUnicode

6000300019Samuel

Morse18211824Louis
shift

TelexBaudot 1903CCITT

HollerithHerman
Decimal Interchange CodeHollerith608EBCDICIBM

ASCIIAmerican Standard Code for Information Interchange
ASCII678ASCII6826261032
33128ASCIIANSI X3.4-198677-Bit
National Standard Code for Information InterchangeAmerican National
Standards Institute2-1ASCIIANSI

ASCII26EBCDIC10BCDIC09

ASCIIInternetASCII

	0-	1-	2-	3-	4-	5-
-0	NUL	DLE	SP	0	@	P
-1	SOH	DC1	!	1	A	Q
-2	STX	DC2	"	2	B	R
-3	ETX	DC3	#	3	C	S
-4	EOT	DC4	$	4	D	T
-5	ENQ	NAK	%	5	E	U
-6	ACK	SYN	&	6	F	V
-7	BEL	ETB	'	7	G	W
-8	BS	CAN	(	8	H	X
-9	HT	EM	)	9	I	Y
-A	LF	SUB	*	:	J	Z
-B	VT	ESC	+	;	K	[
-C	FF	FS	,	<	L	\
-D	CR	GS	-	=	M	]
-E	SO	RS	.	>	N	^
-F	SI	US	/	?	O	_

2-1 ASCII

ASCIIASCII

cperaterésumé

ASCII19671967ISOInternational
ASCII0x400x5B0x5C0x5D0x7B0x7C0x7D0x5E0x60
0x7E8910

ASCII

8128ASCII1981IBM

IBMASCII

IBMROMWindowsWindows

Windows 1.0198511MicrosoftIBMANSIISOWindows
ANSIANSIISOANSI/ISO 8859-1-1987American N
for Information Processing-8-Bit Single-Byte Coded Graphic Character Sets-
Part 1: Latin Alphabet No 1Latin 1

Windows 1.0Programmer's ReferenceANSI2-2

	0-	1-	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
0	□	□		0	@	P	`	p	□	□		°	À	Ð	à	ð
-1	□	□	!	1	A	Q	a	q	□	□	¡	±	Á	Ñ	á	ñ
-2	□	□	"	2	B	R	b	r	□	□	¢	²	Â	Ò	â	ò
-3	□	□	#	3	C	S	c	s	□	□	£	³	Ã	Ó	ã	ó
-4	□	□	$	4	D	T	d	t	□	□	¤	´	Ä	Ô	ä	ô
-5	□	□	%	5	E	U	e	u	□	□	¥	µ	Å	Õ	å	õ
-6	□	□	&	6	F	V	f	v	□	□	¦	¶	Æ	Ö	æ	ö
-7	□	□	'	7	G	W	g	w	□	□	§	·	Ç	□	ç	□
-8	□	□	(	8	H	X	h	x	□	□	¨	¸	È	Ø	è	ø
-9	□	□	)	9	I	Y	I	y	□	□	©	¹	É	Ù	é	ù
-A	□	□	*	:	J	Z	j	z	□	□	ª	º	Ê	Ú	ê	ú
-B	□	□	+	;	K	[	k	{	□	□	«	»	Ë	Û	ë	û
-C	□	□	,	<	L	\	l	|	□	□	¬	¼	Ì	Ü	ì	ü
-D	□	□	-	=	M	]	m	}	□	□	-	½	Í	Ý	í	ý
-E	□	□	.	>	N	^	n	~	□	□	®	¾	Î	Þ	î	þ
-F	□	□	/	?	O	_	o	DEL	□	□	¯	¿	Ï	ß	ï	ÿ

2-2 Windows ANSIANSI/ISO 8859-1

ANSI/ISO 8859-1ANSI/ISO 8859-
0xADANSI/ISO
Windows0x800x9FANSI/ISO 8859-

MS-DOS 3.319874IBM PCcode pageWindowsIB
MS-DOS Latin US850MS-DOS Latin 1
128128

MS-DOSPCPC

128

MS-DOS855Windows1251Macintosh10007IBM

25621,000ASCII

DBCSdouble-byte character setI
ASCII128

Windows932936949950
WindowsDBCS

ASCII1DBCS

Unicode

2568DBCS

816Unicode25612Unicode1665,536

UnicodeDBCSUnicodeCUnicode168Unicode8
8

DBCSUnicode128Unicode160x00000x007FASCII128Unicode
0x00800x00FFISO 8859-1ASCIIUnicode0x03700x03FF
0x04000x04FF0x05300x058F0x05900x05FFCJK0x3000
0x9FFF

UnicodeUnicodeISO
Version 2.0Addison-Wesley1996Unicode

UnicodeUnicodeASCIIUnicode

C

C16charANSI/ISO

ANSI CWindowsC

UnicodeUnicodeWindowsCUnicode

Char

CcharCWin32

```
char c = 'A' ;
```

c10x41AASCII

```
char * p ;
```

Windows32p4

```
char * p = "Hello!" ;
```

p47610

```
char a[10] ;
```

10sizeofa10

```
char a[] = "Hello!" ;
```

static

```
static char a[] = "Hello!" ;
```

07

UnicodecharCchar1sizeof

Cwchar_tWCHAR.H

```
typedef unsigned short wchar_t ;
```

wchar_t16

```
wchar_t c = 'A' ;
```

c0x0041UnicodeAIntel0x410x00Unicode

```
wchar_t * p = L"Hello!" ;
```

Llong2p414202

```
static wchar_t a[] = L"Hello!" ;
```

14sizeof (a) 14aa[1]

LL2LLC

L

```
wchar_t c = L'A' ;
```

C

```
char * pc = "Hello!" ;
```

```
iLength = strlen (pc) ;
```

iLength6

```
wchar_t * pw = L"Hello!" ;
```

strlen

```
iLength = strlen (pw) ;
```

C

'function' : incompatible types - from 'unsigned short *' to 'const char *'

strlencharunsigned

Hello!616

```
0x0048 0x0065 0x006C 0x006C 0x006F 0x0021
```

Intel

```
48 00 65 00 6C 00 6C 00 6F 00 21 00
```

strlen10

CL"Hello!"

CC

strlenwcslenwide-character string lengthSTRING.Hstrlen WCHAR.Hstrlen

```
size_t __cdecl strlen (const char *) ;
```

wcslen

```
size_t __cdecl wcslen (const wchar_t *) ;
```

```
iLength = wcslen (pw) ;
```

6

CwprintfprintfWCHAR.H

UnicodeASCIIUnicodeASCIIUnicode

L

Microsoft Visual C++TCHAR.HANSI

_tcslenUnicodeUnicode

_UNICODETCHAR.H_tcslenwcslen

```
#define _tcslen wcslen
```

UNICODE_tcslenstrlen

```
#define _tcslen strlen
```

TCHAR.HTCHAR_UNICODETCHARwchar_t

```
typedef wchar_t TCHAR ;
```

TCHARChar

```
typedef char TCHAR ;
```

L_UNICODE__T

```
#define __T(x) L##x
```

ANSI Ctoken

_UNICODE__T

```
#define __T(x) x
```

__T

```
#define _Tx__Tx
#define _TEXTx__Tx
```

Win32 console_T_TEXT

```
_TEXT ("Hello!")
```

_UNICODE8

Windows

Windows NTUnicodeWindows NT1616Windows ASCIIUnicodeASCIIUnicodeWindows NTAPI816

Windows NTWindows 98UnicodeWindows 98
Base article Q125671MessageBox.EXEWindows UnicodeWindows 98UnicodeWindowsUnicodeA
Unicode

Windows

WindowsWINDOWS.HWINDEF.HWindowsWINNT.H WINNT.HUnicode

WINNT.HCCTYPE.HCwchar_tWINNT.HCHARWCHAR

```
typedef char CHAR ;
typedef wchar_t WCHAR ;    // wc
```

816WindowsCHARWCHARWCHARWCHARwc

WINNT.H8const 8

```
typedef CHAR * PCHAR, * LPCH, * PCH, * NPSTR, * LPSTR, * PSTR
typedef CONST CHAR * LPCCH, * PCCH, * LPCSTR, * PCSTR ;
```

NLnearlong16WindowsWin32nearlong

WINNT.H16const 16

```
typedef WCHAR * PWCHAR, * LPWCH, * PWCH, * NWPSTR, * LPW
typedef CONST WCHAR * LPCWCH, * PCWCH, * LPCWSTR, * PCW
```

CHAR8charWCHAR16wchar_tCHARWCHARTCHAR.H WINNT.HTCHARUNICODETCHARTCHARWCHAR

WCHARUNICODETCHARTCHARcharchar

```
#ifdef  UNICODE
typedef WCHAR TCHAR, * PTCHAR ;
typedef LPWSTR LPTCH, PTCH, PTSTR, LPTSTR ;
typedef LPCWSTR LPCTSTR ;
#else
typedef char TCHAR, * PTCHAR ;
typedef LPSTR LPTCH, PTCH, PTSTR, LPTSTR ;
typedef LPCSTR LPCTSTR ;
#endif
```

TCHARWINNT.HWCHAR.HWINDOWS.H

WINNT.HLUNICODE

```
#define __TEXT(quote) L##quote
```

UNICODE__TEXT

```
#define __TEXT(quote) quote
```

TEXT

```
#define TEXT(quote) __TEXT(quote)
```

TCHAR.H_TEXTTEXT

ASCIIUnicodeASCIIUnicode8CHARPCHAR16
WCHARPWCHARL816UNICODETCHARPTCHARTEXT

Windows

Windows 1.0Windows 3.116WindowsMessageBoxUSER.EXEWindows 3.1WINDOWS.HMessageBox

```
int WINAPI MessageBox (HWND, LPCSTR, LPCSTR, UINT) ;
```

Win16WindowsMessageBox.EXEWindowsUSER
MessageBox

32WindowsWindows NTWindows 95Windows 9816US
USER32.DLL3232MessageBox

WindowsUnicodeUSER32.DLL32MessageBox
MessageBoxAASCIIMessageBoxWWin32
MessageBoxTCHARWindows

MessageBoxAWINUSER.HMessageBox

```
WINUSERAPI int WINAPI MessageBoxA (HWND hWnd, LPCSTR lpT
                LPCSTR lpCaption, UINT uType) ;
```

MessageBoxW

```
WINUSERAPI int WINAPI MessageBoxW (HWND hWnd, LPCWSTR

                LPCWSTR lpCaption, UINT uType) ;
```

MessageBoxW

ASCIIWindowsMessageBoxAMessageBoxWMessageBox
UNICODEMessageBoxMessageBoxAMessageBoxWWINUSER.H

```
#ifdef UNICODE

#define MessageBox  MessageBoxW

#else
```

```
#define MessageBox  MessageBoxA

#endif
```

UNICODEMessageBoxMessageBoxWMessageBoxA

WindowsWindowsWindows

Windows

Microsoft CCWindowsWindows

```
lLength = lstrlen (pString) ;

pString = lstrcpy (pString1, pString2) ;

pString = lstrcpyn (pString1, pString2, iCount) ;

pString = lstrcat (pString1, pString2) ;

iComp = lstrcmp (pString1, pString2) ;

iComp = lstrcmpi (pString1, pString2) ;
```

CUNICODElstrlenWWindows

Windowsprintf

CprintfputsprintfKernighanRitchiehello,
hello, worldprintf

WindowsprintfWindowsCCI/OWindowsWindowsWindows
fprintfprintf

sprintfsprintfprintfIMessageBox

sprintf Windowsprintf

```
int printf (const char * szFormat, ...) ;
```

sprintf

```
int sprintf (char * szBuffer, const char * szFormat, ...) ;
```

SprintfszBuffer

```
printf ("The sum of %i and %i is %i", 5, 3, 5+3) ;
```

```
char szBuffer [100] ;

sprintf (szBuffer, "The sum of %i and %i is %i", 5, 3, 5+3) ;

puts (szBuffer) ;
```

WindowsMessageBoxputs

printfsprintfMicrosoft_snprintf

vsprintfsprintfvsprintfprintfvsprintfsprintf
va_listva_startva_endSTDARG.HSCRNSIZEvsprintf
sprintf

```
int sprintf (char * szBuffer, const char * szFormat, ...)

{
```

```
    int     iReturn ;
    va_list pArgs ;
    va_start (pArgs, szFormat) ;
    iReturn = vsprintf (szBuffer, szFormat, pArgs) ;
    va_end (pArgs) ;
    return iReturn ;
}
```

va_startpArgszFormat

WindowssprintfvsprintfMicrosoftWindows AI
sprintfvsprintf

sprintf2-1MicrosoftCWindowssprintf

2-1

	ASCII		
	sprintf	swprintf	_stprintf
	_snprintf	_snwprintf	_sntprintf

Windows	wsprintfA	wsprintfW	wsprintf
	vsprintf	vswprintf	_vstprintf
	_vsnprintf	_vsnwprintf	_vsntprintf
Windows	wvsprintfA	wvsprintfW	wvsprintf

sprintf

2-1SCRNSIZEMessageBoxPrintfprintf

```
2-1  SCRNSIZE

SCRNSIZE.C

/*------------------------------------------------------------------------

   SCRNSIZE.C -- Displays screen size in a message box

                (c) Charles Petzold, 1998

------------------------------------------------------------------------*/

#include <windows.h>

#include <tchar.h>
```

```
#include <stdio.h>

int CDECL MessageBoxPrintf (TCHAR * szCaption, TCHAR * szFor

{
   TCHAR   szBuffer [1024] ;
   va_list pArgList ;

   // The va_start macro (defined in STDARG.H) is usually equiva
   // pArgList = (char *) &szFormat + sizeof (szFormat) ;

   va_start (pArgList, szFormat) ;

   // The last argument to wvsprintf points to the arguments

   _vsntprintf ( szBuffer, sizeof (szBuffer) / sizeof (TCHAR),
                 szFormat, pArgList) ;

   // The va_end macro just zeroes out pArgList for no good reas
   va_end (pArgList) ;
   return MessageBox (NULL, szBuffer, szCaption, 0) ;
}

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
```

```
                PSTR szCmdLine, int iCmdShow)
{
   int cxScreen, cyScreen ;
   cxScreen = GetSystemMetrics (SM_CXSCREEN) ;
   cyScreen = GetSystemMetrics (SM_CYSCREEN) ;

   MessageBoxPrintf (    TEXT ("ScrnSize"),
                TEXT ("The screen is %i pixels wide by %i pixels high
                cxScreen, cyScreen) ;
   return 0 ;
}
```

GetSystemMetricsGetSystemMetricsWindows
Windows

WindowsUnicodeNadine
Windows 95 and Windows NTMicrosoft Press1995

UNICODETCHARTEXTSCRNSIZE
sizeof szBuffersizeofTCHAR

Visual C++ Developer StudioDebugReleaseDebug
UNICODEC_UNICODEDebugProjectSettings
C/C++

UnicodeWindows NTUnicodeWindows
MessageBoxWWindows 98WindowsSCRNSIZE.CWindowswprintf
_vsntprintfSCRNSIZE.CUnicodeWindows
wprintfW

WindowsWindowsUnicodeWindowsUnicode
Unicode

MessageBoxMessageBoxWindows

MessageBoxWindows

MessageBox

CreateWindow

CreateWindow/Platform SDK/User I
Services/Windowing/Windows/Window Reference/Window Functions
CreateWindowCreateWindow

WindowsOOPWindowsWindowsWindows

Windows

WindowsWindowsWindows

WindowsWindow

Windows

WindowsWindowsWindows

Windows

WindowsWindows

WindowsWindows

Windows

WindowsWindows

Windows

HELLOWIN

Windows3-1HELLOWIN

3-1　HELLOWIN

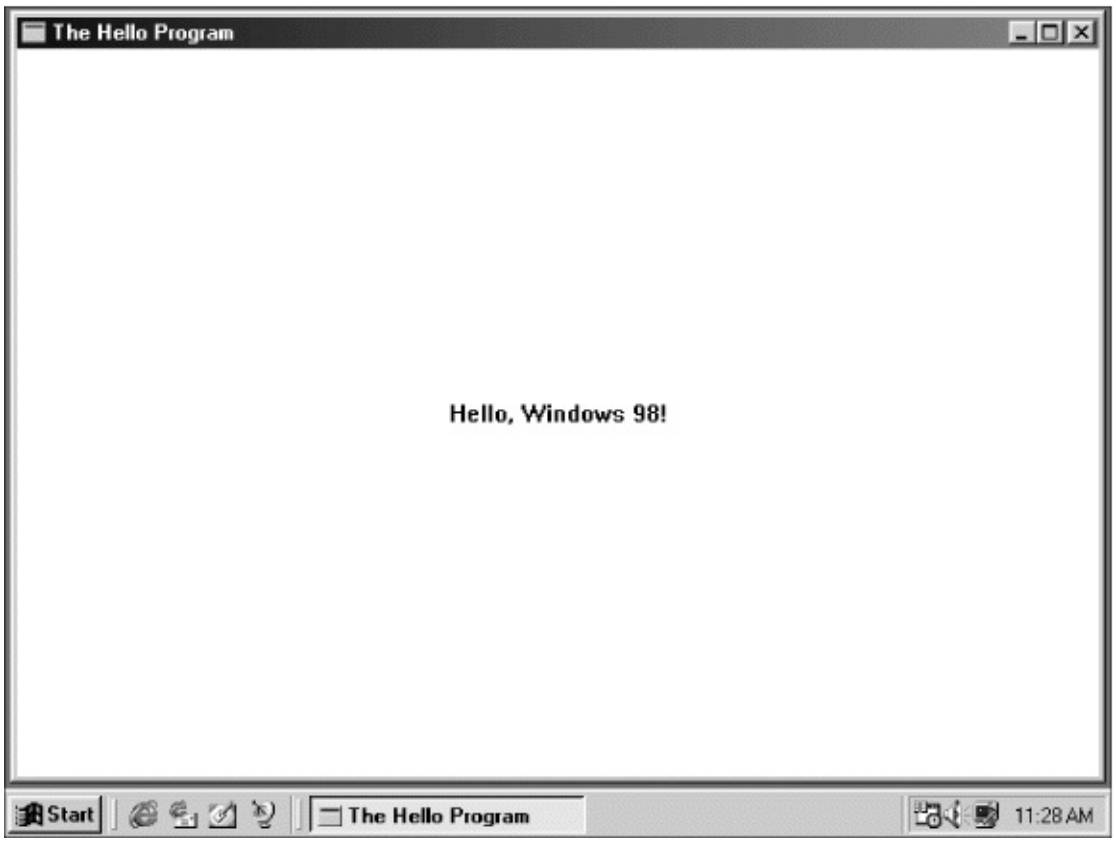
```
HELLOWIN.C
/*------------------------------------------------------------------
  HELLOWIN.C -- Displays "Hello, Windows 98!" in client area
                (c) Charles Petzold, 1998
------------------------------------------------------------------*/
#include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                    PSTR szCmdLine, int iCmdShow)
```

```
{
   static TCHAR szAppName[] = TEXT ("HelloWin") ;
   HWND   hwnd ;
   MSG    msg ;
   WNDCLASwndclass ;

  wndclass.style        = CS_HREDRAW | CS_VREDRAW ;
  wndclass.lpfnWndProc  = WndProc ;
   wndclass.cbClsExtra   = 0 ;
   wndclass.cbWndExtra   = 0 ;
   wndclass.hInstance    = hInstance ;
   wndclass.hIcon        = LoadIcon (NULL, IDI_APPLICATION) ;
 wndclass.hCursor      = LoadCursor (NULL, IDC_ARROW) ;
  wndclass.hbrBackground= (HBRUSH) GetStockObject (WHITE_
 wndclass.lpszMenuNam  = NULL ;
   wndclass.lpszClassName= szAppName ;

   if (!RegisterClass (&wndclass))
   {
```

```
        MessageBox (  NULL, TEXT ("This program requires Windo
                            szAppName, MB_ICONERROR) ;
        return 0 ;
}
hwnd = CreateWindow( szAppName,      // window class name
            TEXT ("The Hello Program"),   // window caption
            WS_OVERLAPPEDWINDOW,  // window style
            CW_USEDEFAULT,// initial x position
            CW_USEDEFAULT,// initial y position
            CW_USEDEFAULT,// initial x size
            CW_USEDEFAULT,// initial y size
            NULL,                 // parent window handle
          NULL,            // window menu handle
          hInstance,   // program instance handle
          NULL) ;      // creation parameters

ShowWindow (hwnd, iCmdShow) ;
UpdateWindow (hwnd) ;
```

```
   while (GetMessage (&msg, NULL, 0, 0))
   {
         TranslateMessage (&msg) ;
        DispatchMessage (&msg) ;
   }
   return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
   HDC                hdc ;
   PAINTSTRUCT ps ;
   RECT         rect ;


   switch (message)
   {
   case WM_CREATE:
   PlaySound (TEXT ("hellowin.wav"), NULL, SND_FILENAME | SN
         return 0 ;
```

```
   case   WM_PAINT:
          hdc = BeginPaint (hwnd, &ps) ;

          GetClientRect (hwnd, &rect) ;

          DrawText (hdc, TEXT ("Hello, Windows 98!"), -1, &rect,
                DT_SINGLELINE | DT_CENTER | DT_VCENTER) ;
               EndPaint (hwnd, &ps) ;
                return 0 ;

   case   WM_DESTROY:
          PostQuitMessage (0) ;
          return 0 ;
   }
  return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```



3-1 HELLOWIN

Microsoft Visual C++
General Object/Library ModulesWINMM.LIBWindows multimedia WindowsHELLOWINPlaySound

HELLOWINHELLOWIN.WAVHELLOWINHELLOWIN.EXE
HELLOWINVisual C++HELLOWINRELEASEDEBUG
HELLOWIN

WindowsHELLOWIN.CWindows

HELLOWIN3-1

80Hello,

HELLOWIN.CWinMainWndProcHELLOWIN.CWndProc
WinMainWndProc

Windows

HELLOWIN18WindowsHELLOWIN

- LoadIcon
- LoadCursor
- GetStockObject
- RegisterClass
- MessageBox
- CreateWindow
- ShowWindow
- UpdateWindow
- GetMessage
- TranslateMessage
- DispatchMessage
- PlaySound

- BeginPaint
- GetClientRect
- DrawText
- EndPaint
- PostQuitMessage
- DefWindowProc

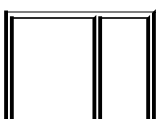
Platform SDKWINUSER.H

HELLOWIN.CWindows

CS_HREDRAW	*DT_VCENTER*	*SND_FILENAME*
CS_VREDRAW	*IDC_ARROW*	*WM_CREATE*
CW_USEDEFAULT	*IDI_APPLICATION*	*WM_DESTROY*
DT_CENTER	*MB_ICONERROR*	*WM_PAINT*
DT_SINGLELINE	*SND_ASYNC*	*WS_OVERLAPPED*

3-1

3-1



CS	
CW	
DT	
IDI	ID
IDC	ID
MB	
SND	
WM	
WS	

WindowsWindows

HELLOWIN.CWindowstypedef#defineWindows1632()

WndProcUINTunsigned
PSTRchar *

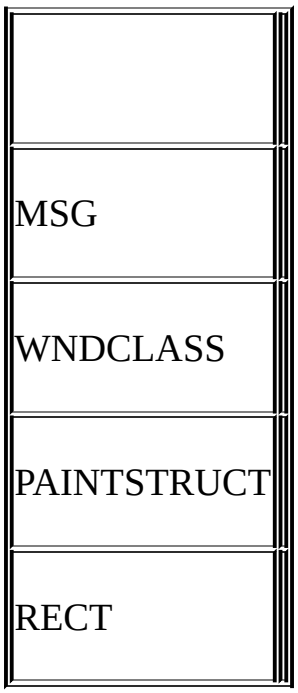
WndProcWPARAMLPARAMWindows16WndProcWORD

16 unsigned shortLONG32PARAMWL32 WindowsWPARAMUINTLPARAMLONGClong32 WORDWindows9816 PARAMW

WndProcLRESULTLONGWinMainWINAPIWindowsWndProc CALLBACK_stdcallWindows

HELLOWINWindows3-2

3-2



WinMainmsgwndclassWndProcpsrect

3-3

3-3

HINSTANCE	
HWND	
HDC	

WindowsHICONHCURSORHBRUSH

32WindowsCMS-DOSWindowsWindows
Windows

HELLOWIN.CszCmdLineWinMain

WindowsMicrosoftCharles
0hInstancehPrevInstancehiCmdShowiWndProc
wParamuiParamWPARAMLPARAM

HELLOWIN.
WNDCLASSEXWndPmcpsPAINTSTRUCTrectRECT

3-4

3-4

c	charWCHARTCHAR
by	BYTE
n	short
i	int
x, y	intxy
cx, cy	intxyC
bf	BOOL (int)f
w	WORD
l	LONG
dw	DWORD
fn	function
s	string
sz	0

h	
p	

Windows

RegisterClassWNDCLASSWINUSER.HASCII
WNDCLASSA

```
typedef struct tagWNDCLASSA
{
   UINT      style ;
   WNDPROC    lpfnWndProc ;
   int       cbClsExtra ;
   int          cbWndExtra ;
   HINSTANCE     hInstance ;
   HICON      hIcon ;
   HCURSOR       hCursor ;
   HBRUSH     hbrBackground ;
   LPCSTR       lpszMenuName ;
```

```
   LPCSTR       lpszClassName ;
}
WNDCLASSA, * PWNDCLASSA, NEAR * NPWNDCLASSA, FAR * LP
```

lpfnWin32
hhbrlpsz0

Unicode

```
typedef struct tagWNDCLASSW
{
        UINT            style ;
        WNDPROC      lpfnWndProc ;
        int          cbClsExtra ;
        int          cbWndExtra ;
        HINSTANCE    hInstance ;
        HICON        hIcon ;
        HCURSOR      hCursor ;
        HBRUSH       hbrBackground ;
        LPCWSTR      lpszMenuName ;
        LPCWSTR      lpszClassName ;
```

```
}

WNDCLASSW, * PWNDCLASSW, NEAR * NPWNDCLASSW, FAR * L
```

ASCII

WINUSER.HWNDCLASSAWNDCLASSWUNICODE
WNDCLASSWNDCLASS

```
#ifdef UNICODE

typedef     WNDCLASSW       WNDCLASS ;

typedef     PWNDCLASSW      PWNDCLASS ;

typedef     NPWNDCLASSW     NPWNDCLASS ;

typedef     LPWNDCLASSW     LPWNDCLASS ;

#else

typedef     WNDCLASSA       WNDCLASS ;

typedef     PWNDCLASSA      PWNDCLASS ;

typedef     NPWNDCLASSA     NPWNDCLASS ;

typedef     LPWNDCLASSA     LPWNDCLASS ;

#endif
```

WNDCLASS

```
typedef struct

```

```
{
  UINT          style ;
   WNDPROC       lpfnWndProc ;
   int               cbClsExtra ;
   int             cbWndExtra ;
   HINSTANCE     hInstance ;
   HICON         hIcon ;
   HCURSOR       hCursor ;
   HBRUSH        hbrBackground ;
   LPCTSTR       lpszMenuName ;
   LPCTSTR       lpszClassName ;
}
WNDCLASS, * PWNDCLASS ;
```

LPNP

WinMainWNDCLASS

```
WNDCLASS wndclass ;
```

10RegisterClass

WNDCLASS(lpfnWndProc)

WNDCLASS

```
wndclass.style = CS_HREDRAW | CS_VREDRAW ;
```

CWINUSER.HCS

```
#define    CS_VREDRAW              0x0001
#define    CS_HREDRAW          0x0002
#define    CS_KEYCVTWINDOW         0x0004
#define    CS_DBLCLKS          0x0008
#define    CS_OWNDC            0x0020
#define    CS_CLASSDC          0x0040
#define    CS_PARENTDC         0x0080
#define    CS_NOKEYCVT         0x0100
#define    CS_NOCLOSE          0x0200
#define    CS_SAVEBITS         0x0800
#define    CS_BYTEALIGNCLIENT      0x1000
#define    CS_BYTEALIGNWINDOW      0x2000
#define    CS_GLOBALCLASS      0x4000
```

```
#define     CS_IME              0x00010000
```

HELLOWIN(CS_HREDRAW)(CS_VREDRAW)
HELLOWIN

WNDCLASS

```
wndclass.lpfnWndProc = WndProc ;
```

WndProcHELLOWIN.CC

Windows

```
wndclass.cbClsExtra = 0 ;

wndclass.cbWndExtra = 0 ;
```

HELLOWIN0

WinMain

```
wndclass.hInstance = hInstance ;
```

```
wndclass.hIcon = LoadIcon (NULL, IDI_APPLICATION) ;
```

WindowsWindows

NULLLoadIcon.EXEhInstanceIDIID
WINUSER.HIDI_APPLICATIONLoadIconhIcon
WNDCLASSHICONhandle to an icon

```
wndclass.hCursor = LoadCursor (NULL, IDC_ARROW) ;
```

LoadCursorIDC_ARROWWNDCLASShCursor

hbrBackgroundhbrhandle
(stock)GetStockObject

```
wndclass.hbrBackground = GetStockObject (WHITE_BRUSH) ;
```

HElLOWINNULL

```
wndclass.lpszMenuName = NULL ;
```

szAppNameHelloWin

```
wndclass.lpszClassName = szAppName ;
```

ASCIIUnicodeUNICODE

10HELLOWINRegisterClassWNDCLASSRegisterClassA
WNDCLASSARegisterClassWWNDCLASSWASCIIUnicode

UNICODERegisterClassWMicrosoft
RegisterClassW0Windows

```
if (!RegisterClass (&wndclass))

{
```

```
   MessageBox (  NULL, TEXT ("This program requires Windows N
              szAppName, MB_ICONERROR) ;
   return 0 ;
}
```

MessageBoxWWindows 98Unicode

RegisterClassWNDCLASSlpfnWndProcNULLGetLastError
GetLastErrorWindowsGetLastErrorWindows
GetLastError120WINERROR.H120
ERROR_CALL_NOT_IMPLEMENTED/Platform SDK/Window
Services/Debugging and Error Handling/Error Codes/System Errors - Numerical Order

WindowsWindowsRegisterClass
RegisterClassWindows

WindowsWinMain

```
if (!hPrevInstance)
{
   wndclass.cbStyle = CS_HREDRAW | CS_VREDRAW ;
         wndclass

   RegisterClass (&wndclass) ;
}
```

16WindowsWinMainhPrevInstance
hPrevInstanceNULL

32WindowshPrevInstanceNULLhPrevInstance

CreateWindow

WindowsWindows

RegisterClassCreateWindowHELLOWIN.CCreateWindows

```
hwnd = CreateWindow (szAppName,    // window class name
   TEXT ( "The Hello Program"), // window caption
          WS_OVERLAPPEDWINDOW,     // window style
         CW_USEDEFAULT,           // initial x position
         CW_USEDEFAULT,           // initial y position
          CW_USEDEFAULT,            // initial x size
        CW_USEDEFAULT,           // initial y size
          NULL,                     // parent window handle
         NULL,                    // window menu handle
  hInstance,                 // program instance handle
  NULL) ;                    // creation parameters
```

CreateWindowACreateWindowWASCIIUnicode

window class nameszAppNameHelloWin

WS_OVERLAPPEDWINDOWCreateWindow
WINUSER.H

```
#define    WS_OVERLAPPEDWINDOW (WS_OVERLAPPED   | \
            WS_CAPTION                    | \
            WS_SYSMENU                     | \
            WS_THICKFRAME                 | \
            WS_MINIMIZEBOX                | \
            WS_MAXIMIZEBOX)
```

initial x positioninitial y positionCW_USEDEFAULTWindows
CW_USEDEFAULT0x80000000Windowsinitial
sizeinitial y sizeCW_USEDEFAULTWindows

NULLCreateWindow

NULLWinMainNULL

CreateWindowhwndHWNDWindowsWindows
hwndWindowsWindows

CreateWindowWindowsWindowsCreateWindowWindows

```
ShowWindow (hwnd, iCmdShow) ;
```

CreateWindowWinMainiCmdShowWinMain
ShowWindowSW_SHOWNORMALSW_SHOWMAXIMIZED
SW_SHOWMINNOACTIVE

ShowWindowShowWindowSW_SHOWNORMAL

```
UpdateWindow (hwnd) ;
```

HELLOWIN.CWndProcWM_PAINTWndProc

UpdateWindowWindowsWindowsWindows

```
while      (GetMessage (&msg, NULL, 0, 0))

{

   TranslateMessage (&msg) ;

   DispatchMessage (&msg) ;

}
```

msgMSGMSGWINUSER.H

```
typedef struct tagMSG
```

```
{
   HWND   hwnd ;
   UINT   message ;
   WPARAM wParam ;
   LPARAM lParam ;
   DWORD  time ;
   POINT  pt ;
}
MSG, * PMSG ;
```

POINTWINDEF.H

```
typedef struct tagPOINT
{
   LONG  x ;
   LONG  y ;
}
POINT, * PPOINT;
```

GetMessage

```
GetMessage (&msg, NULL, 0, 0)
```

WindowsmsgMSGNULL0Windows

- hwnd HELLOWINCreateWindowhwnd
- message WindowsWINUSER.H WMwindow messageHELLOWINWindows messageWM_LBUTTONDOWN0x0201
- wParam 32message parameter
- lParam 32
- time
- pt

messageWM_QUIT0x0012GetMessageWM_QUITGetMessage0

```
TranslateMessage (&msg) ;
```

msgWindows

```
DispatchMessage (&msg) ;
```

msgWindowsWindowsWindowsHELLOWINWndProe WndProcWindowsWindowsDispatchMessageDispatchMessage WindowsHELLOWINGetMessage

HELLOWINWndProcWindowsRegisterClass
CreateWindow

```
LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
```

MSGhwndCreateWindowHELLOWIN
hwnd

MSGmessage3216

WindowsSendMessageSendMessage

messageWindowsWINUSER.HWM

Windowsswitchcase0DefWindowProcWindows
DefWindowProc

HELLOWINWndProcWM_CREATEWM_PAINTWM_DESTROY

```
switch (iMsg)

{

caseWM_CREATE :

  WM_CREATE

    return 0 ;
```

```
caseWM_PAINT :
   WM_PAINT
    return 0 ;

caseWM_DESTROY :
    WM_DESTROY
    return 0 ;
}
return DefWindowProc (hwnd, iMsg, wParam, lParam) ;
```

DefWindowProc

WndProcWM_CREATEWindowsWinMainCreateWindowWndProc
HELLOWINCreateWindowWindowsWindowsWndProc
WM_CREATEWndProcWM_CREATEWindows Windo
HELLOWINWinMain

WM_CREATEHELLOWINHELLOWIN.WAVPlaySound
/Platform SDK/Graphics and Multimedia Services/Multimedia
Audio/Waveform Audio/Platform SDK/Graphics and Multimedia
Services/Multimedia Reference/Multimedia Functions

PlaySoundControl PanelSoundsPlaySou

WndProc0WM_CREATE

WM_PAINT

WndProcWM_PAINTWindows

WM_PAINTWinMainUpdateWindow

HELLOWINHELLOWINwndclassstyleCS_HREDRAW
CS_VREDRAWWindowsWM_PAINT

HELLOWINWindowsWindowsWM_PAINT

WindowsWM_PAINT

WM_PAINTBeginPaint

```
hdc = BeginPaint (hwnd, &ps) ;
```

EndPaint

```
EndPaint (hwnd, &ps) ;
```

PAINTSTRUCTPAINTSTRUCTBeginPaintEndPaint

BeginPaintWindowsWNDCLASShbrBackgroundHELLOWIN
WindowsBeginPaintBeginPaint
EndPaint

WM_PAINTDefWindowProcDefWindowProcBeginPaintEndPaint

BeginPaintWndProcGetClientRect

```
GetClientRect (hwnd, &rect) ;
```

RECTrectangleLONGlefttoprightbottomGetClientRect
lefttop0rightbottom

WndProcRECTDrawText

```
DrawText (  hdc, TEXT ("Hello, Windows 98!"), -1, &rect,

                DT_SINGLELINE | DT_CENTER | DT_VCENTER) ;
```

DrawTextBeginPaint

DrawTextWINUSER.HDrawTextGDIUser/Platform
SDK/Graphics and Multimedia Services/GDI/Fonts and Text
Hello, Windows 98!

WndProcWM_PAINTWndProcGetClientRect

WM_DESTROY

WM_DESTROYWindowsClose

HELLOWINPostQuitMessageWM_DESTROY

```
PostQuitMessage (0) ;
```

WM_QUITGetMessageWM_QUIT0GetMessageWM_QUIT0
WinMain

```
return msg.wParam ;
```

wParamPostQuitMessage0returnWinMain

Windows

HELLOWINCmainHELLOWINWinMain

HELLOWINWndProcWindowsWindows

Cfopenfopen

WindowsWindows1000WindowsWndProcRegisterClass
Windows

WindowsWndProcWindowsWndProc
WindowsWndProc

WndProcWindowsWindowsWMWINUSER.H

CsignalCtrl-CMS-DOS

WindowsDefWindowProc

HELLOWINwParamlParamDefWindowProc

WindowshwndhwndmessageWM_SIZE
WM_SIZEwParamSIZE_RESTOREDSIZE_MINIMIZED
SIZE_MAXIMIZEDSIZE_MAXSHOWSIZE_MAXHIDE WINUSER.H0
4wParam

lParam1632lParamWINDEF.HlParam

DefWindowProcHELLOWIN
CloseWM_SYSCOMMANDWndProcDefWindowProc
DefWindowProcWM_CLOSEWndProcDefWindowProcDestroyWindow
DestroyWindowWM_CLOSEDestroyWindowWindowsWM_DESTROY
WndProcPostQuitMessageWM_QUITWinMain

WindowsWindowsWindowsGetMessage

DispatchMessage

Windows

WindowsWindows
--

WM_KEYDOWNWM_KEYUPWM_CHAR
WM_MOUSEMOVEWM_LBUTTONDOWNWM_TIMER
WM_PAINTWM_QUIT

WindowsWinMainCreateWindowWindowsWM_CREATE
WinMainShowWindowWindowsWM_SIZEWM_SHOWWINDOW
WinMainUpdateWindowWindowsWM_PAINT
WM_COMMAND

Windows

WindowsDispatchMessageWindowsDispatchMessage

UpdateWindowWindowsWM_PAINT
WM_PAINTUpdateWindow

Windows--WindowsWindows

static

Windows 98Windows NTWindowsWindows16Windows

Windows
DefWindowProc

bug

Windows 98

80WindowsWindows

helloWindows

WindowsGDIWindowsWindowsWindows
Windows

WindowsWindows

WindowsWindowsWindows

WindowsWindowsWM_PAINT

WM_PAINT

WindowsWinMainUpdateWindowWindowsWM_PAINT
WM_PAINTWM_PAINT

-
- CS_HREDRAWCS_VREDRAW
- ScrollWindowScrollDC
- InvalidateRectInvalidateRgnWM_PAINT

WindowsWindowsWM_PAINT

- Windows
-
-

Windows

-
-

WM_PAINTWindowsWM_PAINTWindowsWM_PAINT

WM_PAINT

WindowsWM_PAINTWM_PAINT

WindowsWM_PAINTWindows
WindowsWM_PAINT

InvalidateRectWM_PAINTWindowsWM_PAINT
WM_PAINTGetUpdateRect

WM_PAINTBeginPaintValidateRectWM_PAINT

GDI

WindowsGDIWindowsGDI

```
TextOut (hdc, x, y, psText, iLength) ;
```

TextOutpsTextiLengthxyhdcGDIGDI

WindowsWindowsGDI

DCGDI

GDITextOutxyWindows

WindowsGDIGDIGDI

CreateDC

Windows

WM_PAINTBeginPaintEndPaintPAINTSTRUCT
WINUSER.HWindowsps

```
PAINTSTRUCT ps ;
```

WM_PAINTBeginPaintBeginPaintpsBeginPainthdc

```
HDC hdc ;
```

HDC32TextOutGDIEndPaint

WM_PAINT

```
caseWM_PAINT:
```

```
 hdc = BeginPaint (hwnd, &ps) ;

       GDI

 EndPaint (hwnd, &ps) ;

 return 0 ;
```

WM_PAINTBeginPaintEndPaintWM_PAINTWM_PAINTWindows DefWindowProcDefWindowProcWM_PAINT

```
case WM_PAINT:

   BeginPaint (hwnd, &ps) ;

   EndPaint (hwnd, &ps) ;

  return 0 ;
```

BeginPaintEndPaint

```
case WM_PAINT:

   return 0 ;   // WRONG !!!
```

WindowsWM_PAINTBeginPaintEndPaintValidateRectWindows WindowsWM_PAINT

WindowsPAINTSTRUCT

```
typedef struct tagPAINTSTRUCT
```

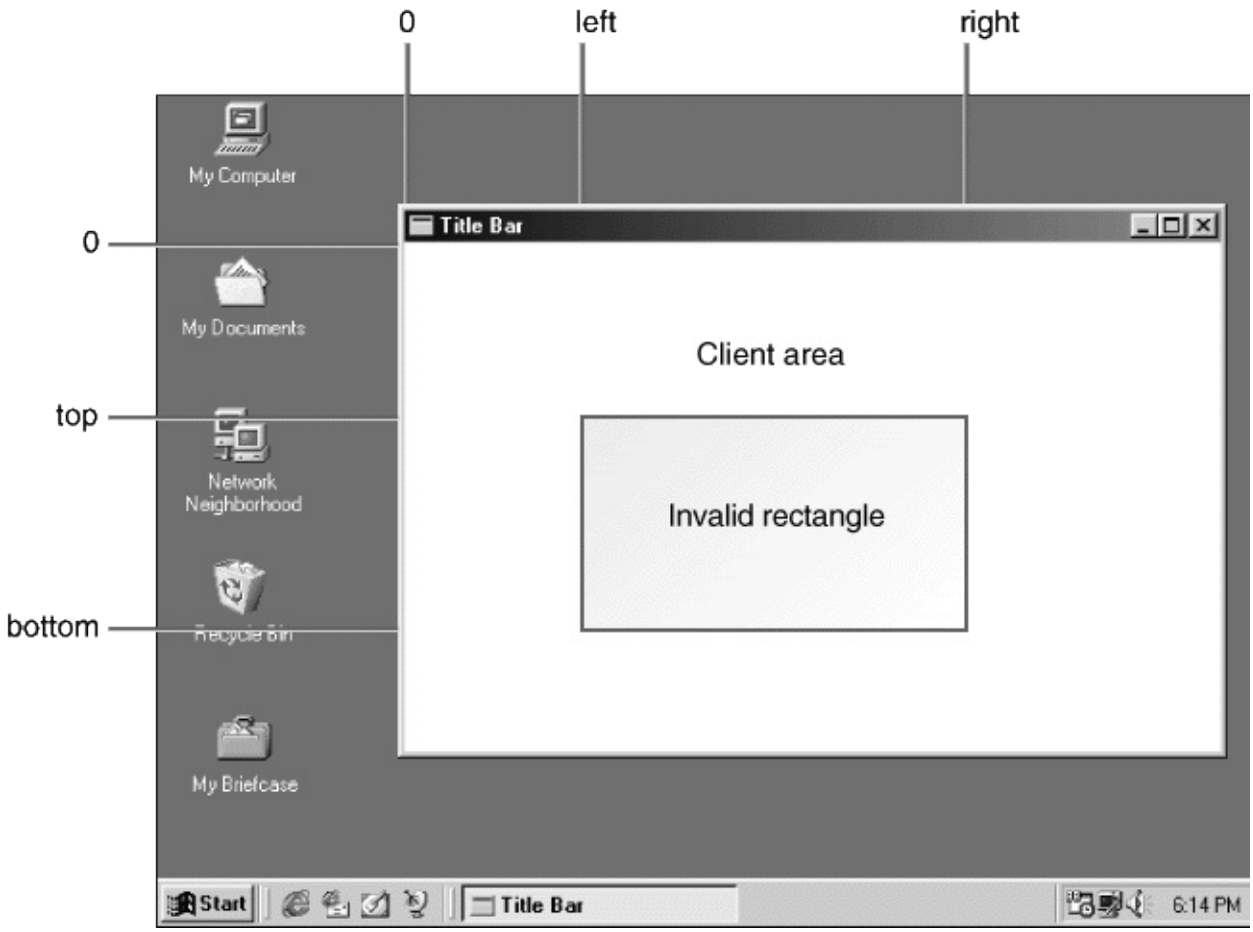
```
{
 HDC         hdc ;
  BOOL     fErase ;
  RECT     rcPaint ;
  BOOL        fRestore ;
  BOOL        fIncUpdate ;
  BYTE        rgbReserved[32] ;
} PAINTSTRUCT ;
```

BeginPaintWindowsWindowshdcWindowsBeginPaint
fEraseFALSE(0)WindowsBeginPaint
WM_ERASEBKGNDWindowsWNDCLASShbrBackground
WNDCLASSWinMainWindows

```
wndclass.hbrBackground = (HBRUSH) GetStockObject (WHITE_B
```

WindowsInvalidateRectFALSE0WindowsBeginPaint
PAINTSTRUCTfEraseTRUE

PAINTSTRUCTrcPaintRECTRECTlefttoprightbottom
PAINTSTRUCTrcPaint4-1



4-1

PAINTSTRUCTrcPaintWindowsWindows

WM_PAINT

```
InvalidateRect (hwnd, NULL, TRUE) ;
```

BeginPaintFALSE

WindowsWM_PAINTrcPaintBeginPaintWindows
rcPaint

HELLOWINWM_PAINTDrawTextDrawTextWindows
WM_PAINTGDI

WM_PAINTWM_PAINT

GetDCReleaseDC

```
hdc = GetDC (hwnd) ;

GDI

ReleaseDC (hwnd, hdc) ;
```

BeginPaintEndPaintGetDCReleaseDCGetDCReleaseDCGetDC
ReleaseDC

BeginPaintGetDCBeginPaintGetDC

```
ValidateRect (hwnd, NULL) ;
```

GetDCReleaseDCWM_PAINTWM_PAINT

GetDCGetWindowDCGetDCGetWindowDCGetWindowDC
WM_NCPAINT

TextOut

TextOutGDI

```
TextOut (hdc, x, y, psText, iLength) ;
```

GetDCWM_PAINTBeginPaint

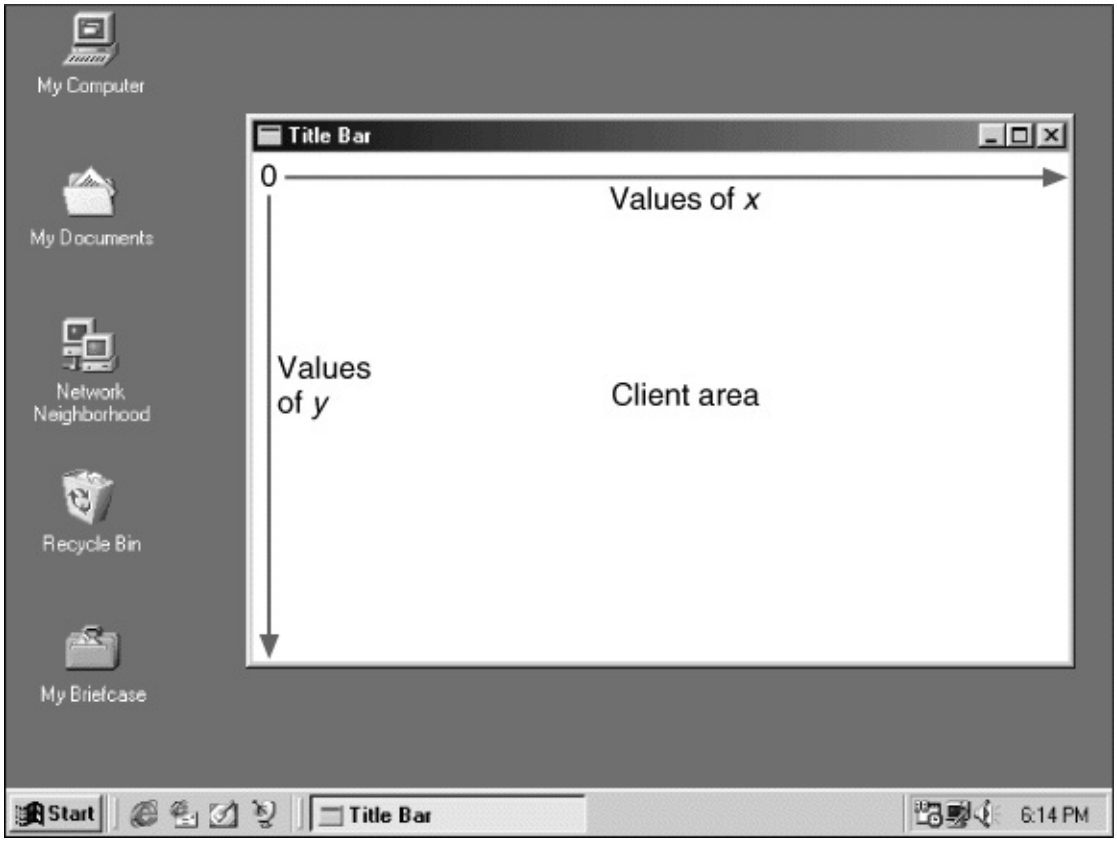
Windows

WindowsWindowsWHITE_BRUSHWindows

psTextiLengthpsTextUnicodeiLengthASCII
WindowsText0utUnicode0nLength

TextOutxyxy(x,y)xy0TextOutxy0

GDITextOut
MM_TEXTWINGDI.HMM_TEXTxy4-2
MM_TEXTWindowsPAINTSTRUCT



4-2　MM_TEXTxy

GetDCBeginPaintWindowsWindows

TextOutWindowsWindowsSYSTEM_FONTWindows

Windows(fixed-pitch)Windows
WiWindows

TrueType2580

TextOut

Windows640×480800×6001024×768Windows

GetSystemMetricsGetTextMetricsGetTextMetricsWindows
WINGDI.HTEXTMETRICTEXTMETRIC20

```
typedef struct tagTEXTMETRIC
{
    LONG tmHeight ;
    LONG tmAscent ;
    LONG tmDescent ;
    LONG tmInternalLeading ;
    LONG tmExternalLeading ;
    LONG tmAveCharWidth ;
```

```
  LONG tmMaxCharWidth ;


}
TEXTMETRIC, * PTEXTMETRIC ;
```

MM_TEXT

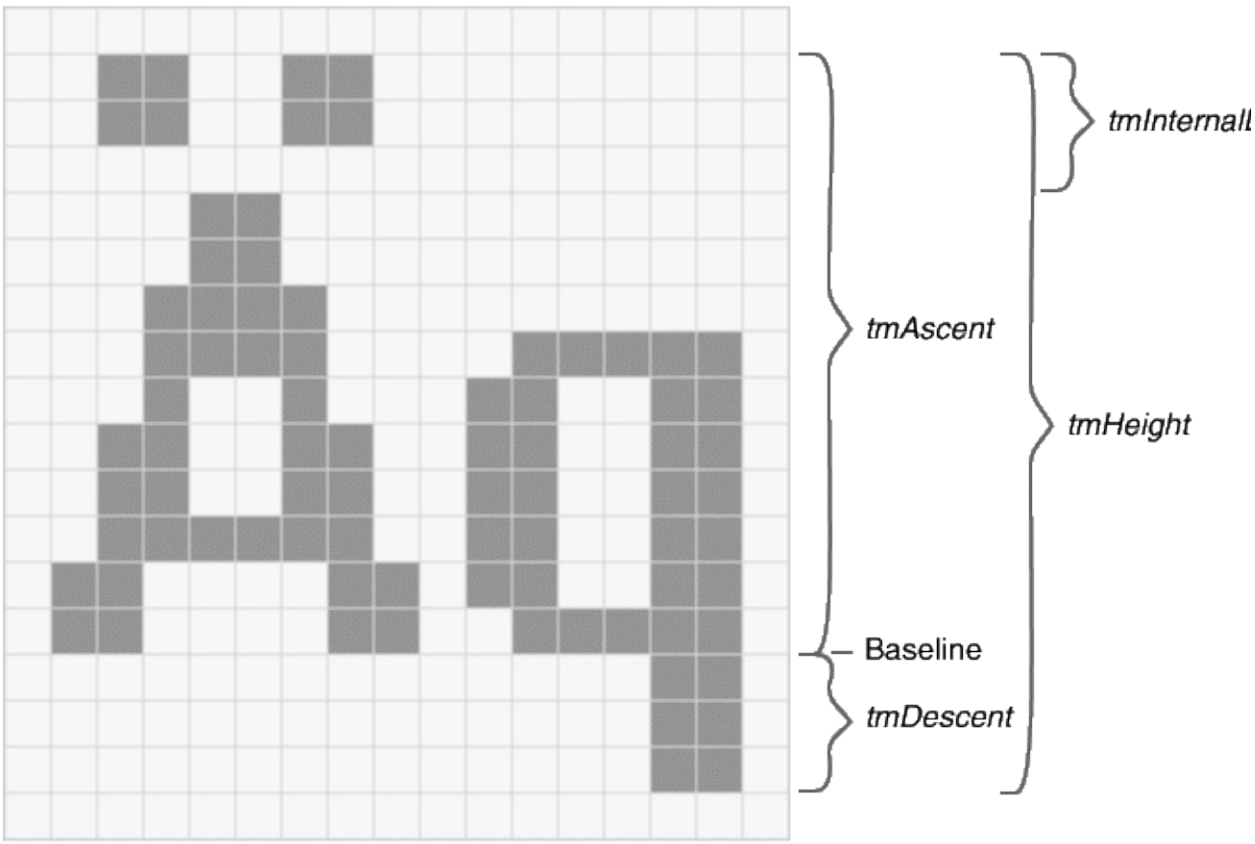
GetTextMetricstm

```
TEXTMETRIC tm ;
```

GetTextMetrics

```
hdc = GetDC (hwnd) ;
GetTextMetrics (hdc, &tm) ;
ReleaseDC (hwnd, hdc) ;
```

TEXTMETRIC544-3



4-3　4

tmHeighttmAscenttmDescentleadingTEXTMETRIC
tmAscenttmHeighttmInternalLeading0

TEXTMETRICtmHeighttmExternalLeading
tmExternalLeading04-34-3Windows640×480

TEXTMETRICStmAveCharWidthtmMaxCharWidth
4-3714

tmAveCharWidth150

WindowsWindowsWindowsGetTextMetrics

WindowsGetTexMetricsWM_CREATEWM_CREATE
WinMainCreateWindowWindowsWM_CREATE

Windows(cxChar)(cyChar)

```
static int cxChar, cyChar ;
```

ccountxystaticWM_PAINTstatic

WM_CREATE

```
case WM_CREATE:
    hdc = GetDC (hwnd) ;
    GetTextMetrics (hdc, &tm) ;
    cxChar = tm.tmAveCharWidth ;
    cyChar = tm.tmHeight + tm.tmExternalLeading ;

    ReleaseDC (hwnd, hdc) ;
    return 0 ;
```

cyChartmExternalLeading0cyChar

printfsprintfWindowssprintfwsprintfprintf
TextOutsprintfwsprintfTextOutiLengthwsprintfTextOut

```
int  iLength ;
TCHAR szBuffer [40] ;
```

```
iLength = wsprintf (szBuffer, TEXT ("The sum of %i and %i is %i"
                    iA, iB, iA + iB) ;
TextOut (hdc, x, y, szBuffer, iLength) ;
```

nLengthTextOutiLength

```
TextOut (hdc, x, y, szBuffer,
  wsprintf (szBuffer, TEXT ("The sum of %i and %i is %i"),
                  iA, iB, iA + iB)) ;
```

WM_PAINTTextOut

WindowsGetSystemMetricsWindows
GetSystemMetrics

Windows75WindowsWindows
GetSystemMetrics

GetSystemMetricsGetSystemMetricsWindows
SYSMETS.H4-1

4-1　SYSMETS.H

```
/*-------------------------------------------------------
```

```
SYSMETS.H -- System metrics display structure

------------------------------------------------------------*/

#define NUMLINES ((int) (sizeof sysmetrics / sizeof sysmetrics [0

struct

{

   int    Index ;

   TCHAR *       szLabel ;

  TCHAR *       szDesc ;

}

sysmetrics [] =

{

 SM_CXSCREEN,    TEXT ("SM_CXSCREEN"),

                 TEXT ("Screen width in pixels"),

   SM_CYSCREEN, TEXT ("SM_CYSCREEN"),

                      TEXT ("Screen height in pixels"),

   SM_CXVSCROLL, TEXT ("SM_CXVSCROLL"),

                       TEXT ("Vertical scroll width"),

   SM_CYHSCROLL,  TEXT ("SM_CYHSCROLL"),

                       TEXT ("Horizontal scroll height"),
```

```
SM_CYCAPTION, TEXT ("SM_CYCAPTION"),
                    TEXT ("Caption bar height"),
SM_CXBORDER, TEXT ("SM_CXBORDER"),
                  TEXT ("Window border width"),
SM_CYBORDER,  TEXT ("SM_CYBORDER"),
                  TEXT ("Window border height"),
SM_CXFIXEDFRAME,TEXT ("SM_CXFIXEDFRAME"),
                  TEXT ("Dialog window frame width"),
SM_CYFIXEDFRAME,TEXT ("SM_CYFIXEDFRAME"),
                  TEXT ("Dialog window frame height"),
SM_CYVTHUMB,  TEXT ("SM_CYVTHUMB"),
                 TEXT ("Vertical scroll thumb height"),
SM_CXHTHUMB,    TEXT ("SM_CXHTHUMB"),
                    TEXT ("Horizontal scroll thumb width"),
SM_CXICON,      TEXT ("SM_CXICON"),
                  TEXT ("Icon width"),
SM_CYICON,      TEXT ("SM_CYICON"),
                TEXT ("Icon height"),
```

```
SM_CXCURSOR,  TEXT ("SM_CXCURSOR"),
                    TEXT ("Cursor width"),
SM_CYCURSOR,    TEXT ("SM_CYCURSOR"),
                    TEXT ("Cursor height"),
SM_CYMENU,      TEXT ("SM_CYMENU"),
                    TEXT ("Menu bar height"),
SM_CXFULLSCREEN,TEXT ("SM_CXFULLSCREEN"),
                    TEXT ("Full screen client area width"),
SM_CYFULLSCREEN,TEXT ("SM_CYFULLSCREEN"),
                    TEXT ("Full screen client area height"),
SM_CYKANJIWINDOW,TEXT ("SM_CYKANJIWINDOW"),
                    TEXT ("Kanji window height"),
SM_MOUSEPRESENT, TEXT ("SM_MOUSEPRESENT"),
                    TEXT ("Mouse present flag"),
SM_CYVSCROLL,TEXT ("SM_CYVSCROLL"),
                    TEXT ("Vertical scroll arrow height"),
SM_CXHSCROLL,TEXT ("SM_CXHSCROLL"),
                    TEXT ("Horizontal scroll arrow width"),
SM_DEBUG,       TEXT ("SM_DEBUG"),
```

```
                TEXT ("Debug version flag"),
SM_SWAPBUTTON,TEXT ("SM_SWAPBUTTON"),
                TEXT ("Mouse buttons swapped flag"),
SM_CXMIN,       TEXT ("SM_CXMIN"),
                TEXT ("Minimum window width"),
SM_CYMIN,       TEXT ("SM_CYMIN"),
                TEXT ("Minimum window height"),
SM_CXSIZE,      TEXT ("SM_CXSIZE"),
                TEXT ("Min/Max/Close button width"),
SM_CYSIZE,      TEXT ("SM_CYSIZE"),
                TEXT ("Min/Max/Close button height"),
SM_CXSIZEFRAME,TEXT ("SM_CXSIZEFRAME"),
                TEXT ("Window sizing frame width"),
SM_CYSIZEFRAME,TEXT ("SM_CYSIZEFRAME"),
                TEXT ("Window sizing frame height"),
SM_CXMINTRACK,TEXT ("SM_CXMINTRACK"),
                TEXT ("Minimum window tracking width"),
SM_CYMINTRACK,TEXT ("SM_CYMINTRACK"),
```

```
                    TEXT ("Minimum window tracking height"),
SM_CXDOUBLECLK,TEXT ("SM_CXDOUBLECLK"),
                    TEXT ("Double click x tolerance"),
SM_CYDOUBLECLK,TEXT ("SM_CYDOUBLECLK"),
                    TEXT ("Double click y tolerance"),
SM_CXICONSPACING,TEXT ("SM_CXICONSPACING"),
                    TEXT ("Horizontal icon spacing"),
SM_CYICONSPACING,TEXT ("SM_CYICONSPACING"),
                    TEXT ("Vertical icon spacing"),
SM_MENUDROPALIGNMENT,TEXT ("SM_MENUDROPALIGNMENT
                        TEXT ("Left or right menu drop"),
SM_PENWINDOWS,      TEXT ("SM_PENWINDOWS"),
                        TEXT ("Pen extensions installed"),
SM_DBCSENABLED,     TEXT ("SM_DBCSENABLED"),
                        TEXT ("Double-Byte Char Set enabled"),
SM_CMOUSEBUTTONS,       TEXT ("SM_CMOUSEBUTTONS"),
                       TEXT ("Number of mouse buttons"),
SM_SECURE,              TEXT ("SM_SECURE"),
                        TEXT ("Security present flag"),
```

```
SM_CXEDGE,              TEXT ("SM_CXEDGE"),
                        TEXT ("3-D border width"),
SM_CYEDGE,              TEXT ("SM_CYEDGE"),
                        TEXT ("3-D border height"),
SM_CXMINSPACING,        TEXT ("SM_CXMINSPACING"),
                        TEXT ("Minimized window spacing width"),
SM_CYMINSPACING,        TEXT ("SM_CYMINSPACING"),
                        TEXT ("Minimized window spacing height")
SM_CXSMICON,        TEXT ("SM_CXSMICON"),
                        TEXT ("Small icon width"),
SM_CYSMICON,       TEXT ("SM_CYSMICON"),
                        TEXT ("Small icon height"),
SM_CYSMCAPTION,      TEXT ("SM_CYSMCAPTION"),
                        TEXT ("Small caption height"),
SM_CXSMSIZE,        TEXT ("SM_CXSMSIZE"),
                       TEXT ("Small caption button width"),
SM_CYSMSIZE,        TEXT ("SM_CYSMSIZE"),
                        TEXT ("Small caption button height"),
```

```
SM_CXMENUSIZE,      TEXT ("SM_CXMENUSIZE"),
                    TEXT ("Menu bar button width"),
SM_CYMENUSIZE,      TEXT ("SM_CYMENUSIZE"),
                    TEXT ("Menu bar button height"),
SM_ARRANGE,         TEXT ("SM_ARRANGE"),
                    TEXT ("How minimized windows arranged"
SM_CXMINIMIZED,     TEXT ("SM_CXMINIMIZED"),
                    TEXT ("Minimized window width"),
SM_CYMINIMIZED,     TEXT ("SM_CYMINIMIZED"),
                    TEXT ("Minimized window height"),
SM_CXMAXTRACK,      TEXT ("SM_CXMAXTRACK"),
                    TEXT ("Maximum draggable width"),
SM_CYMAXTRACK,      TEXT ("SM_CYMAXTRACK"),
                    TEXT ("Maximum draggable height"),
SM_CXMAXIMIZED,     TEXT ("SM_CXMAXIMIZED"),
                    TEXT ("Width of maximized window"),
SM_CYMAXIMIZED,     TEXT ("SM_CYMAXIMIZED"),
                    TEXT ("Height of maximized window"),
SM_NETWORK,         TEXT ("SM_NETWORK"),
```

```
                    TEXT ("Network present flag"),
SM_CLEANBOOT,       TEXT ("SM_CLEANBOOT"),
                    TEXT ("How system was booted"),
SM_CXDRAG,          TEXT ("SM_CXDRAG"),
                    TEXT ("Avoid drag x tolerance"),
SM_CYDRAG,          TEXT ("SM_CYDRAG"),
                    TEXT ("Avoid drag y tolerance"),
SM_SHOWSOUNDS,      TEXT ("SM_SHOWSOUNDS"),
                    TEXT ("Present sounds visually"),
SM_CXMENUCHECK,     TEXT ("SM_CXMENUCHECK"),
                   TEXT ("Menu check-mark width"),
SM_CYMENUCHECK,     TEXT ("SM_CYMENUCHECK"),
                   TEXT ("Menu check-mark height"),
SM_SLOWMACHINE,     TEXT ("SM_SLOWMACHINE"),
                    TEXT ("Slow processor flag"),
SM_MIDEASTENABLED,      TEXT ("SM_MIDEASTENABLED"),
                    TEXT ("Hebrew and Arabic enabled flag"),
SM_MOUSEWHEELPRESENT,TEXT ("SM_MOUSEWHEELPRESEN
```

```
                        TEXT ("Mouse wheel present flag"),
    SM_XVIRTUALSCREEN,      TEXT ("SM_XVIRTUALSCREEN"),
                        TEXT ("Virtual screen x origin"),
    SM_YVIRTUALSCREEN,      TEXT ("SM_YVIRTUALSCREEN"),
                        TEXT ("Virtual screen y origin"),
    SM_CXVIRTUALSCREEN,     TEXT ("SM_CXVIRTUALSCREEN"),
                        TEXT ("Virtual screen width"),
    SM_CYVIRTUALSCREEN,     TEXT ("SM_CYVIRTUALSCREEN"),
                        TEXT ("Virtual screen height"),
    SM_CMONITORS,       TEXT ("SM_CMONITORS"),
                       TEXT ("Number of monitors"),
    SM_SAMEDISPLAYFORMAT,TEXT ("SM_SAMEDISPLAYFORMAT")
                        TEXT ("Same color format flag")
} ;
```

SYSMETS1SYSMETS1.C4-2WinMainHELLOWINWndProc

4-2　SYSMETS1.C

```
/*------------------------------------------------------------
SYSMETS1.C -- System Metrics Display Program No. 1
```

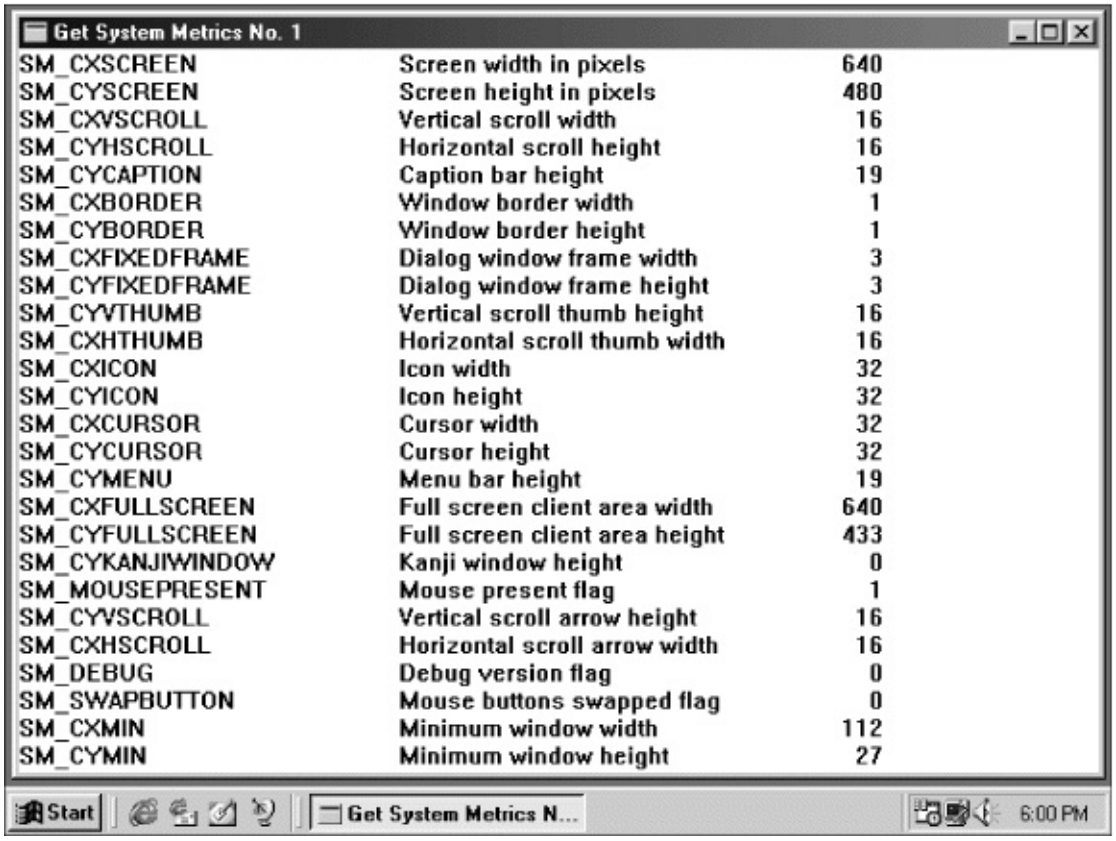
```
                (c) Charles Petzold, 1998
----------------------------------------------------------------*/
#include <windows.h>
#include "sysmets.h"
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                PSTR szCmdLine, int iCmdShow)
{
   static TCHAR szAppName[] = TEXT ("SysMets1") ;
   HWND   hwnd ;
   MSG   msg ;
   WNDCLASS      wndclass ;
   wndclass.style       = CS_HREDRAW | CS_VREDRAW ;
  wndclass.lpfnWndProc  = WndProc ;
  wndclass.cbClsExtra   = 0 ;
  wndclass.cbWndExtra   = 0 ;
  wndclass.hInstance    = hInstance ;
  wndclass.hIcon        = LoadIcon (NULL, IDI_APPLICATION) ;
```

```
wndclass.hCursor      = LoadCursor (NULL, IDC_ARROW) ;
wndclass.hbrBackground        = (HBRUSH) GetStockObject (W
wndclass.lpszMenuName         = NULL ;
wndclass.lpszClassName        = szAppName ;


if (!RegisterClass (&wndclass))
{
          MessageBox (  NULL, TEXT ("This program requires W
                szAppName, MB_ICONERROR) ;


return 0 ;
}
hwnd = CreateWindow (szAppName, TEXT ("Get System Metr
             WS_OVERLAPPEDWINDOW,
             CW_USEDEFAULT, CW_USEDEFAULT,
             CW_USEDEFAULT, CW_USEDEFAULT,
             NULL, NULL, hInstance, NULL) ;
ShowWindow (hwnd, iCmdShow) ;
UpdateWindow (hwnd) ;
```

```
   while (GetMessage (&msg, NULL, 0, 0))
              {
     TranslateMessage (&msg) ;
     DispatchMessage (&msg) ;
           }
   return msg.wParam ;
    }

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
   static int    cxChar, cxCaps, cyChar ;
   HDC                 hdc ;
   int                 i ;
   PAINTSTRUCT ps ;
   TCHAR               szBuffer [10] ;
   TEXTMETRIC  tm ;

   switch (message)
   {
```

```
case   WM_CREATE:
     hdc = GetDC (hwnd) ;

      GetTextMetrics (hdc, &tm) ;
      cxChar = tm.tmAveCharWidth ;
      cxCaps = (tm.tmPitchAndFamily & 1 ? 3 : 2) * cxChar / 2 ;
      cyChar = tm.tmHeight + tm.tmExternalLeading ;

      ReleaseDC (hwnd, hdc) ;
      return 0 ;

 case   WM_PAINT :
      hdc = BeginPaint (hwnd, &ps) ;

      for (i = 0 ; i < NUMLINES ; i++)
            {
       TextOut      (hdc, 0, cyChar * i,
                  sysmetrics[i].szLabel,
                  lstrlen (sysmetrics[i].szLabel)) ;

       TextOut      (hdc, 22 * cxCaps, cyChar * i,
```

```
                    sysmetrics[i].szDesc,
                    lstrlen (sysmetrics[i].szDesc)) ;

                SetTextAlign (hdc, TA_RIGHT | TA_TOP) ;
      TextOut (hdc, 22 * cxCaps + 40 * cxChar, cyChar * i,   szBuf
   wsprintf (szBuffer, TEXT ("%5d"),
   GetSystemMetrics (sysmetrics[i].iIndex))) ;
          SetTextAlign (hdc, TA_LEFT | TA_TOP) ;
      }
        EndPaint (hwnd, &ps) ;
        return 0 ;
   case   WM_DESTROY :
        PostQuitMessage (0) ;
   return 0 ;
   }
   return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```



4-4 SYSMETS1

SYSMETS1.C

SYSMETS1.CWndProcWM_CREATEWM_PAINTWM_DESTROY
WM_DESTROY HELLOWIN

WM_CREATECreateWindowWindowsWM_CREATE
SYSMETS1GetDCGetTextMetricsSYSMETS1cxChar
cyChar

SYSMETS1cxCaps cxCa
TEXTMETRICtmPitchAndFamily10

```
cxCaps = (tm.tmPitchAndFamily & 1 ? 3 : 2) * cxChar / 2 ;
```

SYSMETS1WM_PAINTBeginPaintforSYSMETS.Hsysmetrics
TextOutTextOut

```
cyChar * i
```

TextOutTextOut0sysmetricsszLabelWindowslstrlenTextOut

TextOutsysmetricsszDescTextOut

```
22 * cxCaps
```

2020 × cxC

TextOutGetSystemMetrics09

SetTextAlignSYSMETS1

```
SetTextAlign (hdc, TA_RIGHT | TA_TOP) ;
```

TextOut

TextOut

```
22 * cxCaps + 40 * cxChar
```

40*cxCharTextOutSetTextAlign

SYSMETS1

SYSMETS1Windows

WindowsSM_CXFULLSCREENSM_CYFULLSCREEN
GetSystemMetrics

GetClientRectWM_SIZEWindowsWM_SIZElParam

```
static int cxClient, cyClient ;
```

cxCharcyCharWM_SIZE

```
caseWM_SIZE:
  cxClient = LOWORD (lParam) ;
  cyClient = HIWORD (lParam) ;
  return 0 ;
```

WindowsLOWORDHIWORDWindowsWINDEF.H

```
#define LOWORD(l) ((WORD)(l))
#define HIWORD(l) ((WORD)(((DWORD)(l) >> 16) & 0xFFFF))
```

WORD1600xFFFF32

WindowsWM_SIZEWM_PAINT

```
CS_HREDRAW | CS_VREDRAW
```

Windows

cyClient / cyChar

0

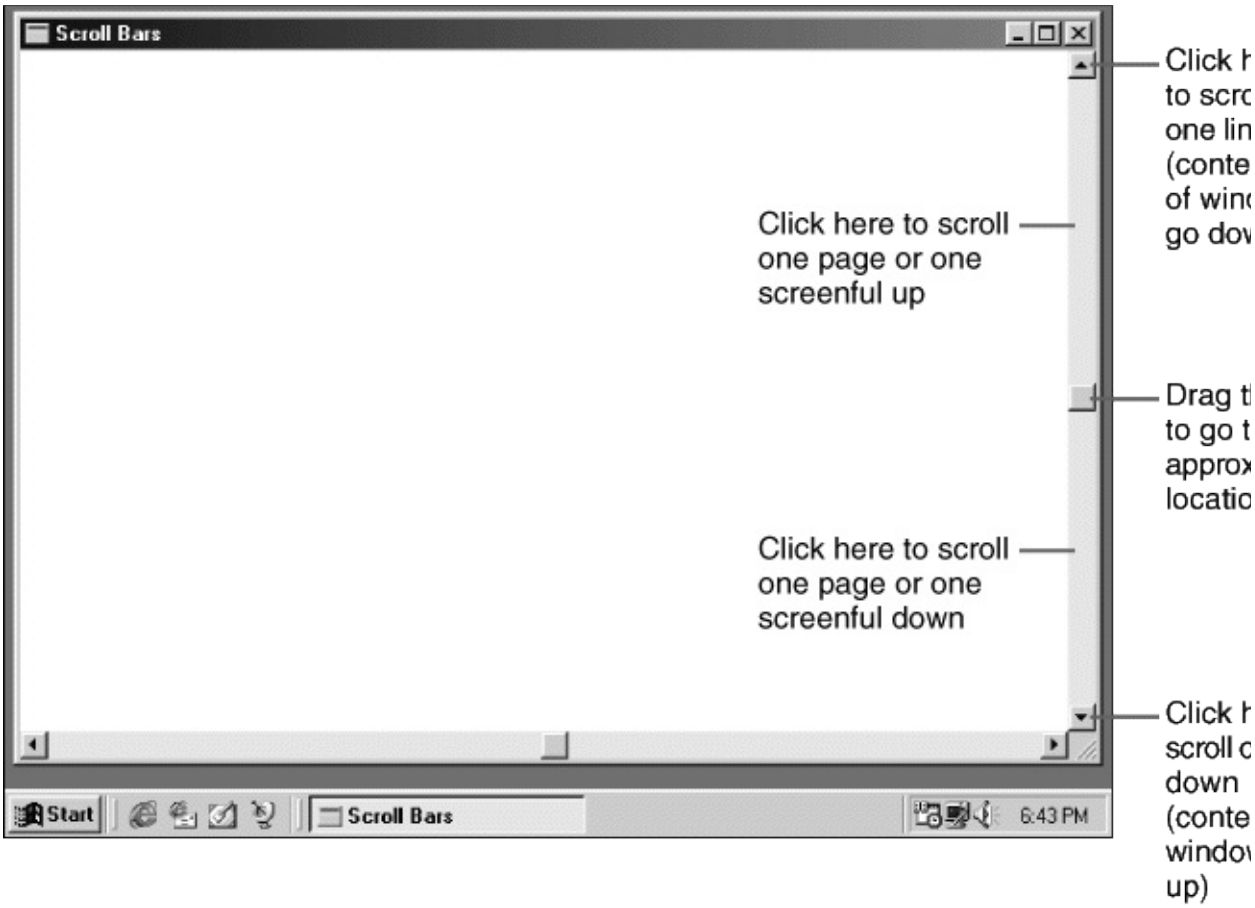
cxClient / cxChar

WM_CREATEcxCharcyChar0WinMainCreateWindowWM_CREATE
WinMainShowWindowWM_CREATEcxCharcyChar

Windows

--

4-5



4-5

Windows

CreateWindowWSWS_VSCROLL/WS_HSCROLL
GetSystemMetrics

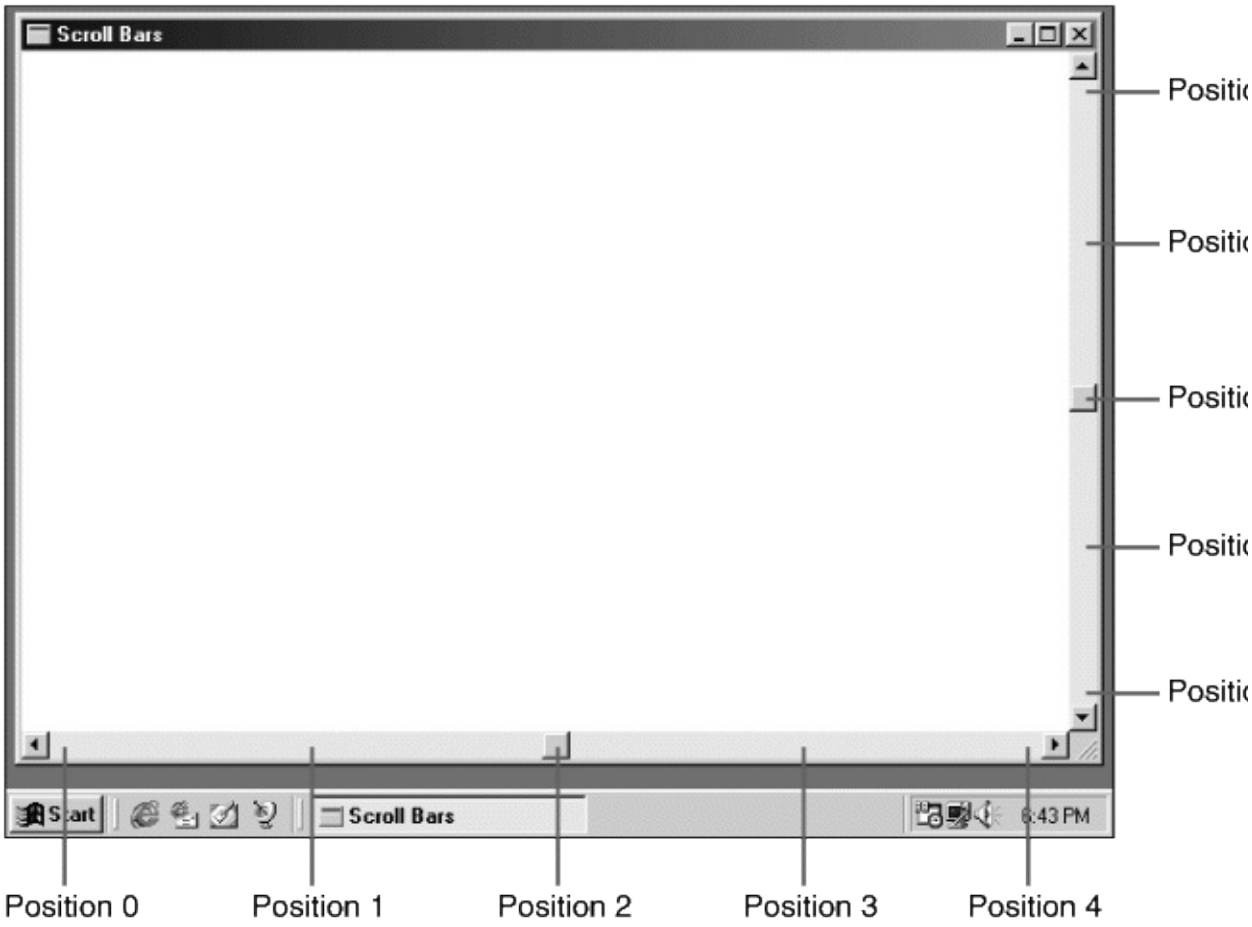
WindowsSYSMETS

0100

```
SetScrollRange (hwnd, iBar, iMin, iMax, bRedraw) ;
```

iBarSB_VERTSB_HORZiMiniMaxWindowsbRedrawTRUE
SetScrollRangebRedrawFALSE

0454-6



SetScrollPos

```
SetScrollPos (hwnd, iBar, iPos, bRedraw) ;
```

iPosiMiniMaxWindowsGetScrollRangeGetScrollPos

WindowsWindows

-
-
-
-
-
-
-
-

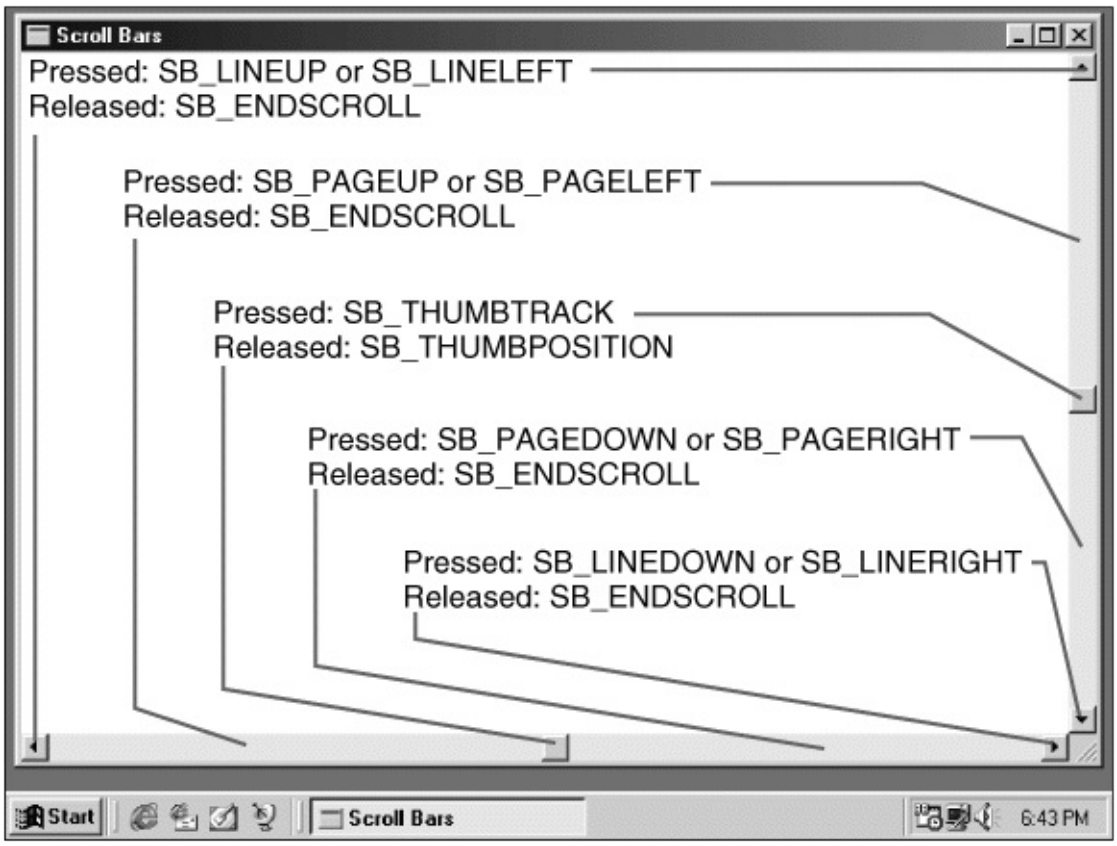
WindowsWM_VSCROLLWM_HSCROLL

WM_VSCROLLWM_HSCROLLwParamlParamlParam

wParamwParamSBscroll
WINUSER.H

```
#define SB_LINEUP      0
#define SB_LINELEFT          0
#define SB_LINEDOWN          1
#define SB_LINERIGHT         1
#define SB_PAGEUP        2
#define SB_PAGELEFT          2
#define SB_PAGEDOWN          3
#define SB_PAGERIGHT         3
#define SB_THUMBPOSITION   4
#define SB_THUMBTRACK        5
#define SB_TOP               6
#define SB_LEFT          6
#define SB_BOTTOM        7
#define SB_RIGHT         7
#define SB_ENDSCROLL         8
```

LEFTRIGHTUPDOWNTOPBOTTOM4-7



4-7 wParam

SB_ENDSCROLLWindowsSetScrollPos

SB_THUMBTRACKSB_THUMBPOSITIONwParam
SB_THUMBTRACKwParamwParamSB_THUMBPOSITION
wParamwParam

WindowsSB_THUMBTRACKSetScrollPos
SB_THUMBTRACKSB_THUMBPOSITION

SB_THUMBTRACKSB_THUMBPOSITIONSB_THUMBTRACK
SB_THUMBPOSITIONSB_THUMBTRACK

WINUSER.HSB_TOPSB_BOTTOMSB_LEFTSB_RIGHT

32wParam16SB_THUMBTRACKSB_THUMBPOSITION
GetScrollInfo

SYSMETS

SYSMET24-3

4-3　SYSMETS2.C

```
/*------------------------------------------------------------
   SYSMETS2.C -- System Metrics Display Program No. 2
         (c) Charles Petzold, 1998
------------------------------------------------------------*/
#include <windows.h>
#include "sysmets.h"
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
               PSTR szCmdLine, int iCmdShow)
{
   static TCHAR szAppName[] = TEXT ("SysMets2") ;
   HWND   hwnd ;
   MSG    msg ;
   WNDCLASS wndclass ;
```

```
wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
wndclass.lpfnWndProc  = WndProc ;
wndclass.cbClsExtra   = 0 ;
wndclass.cbWndExtra   = 0 ;
wndclass.hInstance      = hInstance ;
wndclass.hIcon        = LoadIcon (NULL, IDI_APPLICATION) ;
wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
wndclass.hbrBackground        = (HBRUSH) GetStockObject (W
wndclass.lpszMenuName         = NULL ;
wndclass.lpszClassName       = szAppName ;

if (!RegisterClass (&wndclass))
{
MessageBox (NULL, TEXT ("This program requires Windows NT
      szAppName, MB_ICONERROR) ;
return 0 ;
}

hwnd = CreateWindow (szAppName, TEXT ("Get System Metr
```

```
                WS_OVERLAPPEDWINDOW | WS_VSCROLL,
                CW_USEDEFAULT, CW_USEDEFAULT,
                CW_USEDEFAULT, CW_USEDEFAULT,
                NULL, NULL, hInstance, NULL) ;
   ShowWindow (hwnd, iCmdShow) ;
   UpdateWindow (hwnd) ;

   while (GetMessage (&msg, NULL, 0, 0))
   {
      TranslateMessage (&msg) ;
       DispatchMessage (&msg) ;
   }
   return msg.wParam ;
}
LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
   static int  cxChar, cxCaps, cyChar, cyClient, iVscrollPos ;
   HDC         hdc ;
   int         i, y ;
```

```
    PAINTSTRUCT ps ;
    TCHAR       szBuffer[10] ;
    TEXTMETRIC  tm ;
    switch (message)
    {
case WM_CREATE:
    hdc = GetDC (hwnd) ;
    GetTextMetrics (hdc, &tm) ;
    cxChar = tm.tmAveCharWidth ;
    cxCaps = (tm.tmPitchAndFamily & 1 ? 3 : 2) * cxChar / 2 ;
    cyChar = tm.tmHeight + tm.tmExternalLeading ;

    ReleaseDC (hwnd, hdc) ;
    SetScrollRange (hwnd, SB_VERT, 0, NUMLINES - 1, FALSE) ;
    SetScrollPos   (hwnd, SB_VERT, iVscrollPos, TRUE) ;
          return 0 ;

    case WM_SIZE:
          cyClient = HIWORD (lParam) ;
          return 0 ;
```

```
case WM_VSCROLL:
        switch (LOWORD (wParam))
     {
case SB_LINEUP:
      iVscrollPos -= 1 ;
        break ;

case SB_LINEDOWN:
        iVscrollPos += 1 ;
        break ;

case SB_PAGEUP:
        iVscrollPos -= cyClient / cyChar ;
        break ;

case SB_PAGEDOWN:
        iVscrollPos += cyClient / cyChar ;
        break ;
```

```
case SB_THUMBPOSITION:
        iVscrollPos = HIWORD (wParam) ;
        break ;

default :
        break ;
    }
iVscrollPos = max (0, min (iVscrollPos, NUMLINES - 1)) ;
if (iVscrollPos != GetScrollPos (hwnd, SB_VERT))
    {
        SetScrollPos (hwnd, SB_VERT, iVscrollPos, TRUE) ;
        InvalidateRect (hwnd, NULL, TRUE) ;
    }
        return 0 ;
case WM_PAINT:
        hdc = BeginPaint (hwnd, &ps) ;
        for (i = 0 ; i < NUMLINES ; i++)
```

```
        {
            y = cyChar * (i - iVscrollPos) ;
            TextOut (hdc, 0, y,
                sysmetrics[i].szLabel,
                lstrlen (sysmetrics[i].szLabel)) ;

            TextOut (hdc, 22 * cxCaps, y,
                sysmetrics[i].szDesc,
                lstrlen (sysmetrics[i].szDesc)) ;

            SetTextAlign (hdc, TA_RIGHT | TA_TOP) ;
            TextOut (hdc, 22 * cxCaps + 40 * cxChar, y, szBuffer
                wsprintf (szBuffer, TEXT ("%5d"),
                          GetSystemMetrics (sysmetrics[i].iInde
            SetTextAlign (hdc, TA_LEFT | TA_TOP) ;
    }
        EndPaint (hwnd, &ps) ;
        return 0 ;
```

```
    case WM_DESTROY:
          PostQuitMessage (0) ;
          return 0 ;
    }
    return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

CreateWindowWS_VSCROLL

```
WS_OVERLAPPEDWINDOW | WS_VSCROLL
```

WndProcWM_CREATE

```
SetScrollRange (hwnd, SB_VERT, 0, NUMLINES - 1, FALSE) ;
SetScrollPos (hwnd, SB_VERT, iVscrollPos, TRUE) ;
```

sysmetricsNUMLINES0NUMLINES-100NUMLINES-1

WM_VSCROLLiVscrollPosSB_LINEUPSB_LINEDOWN
SB_PAGEUPSB_PAGEDOWNcyClient /cyCl
SB_THUMBPOSITIONwParamSB_ENDSCROLLSB_THUMBTRACK

WM_VSCROLLiVscrollPosminmaxiVscrollPosiVscrollPos
GetScrollPosSetScrollPosInvalidateRect

InvalidateRectWM_PAINTSYSMETS1WM_PAINTy

```
cyChar * i
```

SYSMETS2

```
cyChar * (i - iVscrollPos)
```

NUMLINESiVscrollPosWindows

WM_VSCROLL

SYSMETS2InvalidateRectWindowsWM_PAINT

WindowsWM_PAINTWM_PAINT

WindowsWM_PAINT

SYSMETS2WM_PAINTInvalidateRect

WM_PAINTInvalidateRectWindowsWM_PAINT
WindowsWM_PAINT

InvalidateRectUpdateWindow

```
UpdateWindow (hwnd) ;
```

UpdateWindowWindowsWM_PAINTWM_PAINT
WindowsWindowsUpdateWindow

UpdateWindowWinMainWM_PAINTUpdateWindow

SYSMETS2Win32

/Platform SDK/User Interface Services/Controls/Scroll Bars
SetScrollRangeSetScrollPosGetScrollRangeGetScrollPos
Windows 1.0Win32 API32Windows

Win32 APISetScrollInfoGetScrollInfo

SYSMETS2Windows

$$\frac{\text{捲動方塊大小}}{\text{滾動長度}} \approx \frac{\text{頁面大小}}{\text{範圍}} \approx \frac{\text{顯示的文件數量}}{\text{文件的總大小}}$$

SetScrollInfoSYSMETS3

GetScrollInfoAPI65,53616WindowsWin3232
SB_THUMBTRACKSB_THUMBPOSITIONWM_VSCROLL
WM_HSCROLL16GetScrollInfo32

SetScrollInfoGetScrollInfo

```
SetScrollInfo (hwnd, iBar, &si, bRedraw) ;

GetScrollInfo (hwnd, iBar, &si) ;
```

iBarSB_VERTSB_HORZSB_CTLSetScrollInfoTRUEFALSE
Windows

SCROLLINFO

```
typedef struct tagSCROLLINFO

{
```

```
    UINT cbSize ;// set to sizeof (SCROLLINFO)
    UINT fMask ;  // values to set or get
    int  nMin ;      // minimum range value
    int  nMax ;   // maximum range value
    UINT nPage ;  // page size
    int  nPos ;   // current position
    int  nTrackPos ;// current tracking position
}
SCROLLINFO, * PSCROLLINFO ;
```

SCROLLINFO

```
SCROLLINFO si ;
```

SetScrollInfoGetScrollInfocbSize

```
si.cbSize = sizeof (si) ;
```

```
si.cbSize = sizeof (SCROLLINFO) ;
```

WindowsWindows

fMaskSIFCOR(|)

SetScrollInfoSIF_RANGEnMinnMaxGetScrollInfoSIF_RANGE
nMinnMax

SIF_POSSetScrollInfonPosGetScrollInfoSIF_POS

SIF_PAGESetScrollInfonPageGetScrollInfoSIF_PAGE

SB_THUMBTRACKSB_THUMBPOSITIONWM_VSCROLLWM_HSCROLL
GetScrollInfoSIF_TRACKPOSSCROLLINFOnTrackPos32

SetScrollInfoSIF_DISABLENOSCROLL

SIF_ALLSIF_RANGESIF_POSSIF_PAGESIF_TRACKPOSWM_SIZE
SetScrollInfoSIF_TRACKPOS

SYSMETS20NUMLINES-10NUMLINES-1

SYSMETS2SYSMETS2WM_CREATEWM_SIZE

```
iVscrollMax = max (0, NUMLINES - cyClient / cyChar) ;

SetScrollRange (hwnd, SB_VERT, 0, iVscrollMax, TRUE) ;
```

NUMLINES7550cyCharcyClient75500250049
1150252574

SCROLLINFOSetScrollInfo

```
si.cbSize   = sizeof (SCROLLINFO) ;

si.cbMask   = SIF_RANGE | SIF_PAGE ;

si.nMin     = 0 ;

si.nMax     = NUMLINES - 1 ;
```

```
si.nPage    = cyClient / cyChar ;

SetScrollInfo (hwnd, SB_VERT, &si, TRUE) ;
```

Windowssi.nMax - si.nPage +1si.nMaxNUM
74si.nPage5074 - 50 + 125

nPage75WindowsSetScrollInfo
SIF_DISABLENOSCROLLWindows

SYSMETS

SYSMETS3SYSMETS4-4SetScrollInfoGetScrollInfo

4-4 SYSMETS3

```
SYSMETS3.C

/*------------------------------------------------------------

  SYSMETS3.C -- System Metrics Display Program No. 3

               (c) Charles Petzold, 1998

------------------------------------------------------------*/

#include <windows.h>

#include "sysmets.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns

          PSTR szCmdLine, int iCmdShow)
```

```
{
   static TCHAR szAppName[] = TEXT ("SysMets3") ;
   HWND  hwnd ;
   MSG    msg ;
   WNDCLASS      wndclass ;

   wndclass.style    = CS_HREDRAW | CS_VREDRAW ;
   wndclass.lpfnWndProc= WndProc ;
   wndclass.cbClsExtra   = 0 ;
   wndclass.cbWndExtra   = 0 ;
   wndclass.hInstance    = hInstance ;
   wndclass.hIcon        = LoadIcon (NULL, IDI_APPLICATION) ;
   wndclass.hCursor      = LoadCursor (NULL, IDC_ARROW) ;
   wndclass.hbrBackground= (HBRUSH) GetStockObject (WHITE_
   wndclass.lpszMenuName         = NULL ;
   wndclass.lpszClassName        = szAppName ;

   if (!RegisterClass (&wndclass))
```

```
{
MessageBox (NULL, TEXT ("Program requires Windows NT!"),
    szAppName, MB_ICONERROR) ;
     return 0 ;
}

hwnd = CreateWindow (szAppName, TEXT ("Get System Metr
            WS_OVERLAPPEDWINDOW | WS_VSCROLL | WS_HSC
            CW_USEDEFAULT, CW_USEDEFAULT,
            CW_USEDEFAULT, CW_USEDEFAULT,
            NULL, NULL, hInstance, NULL) ;
ShowWindow (hwnd, iCmdShow) ;
UpdateWindow (hwnd) ;

while (GetMessage (&msg, NULL, 0, 0))
{
      TranslateMessage (&msg) ;
      DispatchMessage (&msg) ;
}
```

```
    return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
    static int  cxChar, cxCaps, cyChar, cxClient, cyClient, iMaxWid
    HDC    hdc ;
    int    i, x, y, iVertPos, iHorzPos, iPaintBeg, iPaintEnd ;
    PAINTSTRUCT ps ;
    SCROLLINFO  si ;
    TCHAR       szBuffer[10] ;
    TEXTMETRIC  tm ;

    switch (message)
    {
    case WM_CREATE:
          hdc = GetDC (hwnd) ;
    GetTextMetrics (hdc, &tm) ;
    cxChar = tm.tmAveCharWidth ;
```

```
    cxCaps = (tm.tmPitchAndFamily & 1 ? 3 : 2) * cxChar / 2 ;
    cyChar = tm.tmHeight + tm.tmExternalLeading ;

    ReleaseDC (hwnd, hdc) ;
          // Save the width of the three columns
    iMaxWidth = 40 * cxChar + 22 * cxCaps ;
    return 0 ;

case WM_SIZE:
    cxClient = LOWORD (lParam) ;
    cyClient = HIWORD (lParam) ;

          // Set vertical scroll bar range and page size
    si.cbSize     = sizeof (si) ;
    si.fMask      = SIF_RANGE | SIF_PAGE ;
    si.nMin       = 0 ;
    si.nMax       = NUMLINES - 1 ;
    si.nPage      = cyClient / cyChar ;
    SetScrollInfo (hwnd, SB_VERT, &si, TRUE) ;
          // Set horizontal scroll bar range and page size
```

```
    si.cbSize     = sizeof (si) ;
    si.fMask      = SIF_RANGE | SIF_PAGE ;
    si.nMin       = 0 ;
    si.nMax       = 2 + iMaxWidth / cxChar ;
    si.nPage      = cxClient / cxChar ;
    SetScrollInfo (hwnd, SB_HORZ, &si, TRUE) ;
    return 0 ;

case WM_VSCROLL:
          // Get all the vertical scroll bar information
    si.cbSize     = sizeof (si) ;
    si.fMask      = SIF_ALL ;
    GetScrollInfo (hwnd, SB_VERT, &si) ;
          // Save the position for comparison later on
    iVertPos = si.nPos ;
    switch (LOWORD (wParam))
    {
    case   SB_TOP:
```

```
        si.nPos      = si.nMin ;
        break ;

    case   SB_BOTTOM:
             si.nPos      = si.nMax ;
       break ;

    case SB_LINEUP:
        si.nPos -    = 1 ;
        break ;

    case   SB_LINEDOWN:
        si.nPos += 1 ;
        break ;

    case   SB_PAGEUP:
        si.nPos -= si.nPage ;
        break ;
```

```
case   SB_PAGEDOWN:
       si.nPos += si.nPage ;
       break ;

case   SB_THUMBTRACK:
       si.nPos = si.nTrackPos ;
       break ;

default:
break ;
}
       // Set the position and then retrieve it.  Due to adjustmer
       //  by Windows it may not be the same as the value set.

si.fMask = SIF_POS ;
SetScrollInfo (hwnd, SB_VERT, &si, TRUE) ;
GetScrollInfo (hwnd, SB_VERT, &si) ;

       // If the position has changed, scroll the window and upd
if (si.nPos != iVertPos)
```

```
{
        ScrollWindow (hwnd, 0, cyChar * (iVertPos - si.nPos),
                                    NULL, NULL) ;
        UpdateWindow (hwnd) ;
  }
  return 0 ;
  case WM_HSCROLL:
              // Get all the vertical scroll bar information
        si.cbSize = sizeof (si) ;
        si.fMask  = SIF_ALL ;

              // Save the position for comparison later on
        GetScrollInfo (hwnd, SB_HORZ, &si) ;
        iHorzPos = si.nPos ;

        switch (LOWORD (wParam))
        {
  case   SB_LINELEFT:
        si.nPos -= 1 ;
```

```
        break ;

    case   SB_LINERIGHT:
        si.nPos += 1 ;
        break ;

    case   SB_PAGELEFT:
        si.nPos -= si.nPage ;
        break ;

    case   SB_PAGERIGHT:
        si.nPos += si.nPage ;
        break ;

    case   SB_THUMBPOSITION:
        si.nPos = si.nTrackPos ;
        break ;

        default :
```

```
        break ;
}
        // Set the position and then retrieve it.  Due to adjustmer
        //   by Windows it may not be the same as the value set.

        si.fMask = SIF_POS ;
        SetScrollInfo (hwnd, SB_HORZ, &si, TRUE) ;
        GetScrollInfo (hwnd, SB_HORZ, &si) ;

        // If the position has changed, scroll the window

if (si.nPos != iHorzPos)
{
        ScrollWindow (hwnd, cxChar * (iHorzPos - si.nPos), 0,
                              NULL, NULL) ;
}
        return 0 ;
case WM_PAINT :
        hdc = BeginPaint (hwnd, &ps) ;
```

```
        // Get vertical scroll bar position
si.cbSize = sizeof (si) ;
si.fMask  = SIF_POS ;
GetScrollInfo (hwnd, SB_VERT, &si) ;
iVertPos = si.nPos ;

        // Get horizontal scroll bar position
GetScrollInfo (hwnd, SB_HORZ, &si) ;
iHorzPos = si.nPos ;
        // Find painting limits
iPaintBeg = max (0, iVertPos + ps.rcPaint.top / cyChar) ;
iPaintEnd = min (     NUMLINES - 1,
                iVertPos + ps.rcPaint.bottom / cyChar) ;

for (i = iPaintBeg ; i <= iPaintEnd ; i++)
{
        x = cxChar * (1 - iHorzPos) ;
        y = cyChar * (i - iVertPos) ;
```

```
            TextOut (hdc, x, y,
                  sysmetrics[i].szLabel,
                  lstrlen (sysmetrics[i].szLabel)) ;

            TextOut (hdc, x + 22 * cxCaps, y,
                  sysmetrics[i].szDesc,
                  lstrlen (sysmetrics[i].szDesc)) ;

            SetTextAlign (hdc, TA_RIGHT | TA_TOP) ;
            TextOut (hdc, x + 22 * cxCaps + 40 * cxChar, y, szBuffer
                  wsprintf (szBuffer, TEXT ("%5d"),
                  GetSystemMetrics (sysmetrics[i].iIndex))) ;

            SetTextAlign (hdc, TA_LEFT | TA_TOP) ;
      }

            EndPaint (hwnd, &ps) ;
            return 0 ;

      case   WM_DESTROY :
```

```
        PostQuitMessage (0) ;

        return 0 ;

}

   return DefWindowProc (hwnd, message, wParam, lParam) ;

}
```

WindowsWM_VSCROLLWM_HSCROLLSetScrollInfo GetScrollInfoSetScrollInfoWindowsGetScrollInfo

SYSMETS3ScrollWindowWindowsScrollWindowExSYSMETS3

ScrollWindowNULLWindowsWM_PAINTInvalidateRect ScrollWindowGDIGDIWindows

WM_HSCROLLSB_THUMBPOSITIONSB_THUMBTRACK

WM_VSCROLLSB_THUMBTRACKSB_THUMBPOSITION PCGetSystemMetricsSB_SLOWMACHINE

WM_PAINTSYSMETS3WM_PAINT

WindowsWindowsPC

SYSMETS3SYSMETS3SB_TOPSB_BOTTOMWM_VSCROLL

GDIGraphics Device InterfaceWindowsGDIWindows
WindowsGDIWindowsGDI

GDIGDIGDImetafile

GDI

GDIGDI

GDI

Windows 98Microsoft Windows NTGDI32.DLLWindows
GDI32.DLL16GDI.EXEWindows NTGDI

GDI

PCGDIWindowsWindowsGDI

PC

Windows

Windows GDICCCCGDI

Windows032,767Windows

SYS

WindowsGDIWindowsWindows
high-Colortrue-color

CWindowsWindows
DirectX

GDI

GDI

- GetDCRealseDCWM_PAINTI
  EndPaintUSERGDIWM_PAINT
- SYSMETSGetTextMetrics DEVCAPS1
- TextOutGDI
- SetTextColorTextOut SYSMETS
  SetTextAlignGDITextOutSetGet
- **GDI**GDIGDICreatePen
  CreatePenIndirectExtCreatePenGDIGDI
  GDIGDI

GDI

- GDIBezier

- GDI

- GDIGDI
  Windows 3.0DIB

- GDIWindowsWindows
  TrueTypeGDIWindows

GDI

- GDIWindows

- **Metafile**MetafileGDIMetafile metafile
- GDI
- GDI
-
- 256Windows20236
-

DCWindowsGDIWindows

GDIGDIWindowsTextOut
TextOut

Windows

WM_PAINTBeginPaintEndPaint

```
hdc = BeginPaint (hwnd, &ps) ;



EndPaint (hwnd, &ps) ;
```

psPAINTSTRUCThdcBeginPaint
rcPaintBeginPaintBeginPaint

WindowsWM_PAINT

```
hdc = GetDC (hwnd) ;


ReleaseDC (hwnd, hdc) ;
```

hwndBeginPaintEndPaintGetDC

Windows

```
hdc = GetWindowDC (hwnd) ;


ReleaseDC (hwnd, hdc) ;
```

frameGetWindowDCWM_NCPAINTWindows

BeginPaintGetDCGetWindowDCCreateDC

```
hdc = CreateDC (pszDriver, pszDevice, pszOutput, pData) ;


DeleteDC (hdc) ;
```

```
hdc = CreateDC (TEXT ("DISPLAY"), NULL, NULL, NULL) ;
```

GetDCNULL

CreateICCreateDC

```
hdc = CreateIC (TEXT ("DISPLAY"), NULL, NULL, NULL) ;
```

```
hdcMem = CreateCompatibleDC (hdc) ;



DeleteDC (hdcMem) ;
```

GDI

metafileGDImetafilemetafile

```
hdcMeta = CreateMetaFile (pszFilename) ;



hmf = CloseMetaFile (hdcMeta) ;
```

metafilehdcMetaGDImetafileCloseMetaFilemetafilehmf
metafile

GetDeviceCaps

```
iValue = GetDeviceCaps (hdc, iIndex) ;
```

iIndexWINGDI.H29iIndexHORZRESGetDeviceCaps
iIndexVERTRESGetDeviceCapshdcGetDeviceCaps

GetDeviceCapsGetDeviceCaps

DEVCAPS1

5-1DEVCAPS1 GetDeviceCaps

5-1　　DEVCAPS1

```
DEVCAPS1.C

/*--------------------------------------------------------------------

  DEVCAPS1.C -- Device Capabilities Display Program No. 1

          (c) Charles Petzold, 1998

--------------------------------------------------------------------*/

#include <windows.h>

#define NUMLINES ((int) (sizeof devcaps / sizeof devcaps [0]))

struct

{

   int    iIndex ;

   TCHAR *szLabel ;

   TCHAR *szDesc ;

}
```

```
devcaps [] =

{

   HORZSIZE,     TEXT ("HORZSIZE"),TEXT ("Width in millimeters:
   VERTSIZE,     TEXT ("VERTSIZE"),TEXT ("Height in millimeters:"
   HORZRES,      TEXT ("HORZRES"),     TEXT ("Width in pixels:"),
  VERTRES,      TEXT ("VERTRES"),     TEXT ("Height in raster line
   BITSPIXEL,    TEXT ("BITSPIXEL"),TEXT ("Color bits per pixel:"),
   PLANES,       TEXT ("PLANES"),   TEXT ("Number of color plane
  NUMBRUSHES,   TEXT ("NUMBRUSHES"),  TEXT ("Number of de
   NUMPENS,      TEXT ("NUMPENS"),     TEXT ("Number of device
  NUMMARKERS,   TEXT ("NUMMARKERS"),  TEXT ("Number of de
   NUMFONTS,     TEXT ("NUMFONTS"),    TEXT ("Number of devi
   NUMCOLORS,    TEXT ("NUMCOLORS"),   TEXT ("Number of dev
   PDEVICESIZE,  TEXT ("PDEVICESIZE"),TEXT ("Size of device str
   ASPECTX,      TEXT ("ASPECTX"),     TEXT ("Relative width of pi
   ASPECTY,      TEXT ("ASPECTY"),     TEXT ("Relative height of p
   ASPECTXY,     TEXT ("ASPECTXY"),    TEXT ("Relative diagonal
   LOGPIXELSX,   TEXT ("LOGPIXELSX"),  TEXT ("Horizontal dots p
   LOGPIXELSY,   TEXT ("LOGPIXELSY"),  TEXT ("Vertical dots per
```

```
   SIZEPALETTE,  TEXT ("SIZEPALETTE"),TEXT ("Number of palett
   NUMRESERVED,  TEXT ("NUMRESERVED"),TEXT ("Reserved pa
   COLORRES,     TEXT ("COLORRES"),    TEXT ("Actual color reso
} ;
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
         PSTR szCmdLine, int iCmdShow)
{
   static TCHAR szAppName[] = TEXT ("DevCaps1") ;
   HWND         hwnd ;
   MSG          msg ;
   WNDCLASS     wndclass ;

   wndclass.style       = CS_HREDRAW | CS_VREDRAW ;
   wndclass.lpfnWndProc= WndProc ;
   wndclass.cbClsExtra   = 0 ;
   wndclass.cbWndExtra = 0 ;
   wndclass.hInstance    = hInstance ;
```

```
wndclass.hIcon        = LoadIcon (NULL, IDI_APPLICATION) ;
wndclass.hCursor      = LoadCursor (NULL, IDC_ARROW) ;
wndclass.hbrBackground= (HBRUSH) GetStockObject (WHITE_
wndclass.lpszMenuName= NULL ;
wndclass.lpszClassName= szAppName ;

if (!RegisterClass (&wndclass))
{
      MessageBox (  NULL, TEXT ("This program requires Wind
                        szAppName, MB_ICONERROR) ;
      return 0 ;
}

hwnd = CreateWindow (szAppName, TEXT ("Device Capabiliti
                        WS_OVERLAPPEDWINDOW,
                        CW_USEDEFAULT, CW_USEDEFAULT,
                  CW_USEDEFAULT, CW_USEDEFAULT,
               NULL, NULL, hInstance, NULL) ;
```

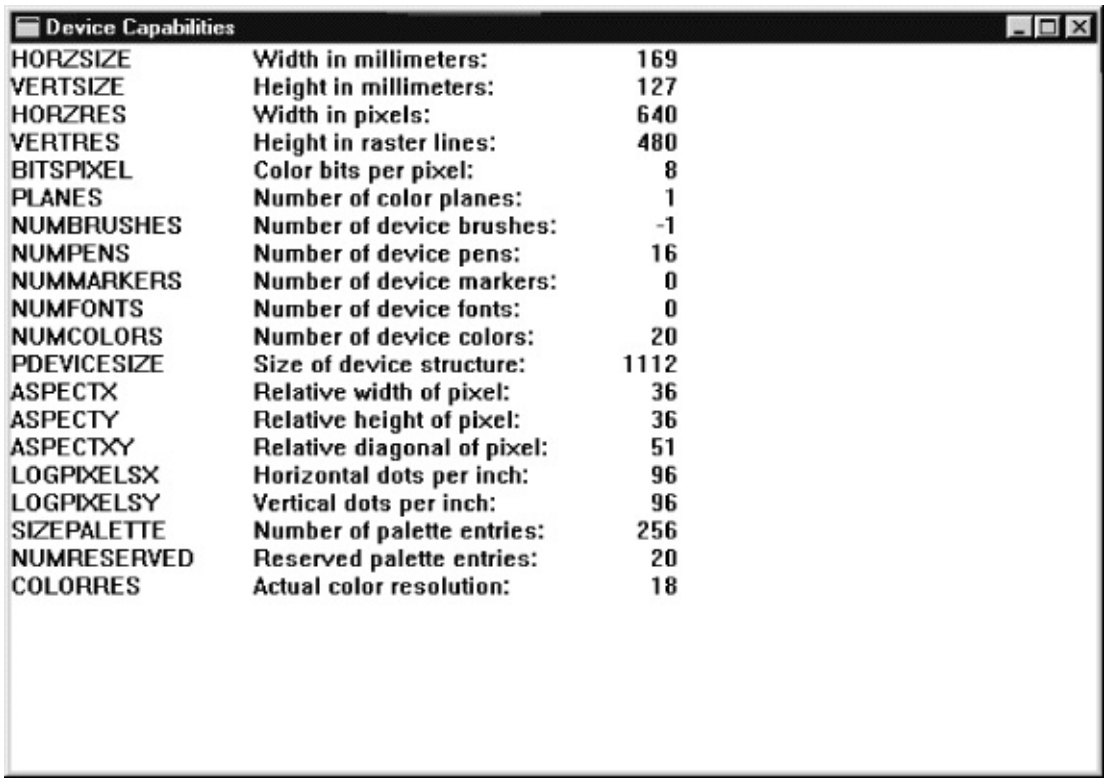
```
   ShowWindow (hwnd, iCmdShow) ;
   UpdateWindow (hwnd) ;

   while (GetMessage (&msg, NULL, 0, 0))
   {
         TranslateMessage (&msg) ;
         DispatchMessage (&msg) ;
   }
   return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
   static int    cxChar, cxCaps, cyChar ;
   TCHAR         szBuffer[10] ;
   HDC           hdc ;
   int           i ;
   PAINTSTRUCT   ps ;
   TEXTMETRIC    tm ;
```

```
switch (message)
{
case WM_CREATE:
        hdc = GetDC (hwnd) ;
        GetTextMetrics (hdc, &tm) ;
        cxChar= tm.tmAveCharWidth ;
        cxCaps= (tm.tmPitchAndFamily & 1 ? 3 : 2) * cxChar / 2 ;
        cyChar= tm.tmHeight + tm.tmExternalLeading ;

        ReleaseDC (hwnd, hdc) ;
        return 0 ;

case WM_PAINT:
        hdc = BeginPaint (hwnd, &ps) ;
        for (i = 0 ; i < NUMLINES ; i++)
        {
              TextOut (     hdc, 0, cyChar * i,
```

```
                devcaps[i].szLabel,
                lstrlen (devcaps[i].szLabel)) ;

            TextOut (     hdc, 14 * cxCaps, cyChar * i,
                devcaps[i].szDesc,
                lstrlen (devcaps[i].szDesc)) ;

            SetTextAlign (hdc, TA_RIGHT | TA_TOP) ;
            TextOut (hdc, 14*cxCaps+35*cxChar, cyChar*i, szBu
                wsprintf (szBuffer, TEXT ("%5d"),
                GetDeviceCaps (hdc, devcaps[i].iIndex))) ;

        SetTextAlign (hdc, TA_LEFT | TA_TOP) ;
        }

        EndPaint (hwnd, &ps) ;
        return 0 ;

case WM_DESTROY:
```

```
        PostQuitMessage (0) ;

        return 0 ;

    }

    return DefWindowProc (hwnd, message, wParam, lParam) ;

}
```

SYSMETS1256640×480VGA5-1



5-1　256640×480VGADEVCAPS1

1Windows1GetDeviceCaps

dpi300600dpi1024×768

Windows33%1.33:14:3Thomas

Windows4:34:33:4

WindowsWindowsIBM AdapterCGA640×200Enhanced Graphics AdapterEGA640› Hercules Graphics Card720×3484:34:3

Windows

- 640×480
- 800×600
- 1024×768
- 1280×1024
- 1600×1200

4:31280×10244:3

WindowsSM_CXSCREENSM_CYSCREENGetSystemMetricsDEVCAPS1 HORZRESVERTRESGetDeviceCaps

HORZSIZEVERTSIZE/Platform

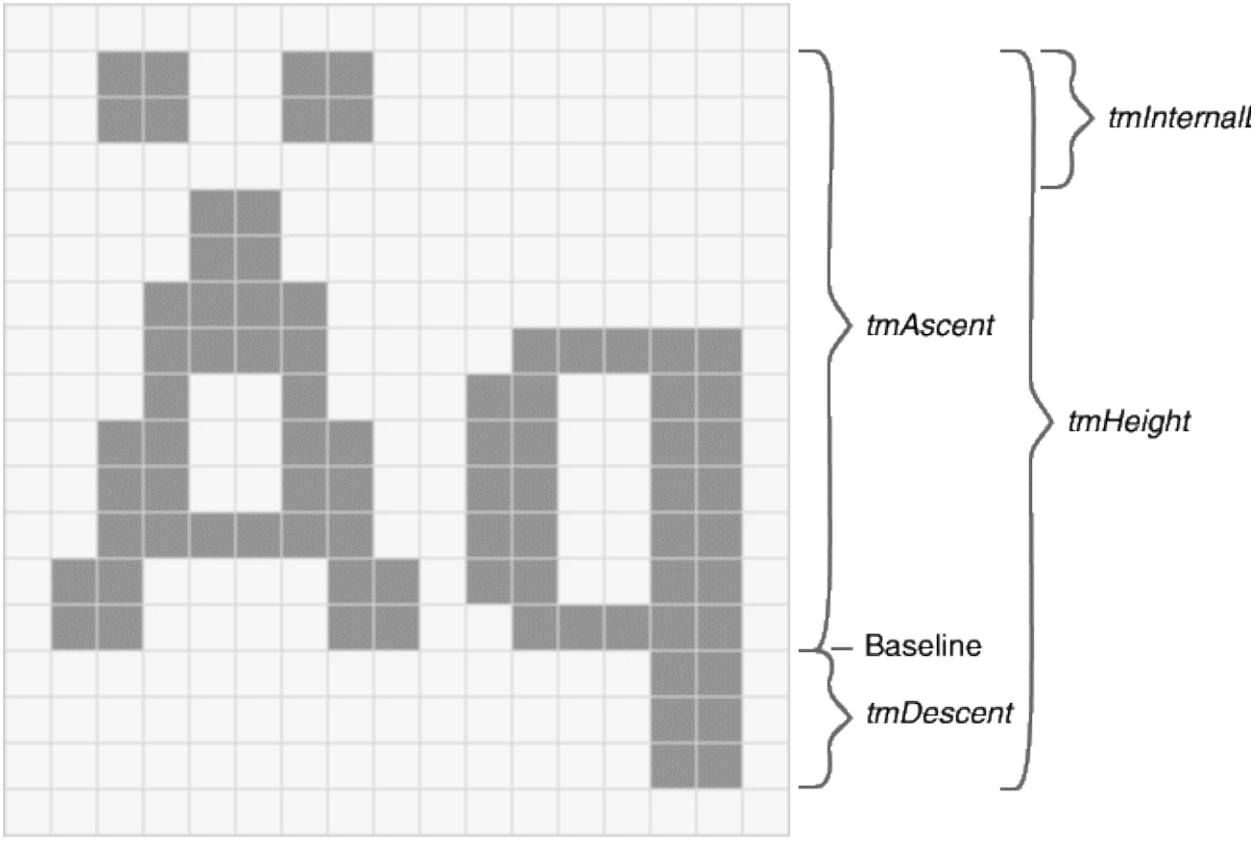
Multimedia Services/GDI/Device Contexts/Device Context Reference/Device Context Functions/GetDeviceCapsWindows

Windows16Windows NTWindowsHORZSIZEVERTSIZE Windows 95HORZSIZEVERTSIZEHORZRESVERTRES LOGPIXELSXLOGPIXELSY

640×4801024×768

11/7211/72

jpqy1010/72TEXTMETRICtmHeight tmInternalLeading5-2 4-3



5-2 TEXTMETRIC

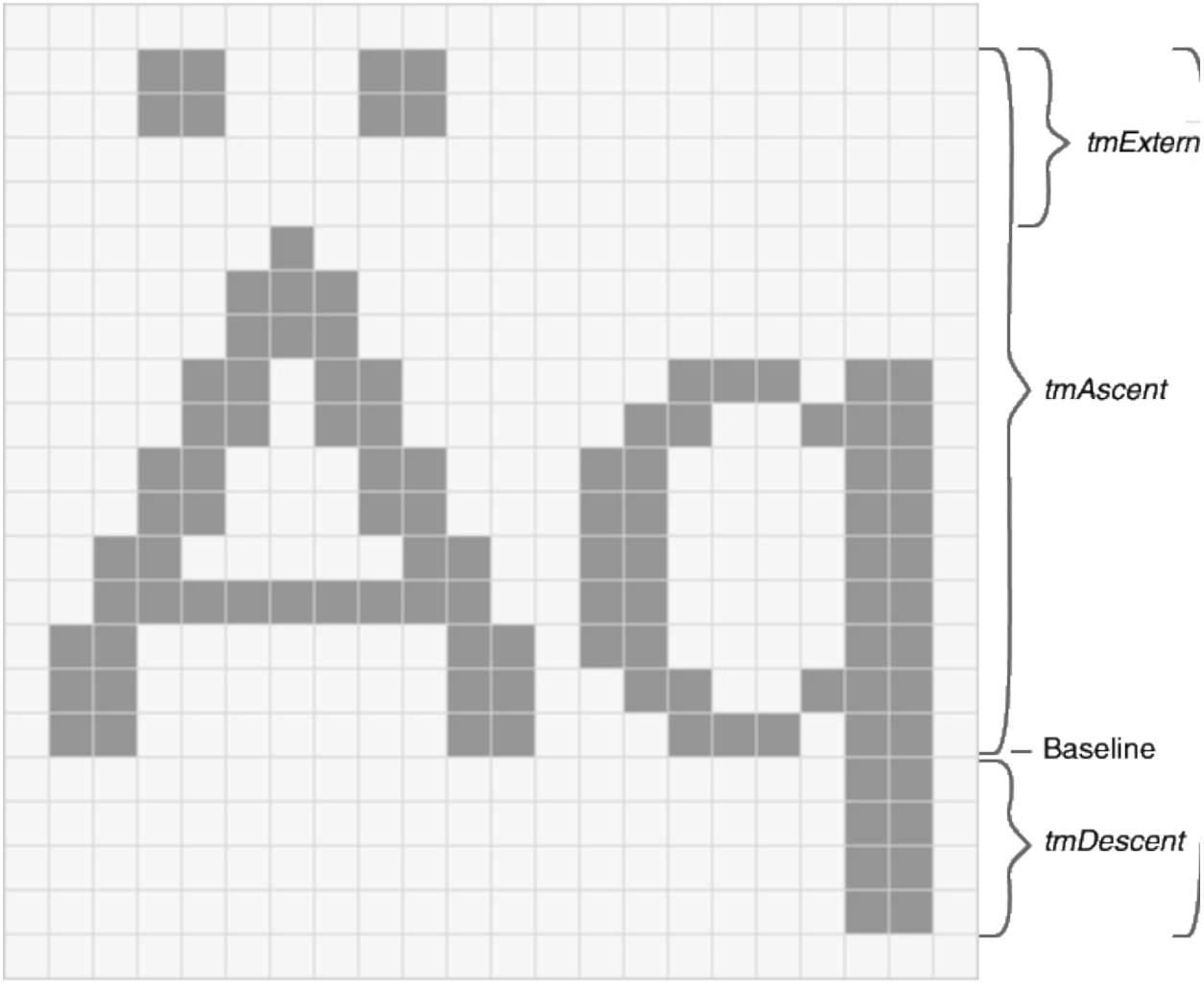
TEXTMETRICtmHeight1212/721/61010

1010

Windows101210

5-296101010/729613tmHeighttmInternalLeading12
12/729616tmHeight

5-31201010/7212016tmHeighttmInternalLeading12
20tmHeight



5-3
FONTMETRIC

WindowsGetDeviceCapsLOGPIXELSX
LOGPIXELSYLOGPIXELS

HORZSIZEVERTSIZEGetDeviceCaps
HORZRESVERTRESLOGPIXELSXLOGPIXELSY

$$\text{水平大小(mm)} = 25.4 \times \frac{\text{水平解析度(圖素)}}{\text{邏輯圖素}X\text{(每英寸的點數)}}$$

$$\text{垂直大小(mm)} = 25.4 \times \frac{\text{垂直解析度(圖素)}}{\text{邏輯圖素}Y\text{(每英寸的點數)}}$$

25.4

Windows

17129640×480Windows5310Aq7Color
Graphics AdapterWindows

43640×4801310

1010Windows1081010

Windows NTHORZSIZEVERTSIZEWindows16HORZRES
VERTRESLOGPIXELSXLOGPIXELSYWindows
LOGPIXELSXLOGPIXELSY96120 dpi

Windows NTHORZSIZEVERTSIZEHORZSIZEVERTSIZE320240
HORZRESVERTRESLOGPIXELSXLOGPIXELSY
GetDeviceCapsWindows 98HORZSIZ

GetDeviceCapsASPECTXASPECTYASPECTXY
ASPECTXASPECTYASPECTXYASPECTXASPECTY

2

Full-Color24888

High-Color16565

25688

1641616IBM

GetDeviceCaps

```
iPlanes = GetDeviceCaps (hdc, PLANES) ;
```

```
iBitsPixel = GetDeviceCaps (hdc, BITSPIXEL) ;
```

1

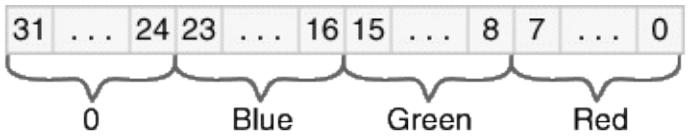
```
iColors = 1 << (iPlanes * iBitsPixel) ;
```

NUMCOLORS

```
iColors = GetDeviceCaps (hdc, NUMCOLORS) ;
```

256NUMCOLORSGetDeviceCapsWindows20236
WindowsHigh-ColorTrue-ColorNUMCOLORSGetDeviceCaps-1
PLANESBITSPIXELiColors

GDICOLORREF32COLORREFRGB
COLORREF5-4



5-4 32COLORREF

088COLORREF

RGBWindowsWINGDI.HRGBRGB

```
#define RGB(r,g,b) ((COLORREF)(((BYTE)(r) | \
                ((WORD)((BYTE)(g)) << 8)) | \
                (((DWORD)(BYTE)(b)) << 16)))
```

```
RGB (255, 255, 0)
```

0x0000FFFF0255GetRValueGetGValueGetBValue
COLORREFRGBWindows

16256WindowsGetNearestColor

```
crPureColor = GetNearestColor (hdc, crColor) ;
```

WindowsGDITextOutWindows

Windows5-1Windows

Mapping Mode	MM_TEXT	SetMapMode	GetMapMode
Window Origin	(0, 0)	SetWindowOrgEx OffsetWindowOrgEx	GetWindowOrgEx
Viewport Origin	(0, 0)	SetViewportOrgEx OffsetViewportOrgEx	GetViewportOrgEx
Window Extents	(1, 1)	SetWindowExtEx SetMapMode ScaleWindowExtEx	GetWindowExtEx
Viewport Extents	(1, 1)	SetViewportExtEx SetMapMode ScaleViewportExtEx	GetViewportExtEx
Pen	BLACK_PEN	SelectObject	SelectObject

Brush	WHITE_BRUSH	SelectObject	SelectObject
Font	SYSTEM_FONT	SelectObject	SelectObject
Bitmap	None	SelectObject	SelectObject
Current Position	(0, 0)	MoveToEx LineTo PolylineTo PolyBezierTo	GetCurrentPositionEx
Background Mode	OPAQUE	SetBkMode	GetBkMode
Background Color	White	SetBkColor	GetBkColor
Text Color	Black	SetTextColor	GetTextColor
Drawing Mode	R2_COPYPEN	SetROP2	GetROP2
Stretching Mode	BLACKONWHITE	SetStretchBltMode	GetStretchBltMode
Polygon Fill	ALTERNATE	SetPolyFillMode	GetPolyFillMode

Mode			
Intercharacter Spacing	0	SetTextCharacterExtra	GetTextCharacterExtı
Brush Origin	(0, 0)	SetBrushOrgEx	GetBrushOrgEx
Clipping Region	None	SelectObject SelectClipRgn IntersectClipRgn OffsetClipRgn ExcludeClipRect SelectClipPath	GetClipBox

GetDCBeginPaintWindowsReleaseDCEndPaint

```
case WM_PAINT:
   hdc = BeginPaint (hwnd, &ps) ;



   EndPaint (hwnd, &ps) ;
   return 0 ;
```

GetDCBeginPaintCS_OWNDC

```
wndclass.style = CS_HREDRAW | CS_VREDRAW | CS_OWNDC ;
```

CS_OWNDCWM_CREATE

```
case       WM_CREATE:

     hdc = GetDC (hwnd) ;



      ReleaseDC (hwnd, hdc) ;
```

CS_OWNDCGetDCBeginPaintGetWindowDCCS_OWNDC
Windows NTCS_OWNDC

```
idSaved = SaveDC (hdc) ;
```

SaveDC

```
RestoreDC (hdc, idSaved) ;
```

RestoreDCSaveDC

SaveDCRestoreDCPUSHPOPSaveDC

```
SaveDC (hdc) ;
```

SaveDC

```
RestoreDC (hdc, -1) ;
```

SaveDC

WindowsSetPixelGetPixelGDIGDISetPixelxy

SetPixelGetPixelSetPixel

Windows GDISetPixelGetPixel CONNECTSetPixel WHATCLR GetPixel

SetPixelxy

```
SetPixel (hdc, x, y, crColor) ;
```

xyCOLORREF

GetPixel

```
crColor = GetPixel (hdc, x, y) ;
```

WindowsWindows 987

- LineTo

- PolylinePolylineTo
- PolyPolyline
- Arc
- PolyBezierPolyBezierTo

Windows NT3

- ArcToAngleArc
- PolyDraw

Windows 98

- Rectangle
- Ellipse
- RoundRect
- Pie
- Chord

LineToPolylineToPolyBezierToArcTo

```
MoveToEx (hdc, xBeg, yBeg, NULL) ;
```

```
LineTo (hdc, xEnd, yEnd) ;
```

MoveToExLineToGDI0,0LineTo

Windows16MoveToxy1632Windows3232C3264
MoveToMoveToMoveToEx

MoveToExPOINTPOINTxyNULL

Windows　　983216-32,76832,767Windows

```
GetCurrentPositionEx (hdc, &pt) ;
```

ptPOINT

100hwndhdcxy

```
GetClientRect (hwnd, &rect) ;

for (       x = 0 ; x < rect.right ; x+= 100)

{

    MoveToEx (hdc, x, 0, NULL) ;

  LineTo (hdc, x, rect.bottom) ;
```

```
}
for (y = 0 ; y < rect.bottom ; y += 100)
{
   MoveToEx (hdc, 0, y, NULL) ;
   LineTo (hdc, rect.right, y) ;
}
```

510

```
POINT apt[5] = { 100, 100, 200, 100, 200, 200, 100, 200, 100, 1
```

MoveToExLineTo

```
MoveToEx (hdc, apt[0].x, apt[0].y, NULL) ;
for (      i = 1 ; i < 5 ; i++)
   LineTo (hdc, apt[i].x, apt[i].y) ;
```

LineToLineTo

Polyline

```
Polyline (hdc, apt, 5) ;
```

(sizeof (apt) / sizeof (POINT))PolylineMove
PolylineTo

```
MoveToEx (hdc, apt[0].x, apt[0].y, NULL) ;

PolylineTo (hdc, apt + 1, 4) ;
```

PolylinePolylineTo5-2SINEWAVE

```
5-2  SINEWAVE

SINEWAVE.C

/*------------------------------------------------------------------

  SINEWAVE.C -- Sine Wave Using Polyline

          (c) Charles Petzold, 1998

------------------------------------------------------------------*/

#include <windows.h>

#include <math.h>

#define NUM 1000

#define TWOPI      (2 * 3.14159)

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;


int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns

                PSTR szCmdLine, int iCmdShow)

{

   static TCHAR szAppName[] = TEXT ("SineWave") ;
```

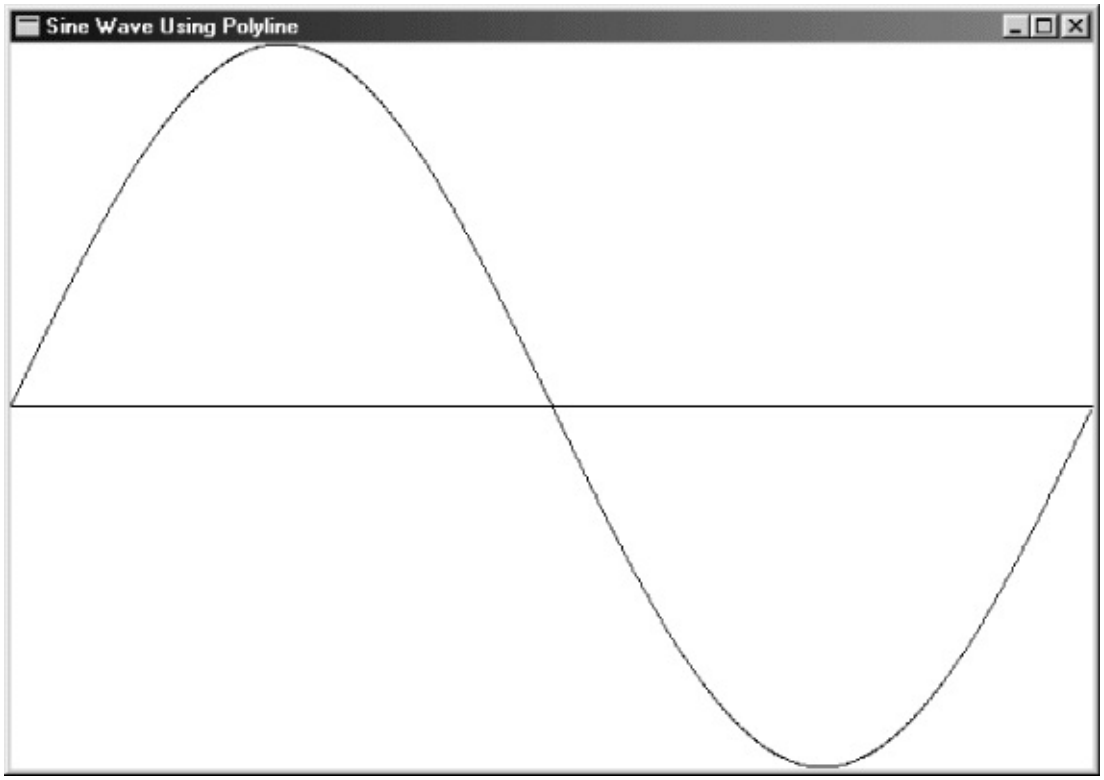
```
HWND         hwnd ;
MSG          msg ;
WNDCLASS     wndclass ;

wndclass.style       = CS_HREDRAW | CS_VREDRAW ;
wndclass.lpfnWndProc= WndProc ;
wndclass.cbClsExtra   = 0 ;
wndclass.cbWndExtra   = 0 ;
wndclass.hInstance    = hInstance ;
wndclass.hIcon = LoadIcon (NULL, IDI_APPLICATION) ;
wndclass.hCursor      = LoadCursor (NULL, IDC_ARROW) ;
wndclass.hbrBackground= (HBRUSH) GetStockObject (WHITE_
wndclass.lpszMenuName  = NULL ;
wndclass.lpszClassName = szAppName ;

if (!RegisterClass (&wndclass))
{
      MessageBox (  NULL, TEXT ("Program requires Windows N
```

```
                szAppName, MB_ICONERROR) ;
                return 0 ;
    }

    hwnd = CreateWindow ( szAppName, TEXT ("Sine Wave Using
                    WS_OVERLAPPEDWINDOW,
                     CW_USEDEFAULT, CW_USEDEFAULT,
                     CW_USEDEFAULT, CW_USEDEFAULT,
                     NULL, NULL, hInstance, NULL) ;
    ShowWindow (hwnd, iCmdShow) ;
    UpdateWindow (hwnd) ;

    while (GetMessage (&msg, NULL, 0, 0))
    {
          TranslateMessage (&msg) ;
               DispatchMessage (&msg) ;
    }
          return msg.wParam ;
}
```

```
LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
   static int  cxClient, cyClient ;
   HDC         hdc ;
   int         i ;
   PAINTSTRUCT ps ;
   POINT       apt [NUM] ;

   switch (message)
   {
   case   WM_SIZE:
          cxClient = LOWORD (lParam) ;
         cyClient = HIWORD (lParam) ;
          return 0 ;

   case   WM_PAINT:
          hdc = BeginPaint (hwnd, &ps) ;
```

```
        MoveToEx (hdc, 0,            cyClient / 2, NULL) ;
        LineTo   (hdc, cxClient, cyClient / 2) ;

        for (i = 0 ; i < NUM ; i++)
      {
            apt[i].x = i * cxClient / NUM ;
            apt[i].y = (int) (cyClient / 2 * (1 - sin (TWOPI * i / NUM
      }

        Polyline (hdc, apt, NUM) ;
        return 0 ;

  case   WM_DESTROY:
        PostQuitMessage (0) ;
        return 0 ;
  }
  return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

1000POINTfor0999x0cxClientyPolylinePolyline1000
LineTo5-5



5-5 SINEWAVE

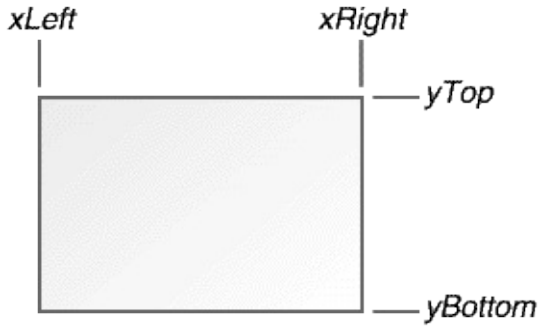
ArcEllipseArcRectangleEllipseEllipseRectangle
RoundRectChordPie

RectangleEllipseRoundRectChordPie

(bounding

```
Rectangle (hdc, xLeft, yTop, xRight, yBottom) ;
```

(xLeft, yTop)(xRight, yBottom)Rectangle5-6

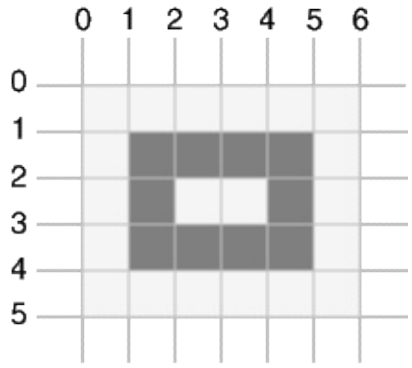


5-6
Rectangle

Windows

```
Rectangle (hdc, 1, 1, 5, 4) ;
```

Windows

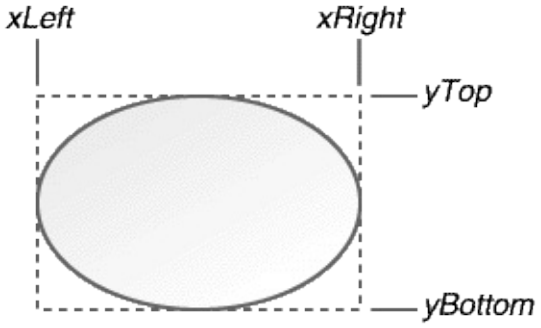


l

RectangleGDIGDI

```
Ellipse (hdc, xLeft, yTop, xRight, yBottom) ;
```

Ellipse5-7

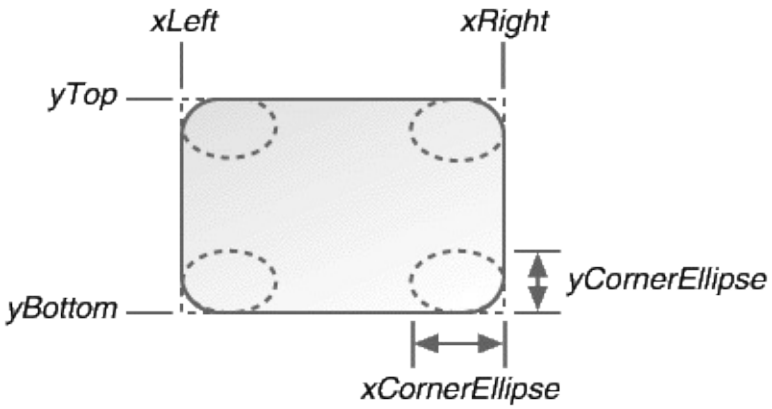


5-7
Ellipse

RectangleEllipse

```
RoundRect (hdc, xLeft, yTop, xRight, yBottom,

          xCornerEllipse, yCornerEllipse) ;
```

5-8



5-8
RoundRect

WindowsxCornerEllipseyCornerEllipsexCornerEllipse
yCornerEllipsexCornerEllipsexLeftxRightyCornerEllipseyTop
yBottomRoundRect

5-8

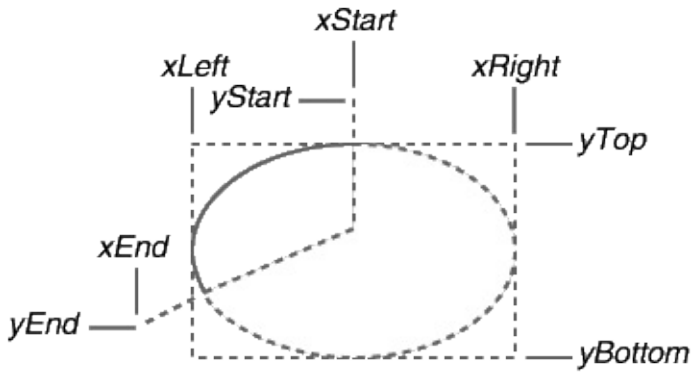
```
xCornerEllipse = (xRight - xLeft) / 4 ;

yCornerEllipse = (yBottom- yTop) / 4 ;
```

xCornerEllipseyCornerEllipse

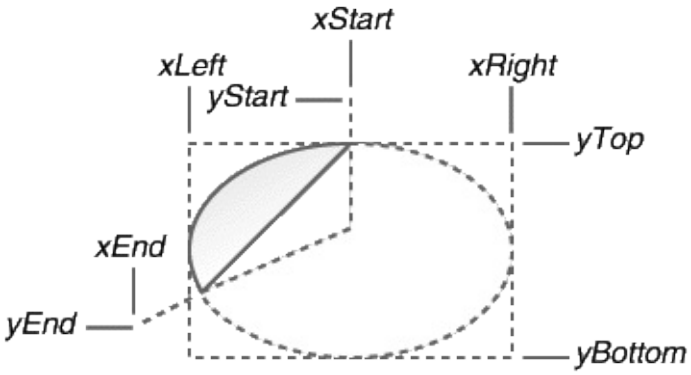
ArcChordPie

```
Arc(hdc, xLeft, yTop, xRight, yBottom, xStart, yStart, xEnd, yEnd
Chord      (hdc, xLeft, yTop, xRight, yBottom, xStart, yStart, xEnd
Pie(hdc, xLeft, yTop, xRight, yBottom, xStart, yStart, xEnd, yEnd)
```

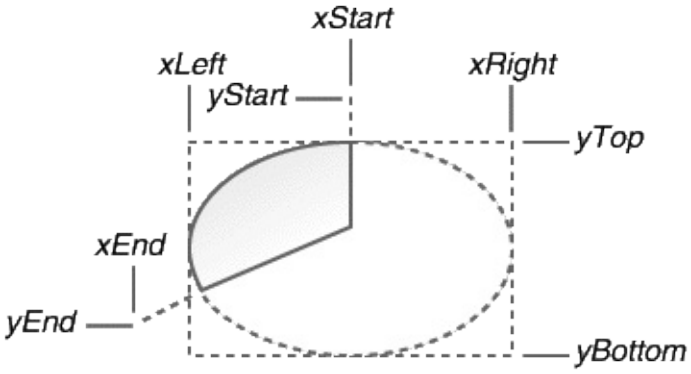
Arc5-9ChordPie5-105-11Windows(xStart,
Windows(xEndyEnd)Windows



5-9 Arc



5-10 Chord



5-11 Pie

ArcChordWindowsPieWindows

ArcChordPieWindows

5-3 LINEDEMO5-12

5-3 LINEDEMO LINEDEMO.C

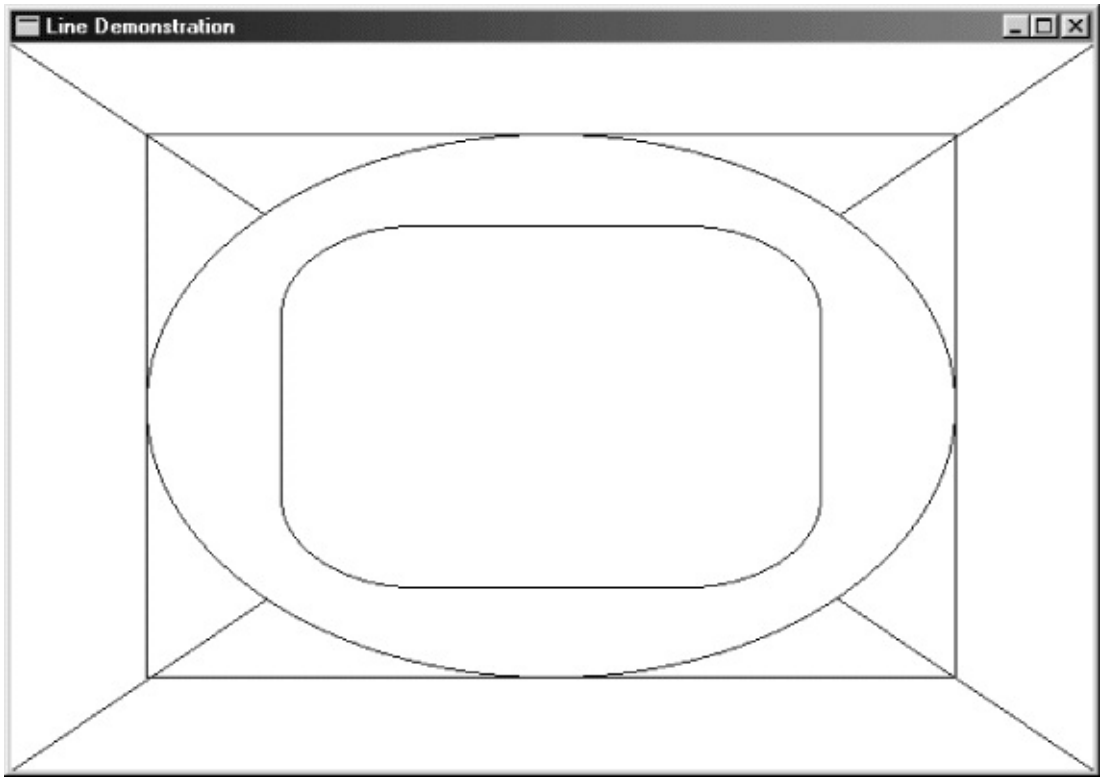
```
/*------------------------------------------------------
  LINEDEMO.C -- Line-Drawing Demonstration Program
                    (c) Charles Petzold, 1998
------------------------------------------------------*/
#include <windows.h>
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                    PSTR szCmdLine, int iCmdShow)
{
   static TCHAR szAppName[] = TEXT ("LineDemo") ;
   HWND         hwnd ;
   MSG          msg ;
   WNDCLASS     wndclass ;

   wndclass.style       = CS_HREDRAW | CS_VREDRAW ;
   wndclass.lpfnWndProc= WndProc ;
   wndclass.cbClsExtra   = 0 ;
   wndclass.cbWndExtra = 0 ;
```

```
wndclass.hInstance    = hInstance ;
wndclass.hIcon        = LoadIcon (NULL, IDI_APPLICATION) ;
wndclass.hCursor      = LoadCursor (NULL, IDC_ARROW) ;
wndclass.hbrBackground= (HBRUSH) GetStockObject (WHITE_
wndclass.lpszMenuName= NULL ;
wndclass.lpszClassName= szAppName ;

if (!RegisterClass (&wndclass))
{
      MessageBox (NULL, TEXT ("Program requires Windows N
            szAppName, MB_ICONERROR) ;
      return 0 ;
}

hwnd = CreateWindow (szAppName, TEXT ("Line Demonstrati
                        WS_OVERLAPPEDWINDOW,
                  CW_USEDEFAULT, CW_USEDEFAULT,
               CW_USEDEFAULT, CW_USEDEFAULT,
               NULL, NULL, hInstance, NULL) ;
```

```
   ShowWindow (hwnd, iCmdShow) ;
   UpdateWindow (hwnd) ;

   while (GetMessage (&msg, NULL, 0, 0))
   {
         TranslateMessage (&msg) ;
         DispatchMessage (&msg) ;
   }
         return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
   static int  cxClient, cyClient ;
   HDC         hdc ;
   PAINTSTRUCT ps ;

   switch (message)
```

```
{
case   WM_SIZE:
       cxClient = LOWORD (lParam) ;
       cyClient = HIWORD (lParam) ;
       return 0 ;

case   WM_PAINT:
       hdc = BeginPaint (hwnd, &ps) ;

       Rectangle (hdc,    cxClient / 8,     cyClient / 8,
              7 * cxClient / 8, 7 * cyClient / 8) ;

       MoveToEx  (hdc,        0,        0, NULL) ;
       LineTo    (hdc, cxClient, cyClient) ;

       MoveToEx  (hdc,        0, cyClient, NULL) ;
       LineTo    (hdc, cxClient,        0) ;
```

```
            Ellipse   (hdc,    cxClient / 8,     cyClient / 8,
                        7 * cxClient / 8, 7 * cyClient / 8) ;

            RoundRect (hdc,     cxClient / 4,     cyClient / 4,
                        3 * cxClient / 4, 3 * cyClient / 4,
                             cxClient / 4,     cyClient / 4) ;

            EndPaint (hwnd, &ps) ;
            return 0 ;

       case   WM_DESTROY:
            PostQuitMessage (0) ;
            return 0 ;
       }
       return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```



5-12 LINEDEMO

RenaultPierm

PostScriptPostScriptTrueType

BEZIER5-4

```
5-4  BEZIER

BEZIER.C

/*------------------------------------------------------------------
```

```
   BEZIER.C -- Bezier Splines Demo

                    (c) Charles Petzold, 1998

----------------------------------------------------------------------*/

#include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns

                    PSTR szCmdLine, int iCmdShow)

{

   static TCHAR szAppName[] = TEXT ("Bezier") ;

   HWND         hwnd ;

   MSG          msg ;

   WNDCLASS     wndclass ;


   wndclass.style       = CS_HREDRAW | CS_VREDRAW ;

   wndclass.lpfnWndProc= WndProc ;

   wndclass.cbClsExtra   = 0 ;

   wndclass.cbWndExtra   = 0 ;

   wndclass.hInstance    = hInstance ;

   wndclass.hIcon       = LoadIcon (NULL, IDI_APPLICATION) ;
```

```
wndclass.hCursor      = LoadCursor (NULL, IDC_ARROW) ;
wndclass.hbrBackground= (HBRUSH) GetStockObject (WHITE_
wndclass.lpszMenuName= NULL ;
wndclass.lpszClassName= szAppName ;

if (!RegisterClass (&wndclass))
{
      MessageBox (NULL, TEXT ("Program requires Windows NT
                  szAppName, MB_ICONERROR) ;
           return 0 ;
}

hwnd = CreateWindow (szAppName, TEXT ("Bezier Splines"),
                        WS_OVERLAPPEDWINDOW,
                  CW_USEDEFAULT, CW_USEDEFAULT,
                  CW_USEDEFAULT, CW_USEDEFAULT,
                  NULL, NULL, hInstance, NULL) ;

ShowWindow (hwnd, iCmdShow) ;
```

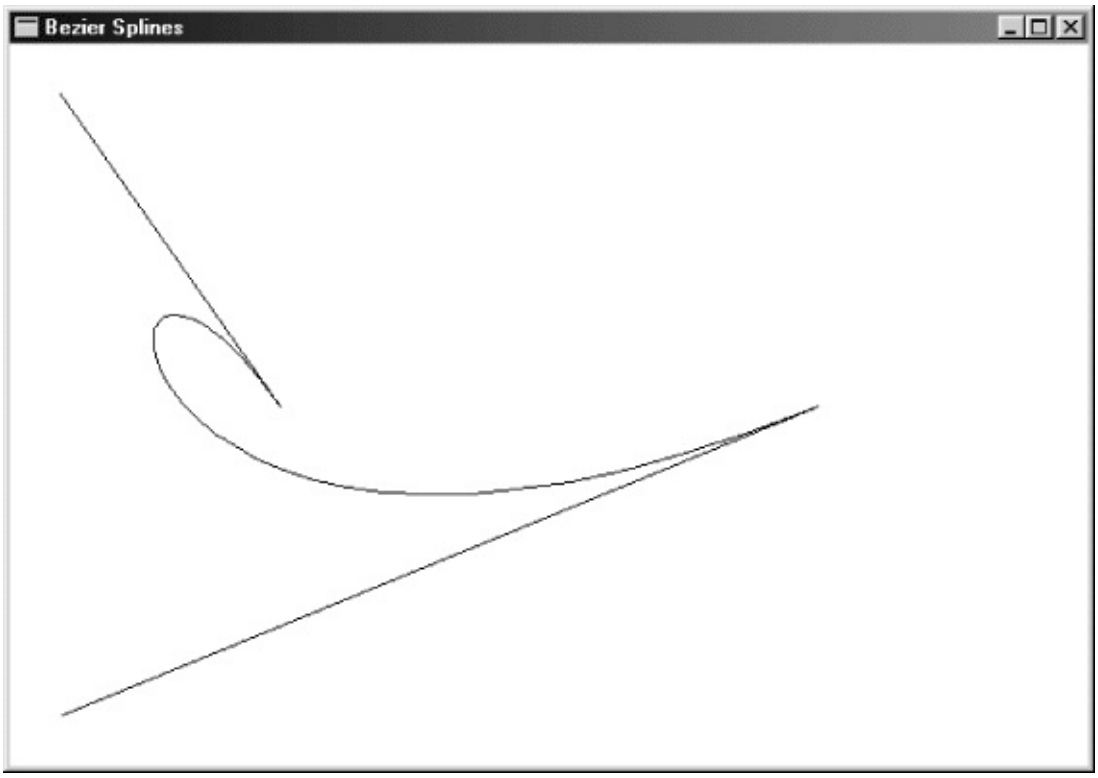
```
    UpdateWindow (hwnd) ;

    while (GetMessage (&msg, NULL, 0, 0))
    {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
    }
    return msg.wParam ;
}

void DrawBezier (HDC hdc, POINT apt[])
{
    PolyBezier (hdc, apt, 4) ;
    MoveToEx (hdc, apt[0].x, apt[0].y, NULL) ;
    LineTo   (hdc, apt[1].x, apt[1].y) ;

    MoveToEx (hdc, apt[2].x, apt[2].y, NULL) ;
    LineTo   (hdc, apt[3].x, apt[3].y) ;
}
```

```
LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
   static POINT apt[4] ;
   HDC          hdc ;
   int          cxClient, cyClient ;
   PAINTSTRUCT  ps ;

   switch (message)
   {
   case WM_SIZE:
         cxClient = LOWORD (lParam) ;
         cyClient = HIWORD (lParam) ;


               apt[0].x = cxClient / 4 ;
               apt[0].y = cyClient / 2 ;


               apt[1].x = cxClient / 2 ;
               apt[1].y = cyClient / 4 ;
```

```
                apt[2].x = cxClient / 2 ;
                apt[2].y = 3 * cyClient / 4 ;

                apt[3].x = 3 * cxClient / 4 ;
                apt[3].y = cyClient / 2 ;

                return 0 ;

        case WM_LBUTTONDOWN:
        case WM_RBUTTONDOWN:
        case WM_MOUSEMOVE:
             if (wParam & MK_LBUTTON || wParam & MK_RBUTTON)
             {
                hdc = GetDC (hwnd) ;
                SelectObject (hdc, GetStockObject (WHITE_PEN)) ;
                DrawBezier (hdc, apt) ;

                if (wParam & MK_LBUTTON)
                {
```

```
                apt[1].x = LOWORD (lParam) ;
                apt[1].y = HIWORD (lParam) ;
            }

        if (wParam & MK_RBUTTON)
            {
                apt[2].x = LOWORD (lParam) ;
                apt[2].y = HIWORD (lParam) ;
            }

            SelectObject (hdc, GetStockObject (BLACK_PEN)) ;
            DrawBezier (hdc, apt) ;
            ReleaseDC (hwnd, hdc) ;
       }
            return 0 ;
case   WM_PAINT:
       InvalidateRect (hwnd, NULL, TRUE) ;
```

```
            hdc = BeginPaint (hwnd, &ps) ;

            DrawBezier (hdc, apt) ;

            EndPaint (hwnd, &ps) ;
            return 0 ;

    case WM_DESTROY:
            PostQuitMessage (0) ;
            return 0 ;
    }
    return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```



5-13 BEZIER

32WindowsPolyline(

```
x(t) = (1 - t)3 x0 + 3t (1 - t)2 x1 + 3t2 (1 - t) x2 + t3 x3

y(t) = (1 - t)3 y0 + 3t (1 - t)2 y1 + 3t2 (1 - t) y2 + t3 y3
```

Windows 98

```
PolyBezier (hdc, apt, iCount) ;
```

```
PolyBezierTo (hdc, apt, iCount) ;
```

aptPOINTPolyBezieriCount1

PolyBezierTo

Stock Pens

WindowsBLACK_PENBLACK_PEN
WindowsWHITE_PENNULL_PENNULL_PEN

Windows WindowsWINDEF.HHPENhPen

```
HPEN hPen ;
```

GetStockObjectWHITE_PEN

```
hPen = GetStockObject (WHITE_PEN) ;
```

```
SelectObject (hdc, hPen) ;
```

WHITE_PEN

hPenGetStockObjectSelectObject

```
SelectObject (hdc, GetStockObject (WHITE_PEN)) ;
```

```

```

BLACK_PEN

```
SelectObject (hdc, GetStockObject (BLACK_PEN)) ;
```

SelectObject

```
hPen = SelectObject (hdc, GetStockobject (WHITE_PEN)) ;
```

WHITE_PENhPenBLACK_PEN

```
SelectObject (hdc, hPen) ;
```

BLACK_PEN

CreatePenCreatePenIndirectSelectObject
DeleteObject

GDIGDISelectObject

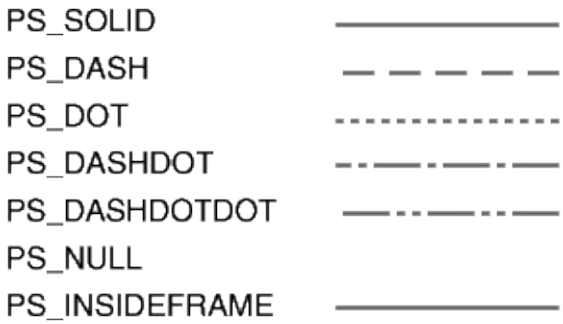
GDI

- GDI
- GDI
-

CreatePen

```
hPen = CreatePen (iPenStyle, iWidth, crColor) ;
```

iPenStyleWINGDI.H5-14



5-14

PS_SOLIDPS_NULLPS_INSIDEFRAMEiWidthiWidth01
Windows

CreatePencrColorCOLORREFPS_INSIDEFRAMEWindows
PS_INSIDEFRAME1

PS_INSIDEFRAMEPS_INSIDEFRAME1
PS_INSIDEFRAME

LOGPENCreatePenIndirect

CreatePenIndirectLOGPEN

```
LOGPEN logpen ;
```

lopnStyleUINTlopnWidthPOINTlopnColor
(COLORREF)WindowslopnWidthxy

CreatePenIndirect

```
hPen = CreatePenIndirect (&logpen) ;
```

CreatePenCreatePenIndirectSelectObject

13

```
static HPEN hPen1, hPen2, hPen3 ;
```

WM_CREATE

```
hPen1 = CreatePen (PS_SOLID, 1, 0) ;

hPen2 = CreatePen (PS_SOLID, 3, RGB (255, 0, 0)) ;

hPen3 = CreatePen (PS_DOT, 0, 0) ;
```

WM_PAINT

```
SelectObject (hdc, hPen2) ;
```

```
SelectObject (hdc, hPen1) ;
```

WM_DESTROY

```
DeleteObject (hPen1) ;

DeleteObject (hPen2) ;

DeleteObject (hPen3) ;
```

WM_PAINTEndPaintEndPaint

CreatePenSelectObject

```
SelectObject (hdc, CreatePen (PS_DASH, 0, RGB (255, 0, 0))) ;
```

SelectObjectSelectObject
BLACK_PENSelectObject

```
DeleteObject (SelectObject (hdc, GetStockObject (BLACK_PEN)))
```

SelectObject

```
hPen = SelectObject (hdc, CreatePen (PS_DASH, 0, RGB (255, 0,
```

hPenSelectObjecthPenBLACK_PENhPenSelectObject

```
DeleteObject (SelectObject (hdc, hPen)) ;
```

GetObjectLOGPEN

```
GetObject (hPen, sizeof (LOGPEN), (LPVOID) &logpen) ;
```

```
hPen = GetCurrentObject (hdc, OBJ_PEN) ;
```

ExtCreatePen

OPAQUEWindowsWHITE_BRUSH

Windows

```
SetBkColor (hdc, crColor) ;
```

crColorWindowsGetBkColor

TRANSPARENTWindows

```
SetBkMode (hdc, TRANSPARENT) ;
```

WindowsGetBkModeTRANSPARENTOPAQUE

WindowsROPROP2Windows16
ROP2WindowsR2_COPYPENWindows15ROP2

16ROP201

ROP2R2_BLACKWindows
R2_NOTMERGEPENWindows

5-216ROP2(P)(D)C

(P):(D):	**1 1**	**1 0**	**0 1**	**0 0**		
	0	0	0	0	0	R2_BLACK
	0	0	0	1	~(P | D)	R2_NOTMERGEPEN
	0	0	1	0	~P & D	R2_MASKNOTPEN
	0	0	1	1	~P	R2_NOTCOPYPEN
	0	1	0	0	P & ~D	R2_MASKPENNOT
	0	1	0	1	~D	R2_NOT
	0	1	1	0	P ^ D	R2_XORPEN
	0	1	1	1	~(P & D)	R2_NOTMASKPEN
	1	0	0	0	P & D	R2_MASKPEN
	1	0	0	1	~(P ^ D)	R2_NOTXORPEN

	1	0	1	0	D	R2_NOP
	1	0	1	1	~P | D	R2_MERGENOTPEN
	1	1	0	0	P	R2_COPYPEN
	1	1	0	1	P | ~D	R2_MERGEPENNOT
	1	1	1	0	P | D	R2_MERGEPEN
	1	1	1	1	1	R2_WHITE

```
SetROP2 (hdc, iDrawMode) ;
```

iDrawMode

```
iDrawMode = GetROP2 (hdc) ;
```

R2_COPYPENR2_NOTCOPYPENR2_BLACK
R2_WHITER2_NOP

Windows16ROP2R2_NOTR2_NOT
BLOKOUT R2_NOT

Windows5-3

Rectangle	
Ellipse	
RoundRect	
Chord	
Pie	
Polygon	
PolyPolygon	

WindowsWindows

WindowsWHITE_BRUSHLTGRAY_BRUSH
GRAY_BRUSHDKGRAY_BRUSHBLACK_BRUSHNULL_BRUSH
HOLLOW_BRUSHWindbwsHBRUSH

```
HBRUSH hBrush ;
```

GetStockObjectGRAY_BRUSH

```
hBrush = GetStockObject (GRAY_BRUSH) ;
```

SelectObject

```
SelectObject (hdc, hBrush) ;
```

NULL_PEN

```
SelectObject (hdc, GetStockObject (NULL_PEN)) ;
```

NULL_BRUSH

```
SelectObject (hdc, GetStockobject (NULL_BRUSH) ;
```

Polygon

PolygonPolyline

```
Polygon (hdc, apt, iCount) ;
```

aptPOINTiCountWindowsPolylineWindows
PolyPolygon

```
PolyPolygon (hdc, apt, aiCounts, iPolyCount) ;
```

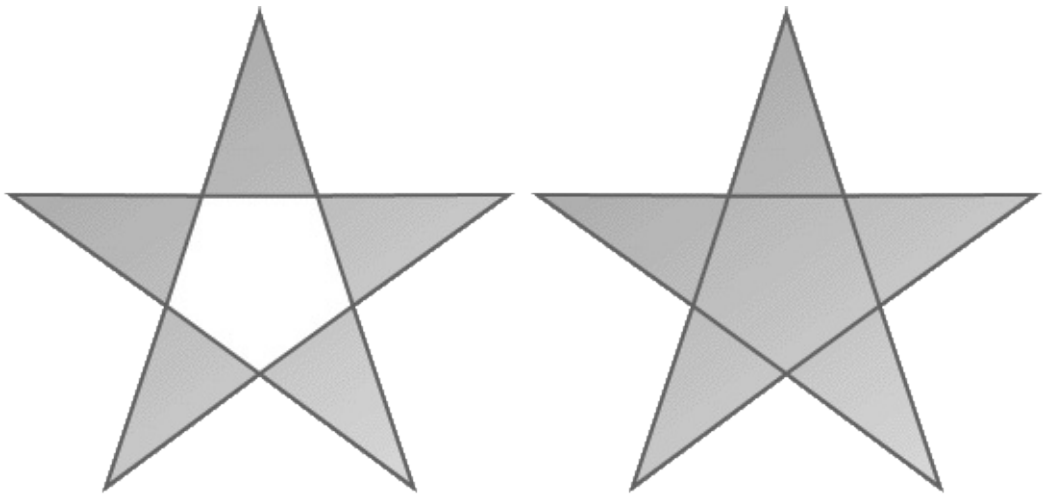
aiCountsaptPolyPolygon

```
for (i = 0, iAccum = 0 ; i < iPolyCount ; i++)
{
   Polygon (hdc, apt + iAccum, aiCounts[i]) ;
   iAccum += aiCounts[i] ;
}
```

PolygonPolyPolygonWindowsSetPolyFillMode

```
SetPolyFillMode (hdc, iMode) ;
```

ALTERNATEWINDING5-15



5-15　ALTERNATEWINDING

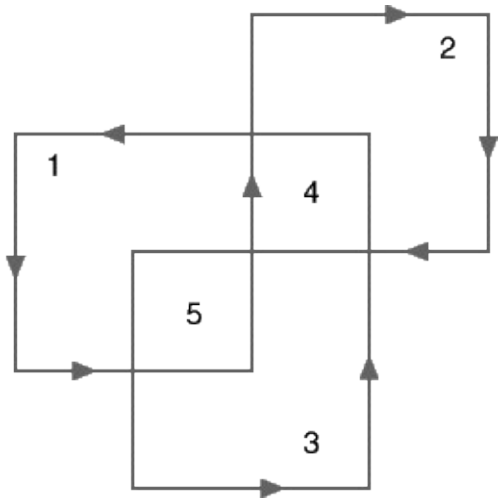
ALTERNATEWINDINGALTERNATE

WINDINGWINDING

WINDINGALTERNATE

5-16L1345ALTERNATEWINDING54

Windows5-5 ALTWIND



5-16 WINDING

5-5 ALTWIND

ALTWIND.C

```
/*------------------------------------------------------------

   ALTWIND.C --  Alternate and Winding Fill Modes

                 (c) Charles Petzold, 1998
```

```
------------------------------------------------------------------*/

#include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                    PSTR szCmdLine, int iCmdShow)

{
   static TCHAR szAppName[] = TEXT ("AltWind") ;
   HWND         hwnd ;
   MSG          msg ;
   WNDCLASS     wndclass ;
   wndclass.style       = CS_HREDRAW | CS_VREDRAW ;
   wndclass.lpfnWndProc= WndProc ;
   wndclass.cbClsExtra   = 0 ;
   wndclass.cbWndExtra   = 0 ;
   wndclass.hInstance    = hInstance ;
   wndclass.hIcon        = LoadIcon (NULL, IDI_APPLICATION) ;
   wndclass.hCursor      = LoadCursor (NULL, IDC_ARROW) ;
   wndclass.hbrBackground= (HBRUSH) GetStockObject (WHITE_
   wndclass.lpszMenuName= NULL ;
```

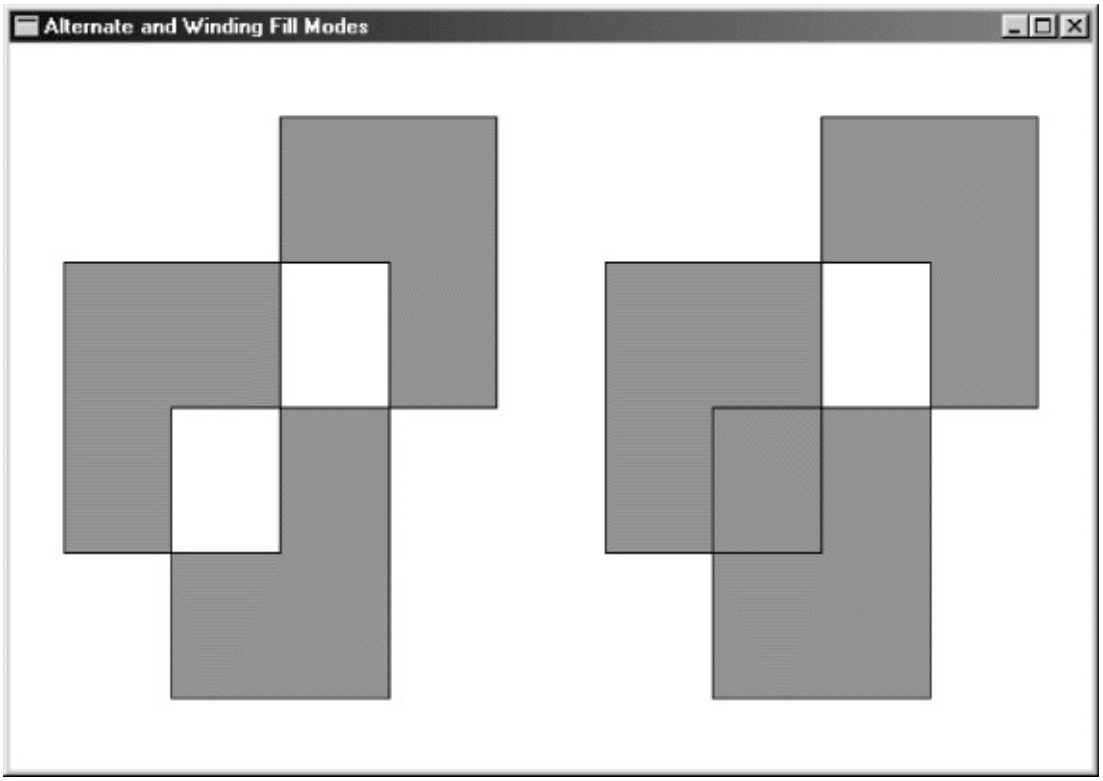
```
wndclass.lpszClassName= szAppName ;


if (!RegisterClass (&wndclass))
{
MessageBox (  NULL, TEXT ("Program requires Windows NT!"),
                    szAppName, MB_ICONERROR) ;
       return 0 ;
}


hwnd = CreateWindow (szAppName, TEXT ("Alternate and Wi
                    WS_OVERLAPPEDWINDOW,
                    CW_USEDEFAULT, CW_USEDEFAULT,
                    CW_USEDEFAULT, CW_USEDEFAULT,
                    NULL, NULL, hInstance, NULL) ;


       ShowWindow (hwnd, iCmdShow) ;
       UpdateWindow (hwnd) ;
```

```
   while (GetMessage (&msg, NULL, 0, 0))
   {
         TranslateMessage (&msg) ;
         DispatchMessage (&msg) ;
   }
         return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
   static POINT aptFigure [10] = {10,70, 50,70, 50,10, 90,10, 90,
                        30,50, 30,90, 70,90, 70,30, 10,30 };
   static int    cxClient, cyClient ;
   HDC           hdc ;
   int           i ;
   PAINTSTRUCT   ps ;
   POINT         apt[10] ;

         switch (message)
```

```
{
case   WM_SIZE:
       cxClient = LOWORD (lParam) ;
      cyClient = HIWORD (lParam) ;
      return 0 ;

case   WM_PAINT:
       hdc = BeginPaint (hwnd, &ps) ;

       SelectObject (hdc, GetStockObject (GRAY_BRUSH)) ;

       for (i = 0 ; i < 10 ; i++)
       {
            apt[i].x = cxClient * aptFigure[i].x / 200 ;
            apt[i].y = cyClient * aptFigure[i].y / 100 ;
       }

       SetPolyFillMode (hdc, ALTERNATE) ;
       Polygon (hdc, apt, 10) ;

       for (i = 0 ; i < 10 ; i++)
```

```
        {
             apt[i].x += cxClient / 2 ;
        }

        SetPolyFillMode (hdc, WINDING) ;
        Polygon (hdc, apt, 10) ;


        EndPaint (hwnd, &ps) ;
        return 0 ;

   case   WM_DESTROY:
        PostQuitMessage (0) ;
        return 0 ;
   }
   return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

100×100aptFigureALTERNATEWINDING5-17



5-17 ALTWIND

RectangleRoundRectEllipseChordPiePolygonPolyPolygon
8×8

WindowsWindows64Windows648×8011
8×8116256

WindowsSelectObjectGDI

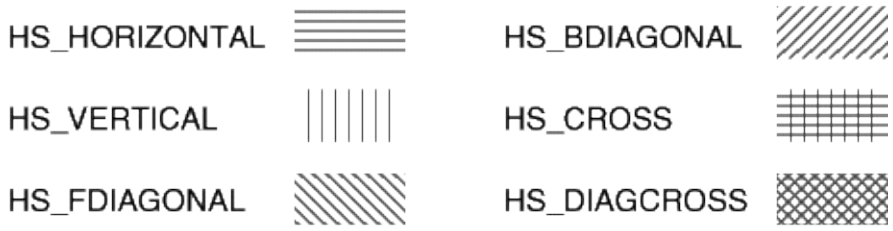
```
hBrush = CreateSolidBrush (crColor) ;
```

SolidWindows

(hatch marks)

```
hBrush = CreateHatchBrush (iHatchStyle, crColor) ;
```

iHatchStyle5-18



5-18

CreateHatchBrushcrColorWindowsOPAQUE
TRANSPARENTWindows

CreatePatternBrushCreateDIBPatternBrushPt

```
hBrush = CreateBrushIndirect (&logbrush) ;
```

logbrushLOGBRUSH5-4lbStyleWindows

5-4

lbStyle (UINT)	lbColor (COLORREF)	lbHatch (LONG)
BS_SOLID		
BS_HOLLOW		
BS_HATCHED		
BS_PATTERN		
BS_DIBPATTERNPT		DIB

SelectObjectDeleteObjectGetObjectSelectObject

```
SelectObject (hdc, hBrush) ;
```

DeleteObject

```
DeleteObject (hBrush) ;
```

GetObject

```
GetObject (hBrush, sizeof (LOGBRUSH), (LPVOID) &logbrush) ;
```

logbrushLOGBRUSH

GDI

GDITextOut

```
TextOut (hdc, x, y, psText, iLength) ;
```

xyxy(x,y)

TextOutGDIWindowsxy(orientation)
xy

Windows8WINGDI.H5-5

5-5

		x	**y**
MM_TEXT			
MM_LOMETRIC	0.1 mm		
MM_HIMETRIC	0.01 mm		
MM_LOENGLISH	0.01 in.		
MM_HIENGLISH	0.001 in.		

MM_TWIPS	1/1440 in.		
MM_ISOTROPIC	(x = y)		
MM_ANISOTROPIC	(x != y)		

METRICENGLISH1/721/72Twip1/201/1440
Isotropicanisotropic

```
SetMapMode (hdc, iMapMode) ;
```

iMapMode8

```
iMapMode = GetMapMode (hdc) ;
```

MM_TEXTTextOut

```
TextOut (hdc, 8, 16, TEXT ("Hello"), 5) ;
```

816

MM_LOENGLISH

```
SetMapMode (hdc, MM_LOENGLISH) ;
```

TextOut

```
TextOut (hdc, 50, -100, TEXT ("Hello"), 5) ;
```

0.51y

MM_TEXTGetDeviceCaps

GDI32Windows NT32Windows Windows32,767

MM_LOENGLISHWM_SIZEWindowsWM_MOVE WM_SIZEWM_MOUSEMOVEGDIGDIGDI GetSystemMetricsGDIGetDeviceCapsGDIWindows HORZRESVERTRES

GetTextMetricsTEXTMETRICMM_LOENGLISHGetTextMetrics GetTextMetrics

WindowsGDIWindowsWindowsxy

(0,0)WM_MOVEWindowsCreateWindow MoveWindowGetMessagePosGetCursorPosSetCursorPos GetWindowRectWindowFromPointDISPLAY CreateDCGDI

0,0WindowsGetWindowDCGDI

0,0GetDCBeginPaintGDI

ClientToScreenScreenToClientGetWindowRect

WindowsGDI

GetWindowDCCreateDC(0,0)
xy

GDI

Windows

$$xViewport = (xWindow - xWinOrg) \times \frac{xViewExt}{xWinExt} + xViewOrg$$

$$yViewport = (yWindow - yWinOrg) \times \frac{yViewExt}{yWinExt} + yViewOrg$$

(xWindow,yWindow)(xViewport,yViewport)

(xWinOrgyWinOrg)(xViewOrg,yViewOrg)(0,0)
(xWinOrg,yWinOrg)(xViewOrg,yViewOrg)(0,0)

$$xViewport = xWindow \times \frac{xViewExt}{xWinExt}$$

$$yViewport = yWindow \times \frac{yViewExt}{yWinExt}$$

(xWinExt,yWinExt)(xViewExt,yViewExt)

MM_LOENGLISHWindowsxViewExtxWinExtxViewExt

xy

Windows

$$xWindow = (xViewport - xViewOrg) \times \frac{xWinExt}{xViewExt} + xWinOrg$$

$$yWindow = (yViewport - yViewOrg) \times \frac{yWinExt}{yViewExt} + yWinOrg$$

Windows

```
DPtoLP (hdc, pPoints, iNumber) ;
```

pPointsPOINTiNumberGetClientRect

```
GetClientRect (hwnd, &rect) ;

DPtoLP (hdc, (PPOINT) &rect, 2) ;
```

```
LPtoDP (hdc, pPoints, iNumber) ;
```

MM_TEXT

MM_TEXT

(0, 0)

(0, 0)

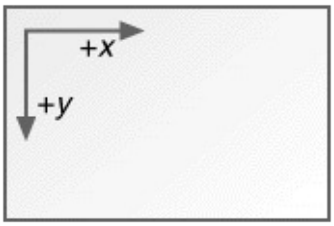
(1, 1)

(1, 1)

1

$$xViewport = xWindow - xWinOrg + xViewOrg$$
$$yViewport = yWindow - yWinOrg + yViewOrg$$

MM_TEXT



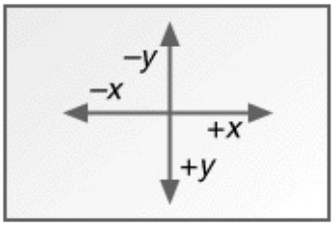
WindowsSetViewportOrgExSetWindowOrgEx(0,0)
SetViewportOrgExSetWindowOrgEx

(xViewOrg,yViewOrg)(0.0)(xViewOrg,yViewOrg)
(xWinOrg,yWinOrg)(xWinOrg,yWinOrg)(0,0)(0,0)

cxClientcyClient(00)

```
SetViewportOrgEx (hdc, cxClient / 2, cyClient / 2, NULL) ;
```

SetViewportOrgEx(0,0)(cxClient/2,cyClient/2)



x-cxClient/2+cxClient/2y-cyClient/2+cyClient/2
(cxClient/2,cyClient/2)

```
TextOut (hdc, -cxClient / 2, -cyClient / 2, "Hello", 5) ;
```

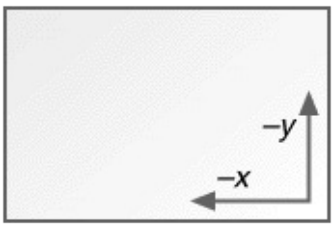
SetWindowOrgExSetViewportOrgEx

```
SetWindowOrgEx (hdc, -cxClient / 2, -cyClient / 2, NULL) ;
```

SetWindowOrgEx(-cxClient / 2,-cyC

```
SetViewportOrgEx (hdc, cxClient / 2, cyClient / 2, NULL) ;

SetWindowOrgEx (hdc, -cxClient / 2, -cyClient / 2, NULL) ;
```

(-cxClient/2,-cyClient/2)(cxClient/2, cyClient/2)



```
GetViewportOrgEx (hdc, &pt) ;

GetWindowOrgEx (hdc, &pt) ;
```

ptPOINTGetViewportOrgExGetWindowOrgEx

SYSMETS2iVscrollPosy

```
case       WM_PAINT:
   hdc = BeginPaint (hwnd, &ps) ;


   for (i = 0 ; i < NUMLINES ; i++)
   {
         y = cyChar * (i - iVscrollPos) ;
         //
   }
   EndPaint (hwnd, &ps) ;
   return 0 ;
```

SetWindowOrgEx

```
case       WM_PAINT:
   hdc = BeginPaint (hwnd, &ps) ;
```

```
SetWindowOrgEx (hdc, 0, cyChar * iVscrollPos) ;


for (i = 0 ; i < NUMLINES ; i++)
{
      y = cyChar * i ;
      //
}
EndPaint (hwnd, &ps) ;
return 0 ;
```

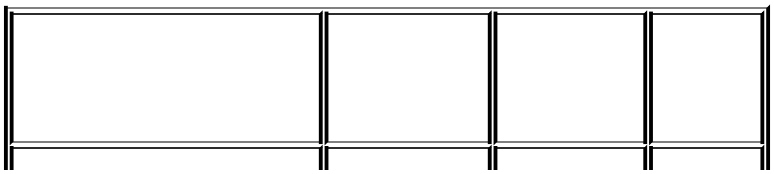
TextOutyiVscrollPosiVscrollPos

0,0MM_TEXTyMM_TEXTy
MM_TEXT

Windowsxy

5-6

5-6



MM_LOENGLISH	0.01 in.	0.01	0.254
MM_LOMETRIC	0.1 mm.	0.00394	0.1
MM_HIENGLISH	0.001 in.	0.001	0.0254
MM_TWIPS	1/1400 in.	0.000694	0.0176
MM_HIMETRIC	0.01 mm.	0.000394	0.01

(0, 0)

(0, 0)

(1, 1)

(1, 1)

$$xViewport = (xWindow - xWinOrg) \times \frac{xViewExt}{xWinExt} + xViewOrg$$

$$yViewport = (yWindow - yWinOrg) \times \frac{yViewExt}{yWinExt} + yViewOrg$$

MM_LOENGLISHWindows

$$\frac{xViewExt}{xWinExt} = 0.01\ in\ 中的水平圖素數$$

$$\frac{-yViewExt}{yWinExt} = 0.01\ in\ 中的垂直圖素數$$

WindowsGetDeviceCapsWindows 98Windo

Windows 9896 dpi
LOGPIXELSY96Windows5-7

5-7

	(x,y)	(x,y)
MM_LOMETRIC	(96, 96)	(254, -254)
MM_HIMETRIC	(96, 96)	(2540, -2540)
MM_LOENGLISH	(96, 96)	(100, -100)
MM_HIENGLISH	(96, 96)	(1000, -1000)
MM_TWIPS	(96, 96)	(1440, -1440)

MM_LOENGLISH961000.01MM_LOMETRIC962540.1

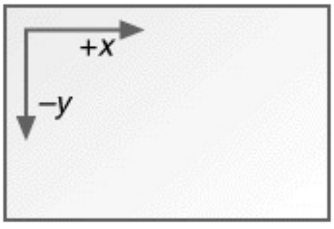
Windows NT16WindowsHORZRESVERTRESGetDeviceCaps
HORZSIZEVERTSIZEGetDeviceCaps3202401024×7685-8
Windows NT

5-8

	(x,y)	**(x,y)**
MM_LOMETRIC	(1024, -768)	(3,200, 2,400)
MM_HIMETRIC	(1024, -768)	(32,000, 24,000)
MM_LOENGLISH	(1024, -768)	(1,260, 945)
MM_HIENGLISH	(1024, -768)	(12,598, 9,449)
MM_TWIPS	(1024, -768)	(18,142, 13,606)

3201260

yy(0,0)

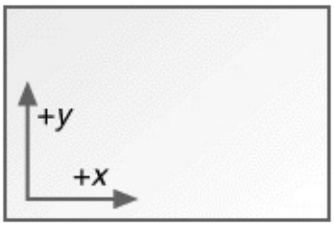


y

```
SetMapMode (hdc, MM_LOENGLISH) ;

TextOut (hdc, 100, -100, "Hello", 5) ;
```

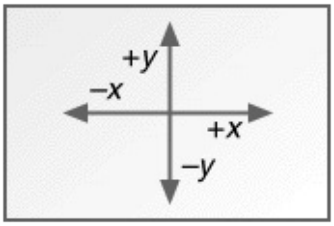
(0,0)SetViewportOrgExcyClient

```
SetViewportOrgEx (hdc, 0, cyClient, NULL) ;
```



(0,0)

```
SetViewportOrgEx (hdc, cxClient / 2, cyClient / 2, NULL) ;
```



4xytwip

SetWindowOrgEx(0,0)SetWindowOrgEx(cxClient,cyClient)DPtoLP ptPOINT(0,0)

```
pt.x = cxClient ;

pt.y = cyClient ;

DptoLP (hdc, &pt, 1) ;

SetWindowOrgEx (hdc, -pt.x / 2, -pt.y / 2, NULL) ;
```

MM_ISOTROPICMM_ANISOTROPICWindowsisotropic anisotropicMM_ISOTROPICxy

MM_ISOTROPICMM_ISOTROPIC

Windows

MM_TEXTWindowsMM_ISOTROPICWindows xyMM_ANISOTROPICWindows

MM_ISOTROPIC

MM_ISOTROPIC

MM_ISOTROPICWindowsMM_LOMETRIC SetWindowExtExSetViewportExtExWindows

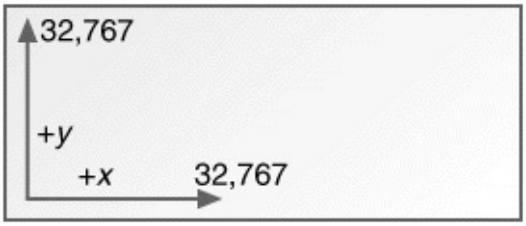
SetWindowExtExSetViewportExtExWindows
SetViewportExtExSetWindowExtEx

(0,0)032,767xy

```
SetMapMode (hdc, MM_ISOTROPIC) ;

SetWindowExtEx (hdc, 32767, 32767, NULL) ;

SetViewportExtEx (hdc, cxClient, -cyClient, NULL) ;

SetViewportOrgEx (hdc, 0, cyClient, NULL) ;
```

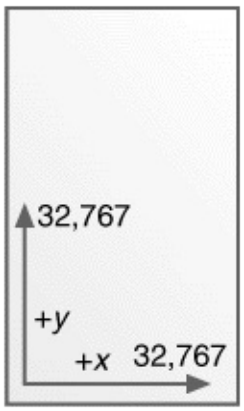
GetWindowExtExGetViewportExtExWindows

Windowsx



Windows 98x16Windows

Windowsy



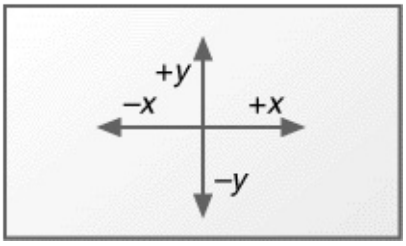
Windows 98

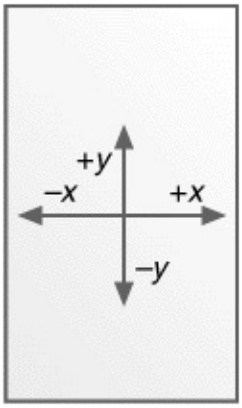
```
SetMapMode (MM_ISOTROPIC) ;
SetWindowExtEx (hdc, 32767, 32767, NULL) ;
SetViewportExtEx (hdc, cxClient, -cyClient, NULL) ;
SetWindowOrgEx (hdc, 0, 32767, NULL) ;
```

SetWindowOrgEx(0, 32767)(0,0)

(0,0)

```
SetMapMode (hdc, MM_ISOTROPIC) ;
SetWindowExtEx (hdc, 1000, 1000, NULL) ;
SetViewportExtEx (hdc, cxClient / 2, -cyClient / 2, NULL) ;
SetViewportOrgEx (hdc, cxClient / 2, cyClient / 2, NULL) ;
```





GDI-10001000xy

MM_ISOTROPIC(0,0)yMM_TEXT1/16

```
SetMapMode (hdc, MM_ISOTROPIC) ;
SetWindowExtEx (hdc, 16, 16, NULL) ;
SetViewportExtEx (hdc, GetDeviceCaps (hdc, LOGPIXELSX),
                  GetDeviceCaps (hdc, LOGPIXELSY), NULL) ;
```

SetWindowExtExSetViewportExtEx

Windows NT

```
SetMapMode (hdc, MM_ISOTROPIC) ;
SetWindowExtEx (hdc, 160 * GetDeviceCaps (hdc, HORZSIZE) / 2
        160 * GetDeviceCaps (hdc, VERTSIZE) / 254, NULL) ;
SetViewportExtEx (hdc, GetDeviceCaps (hdc, HORZRES),
          GetDeviceCaps (hdc, VERTRES), NULL) ;
```

1/16GetDeviceCapsHORZRESVERTRES25.4
16l/16160254

1/16l/32

MM_ANISOTROPIC

MM_ISOTROPICWindowsMM_ANISOTROPICWindows
MM_ANISOTROPIC

MM_ANISOTROPICMM_ISOTROPIC(0,0)xy032,767

```
SetMapMode (hdc, MM_ANISOTROPIC) ;
SetWindowExtEx (hdc, 32767, 32767, NULL) ;
SetViewportExtEx (hdc, cxClient, -cyClient, NULL) ;
SetViewportOrgEx (hdc, 0, cyClient, NULL) ;
```

MM_ISOTROPICMM_ANISOTROPIC(32767,

MM_ISOTROPICxy-1000+1000MM_ANISOTROPIC

```
SetMapMode (hdc, MM_ANISOTROPIC) ;

SetWindowExtEx (hdc, 1000, 1000, NULL) ;

SetViewportExtEx (hdc, cxClient / 2, -cyClient / 2, NULL) ;

SetViewportOrgEx (hdc, cxClient / 2, cyClient / 2, NULL) ;
```

MM_ANISOTROPIC

MM_ANISOTROPICxy

```
SetMapMode (hdc, MM_ANISOTROPIC) ;

SetWindowExtEx (hdc, 1, 1, NULL) ;

SetViewportExtEx (hdc, cxChar, cyChar, NULL) ;
```

cxCharcyChar

```
TextOut (hdc, 3, 2, TEXT ("Hello"), 5) ;
```

WHATSIZE

MM_ANISOTROPICMM_ANISOTROPIC
MM_LOENGLISH0.01yMM_TEXTy

```
SIZE size ;
```

```
SetMapMode (hdc, MM_LOENGLISH) ;
```

```
SetMapMode (hdc, MM_ANISOTROPIC) ;

GetViewportExtEx (hdc, &size) ;

SetViewportExtEx (hdc, size.cx, -size.cy, NULL) ;
```

MM_LOENGLISHMM_ANISOTROPICGetViewportExtExSIZE
SetViewportExtExy

WHATSIZE

WindowsWindowsMicrosoft System
WSZwhat sizeWHATSIZE5-6

```
5-6  WHATSIZE

WHATSIZE.C

/*------------------------------------------------------------

  WHATSIZE.C -- What Size is the Window?

          (c) Charles Petzold, 1998

------------------------------------------------------------*/

#include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns

           PSTR szCmdLine, int iCmdShow)

{
```

```
static TCHAR szAppName[] = TEXT ("WhatSize") ;
HWND         hwnd ;
MSG          msg ;
WNDCLASS     wndclass ;

wndclass.style        = CS_HREDRAW | CS_VREDRAW;
wndclass.lpfnWndProc= WndProc ;
wndclass.cbClsExtra   = 0 ;
wndclass.cbWndExtra   = 0 ;
wndclass.hInstance    = hInstance ;
wndclass.hIcon= LoadIcon (NULL, IDI_APPLICATION) ;
wndclass.hCursor= LoadCursor (NULL, IDC_ARROW) ;
wndclass.hbrBackground= (HBRUSH) GetStockObject (WHITE_
wndclass.lpszMenuName = NULL ;
wndclass.lpszClassName= szAppName ;
if (!RegisterClass (&wndclass))
{
      MessageBox (NULL, TEXT ("This program requires Windo
     szAppName, MB_ICONERROR) ;
```

```
        return 0 ;
    }

    hwnd = CreateWindow (szAppName, TEXT ("What Size is the W
                         WS_OVERLAPPEDWINDOW,
                        CW_USEDEFAULT, CW_USEDEFAULT,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         NULL, NULL, hInstance, NULL) ;

    ShowWindow (hwnd, iCmdShow) ;
    UpdateWindow (hwnd) ;
    while (GetMessage (&msg, NULL, 0, 0))
    {
        TranslateMessage (&msg) ;
        DispatchMessage (&msg) ;
    }
    return msg.wParam ;
}
```

```
void Show (HWND hwnd, HDC hdc, int xText, int yText, int iMapM
          TCHAR * szMapMode)
{
   TCHAR szBuffer [60] ;
   RECT  rect ;

   SaveDC (hdc) ;
   SetMapMode (hdc, iMapMode) ;
   GetClientRect (hwnd, &rect) ;
   DPtoLP (hdc, (PPOINT) &rect, 2) ;

   RestoreDC (hdc, -1) ;
   TextOut (     hdc, xText, yText, szBuffer,
               wsprintf (szBuffer, TEXT ("%-20s %7d %7d %7d %7d
                rect.left, rect.right, rect.top, rect.bottom)) ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
```

```
static TCHAR szHeading [] =
        TEXT ("Mapping Mode   Left   Right  Top  Bottom") ;
static TCHAR szUndLine [] =
        TEXT ("------------   ----   -----   ---  ------") ;
static int   cxChar, cyChar ;
HDC          hdc ;
PAINTSTRUCT   ps ;
TEXTMETRIC    tm ;

switch (message)
{
case   WM_CREATE:
        hdc = GetDC (hwnd) ;
        SelectObject (hdc, GetStockObject (SYSTEM_FIXED_FONT

        GetTextMetrics (hdc, &tm) ;
        cxChar = tm.tmAveCharWidth ;
        cyChar = tm.tmHeight + tm.tmExternalLeading ;
```

```
        ReleaseDC (hwnd, hdc) ;
        return 0 ;

case   WM_PAINT:
        hdc = BeginPaint (hwnd, &ps) ;
        SelectObject (hdc, GetStockObject (SYSTEM_FIXED_FONT
        SetMapMode (hdc, MM_ANISOTROPIC) ;
        SetWindowExtEx (hdc, 1, 1, NULL) ;
        SetViewportExtEx (hdc, cxChar, cyChar, NULL) ;

        TextOut (hdc, 1, 1, szHeading, lstrlen (szHeading)) ;
        TextOut (hdc, 1, 2, szUndLine, lstrlen (szUndLine)) ;

        Show (hwnd, hdc, 1, 3, MM_TEXT,      TEXT ("TEXT (pixels
        Show (hwnd, hdc, 1, 4, MM_LOMETRIC,  TEXT ("LOMETRIC
        Show (hwnd, hdc, 1, 5, MM_HIMETRIC,  TEXT ("HIMETRIC
        Show (hwnd, hdc, 1, 6, MM_LOENGLISH, TEXT ("LOENGLI
        Show (hwnd, hdc, 1, 7, MM_HIENGLISH,TEXT ("HIENGLISH
```

```
        Show (hwnd, hdc, 1, 8, MM_TWIPS,     EXT ("TWIPS (1/144


        EndPaint (hwnd, &ps) ;
        return 0 ;

   case   WM_DESTROY:
        PostQuitMessage (0) ;
        return 0 ;
   }
   return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

TextOutWHATSIZEWindows

```
SelectObject (hdc, GetStockObject (SYSTEM_FIXED_FONT)) ;
```

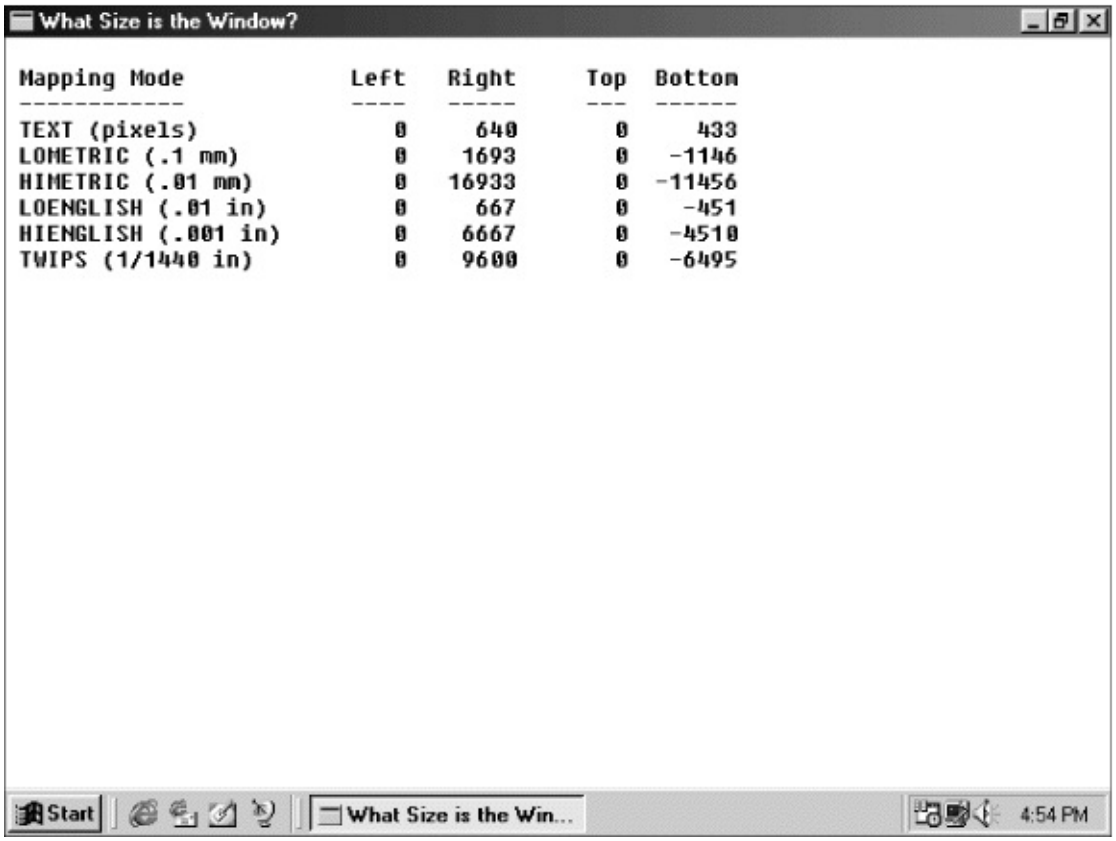
WHATSIZEMM_ANISTROPIC

WHATSIZEWHATSIZEShow

```
SaveDC (hdc) ;
SetMapMode (hdc, iMapMode) ;
GetClientRect (hwnd, &rect) ;
```

```
DptoLP (hdc, (PPOINT) &rect, 2) ;

RestoreDC (hdc, -1) ;
```

5-19WHATSIZE



5-19
WHATSIZE

WindowsRECT

```
FillRect (hdc, &rect, hBrush) ;

FrameRect (hdc, &rect, hBrush) ;

InvertRect (hdc, &rect) ;
```

rectRECT4lefttoprightbottom

FillRectrightbottom

FrameRectRectangleFrameRect2

InvertRect1001

Windows9RECTRECT

```
rect.left         = xLeft ;

rect.top          = xTop ;

rect.right        = xRight ;

rect.bottom       = xBottom ;
```

SetRect

```
SetRect (&rect, xLeft, yTop, xRight, yBottom) ;
```

8

- xy

OffsetRect (&rect, x, y) ;

-

InflateRect (&rect, x, y) ;

- 0

SetRectEmpty (&rect) ;

-

CopyRect (&DestRect, &SrcRect) ;

-

IntersectRect (&DestRect, &SrcRect1, &SrcRect2) ;

-

UnionRect (&DestRect, &SrcRect1, &SrcRect2) ;

-

bEmpty = IsRectEmpty (&rect) ;

-

bInRect = PtInRect (&rect, point) ;

CopyRect

```
DestRect = SrcRect ;
```

WindowsWM_PAINTwhile(TRUE)

WindowsWM_TIMER
WM_TIMER

WindowsWindowsPeekMessage
PeekMessage

```
PeekMessage (&msg, NULL, 0, 0, PM_REMOVE) ;
```

MSGGetMessageNULL0PeekMessage
PeekMessagePM_REMOVEPM_NOREMOVEPeek_Message

GetMessagePeekMessagePeekMessageTRUE0
PeekMessageFALSE0

```
while (GetMessage (&msg, NULL, 0, 0))
{
   TranslateMessage (&msg) ;
   DispatchMessage (&msg) ;
}
return msg.wParam ;
```

```
while (TRUE)
{
   if (PeekMessage (&msg, NULL, 0, 0, PM_REMOVE))
   {
         if (msg.message == WM_QUIT)
               break ;
         TranslateMessage (&msg) ;
         DispatchMessage (&msg) ;
   }
   else
   {
         //
   }
}
return msg.wParam ;
```

WM_QUITGetMessageWM_QUIT0PeekMessage
WM_QUIT

PeekMessageTRUEFALSEWindows

WindowsPeekMessageWM_PAINTGetMessageWM_PAINT
WM_PAINTValidateRectValidateRgnBeginPaintEndPaint
PeekMessageWM_PAINT

```
while (PeekMessage (&msg, NULL, 0, 0, PM_REMOVE)) ;
```

WM_PAINTWM_PAINTwhile

PeekMessageWindowsWindows 98Windows16 WindowsTerminal
PeekMessageWindowsPeekMessageWindows

PeekMessageRANDRECT5-7

5-7 RANDRECT

```
RANDRECT.C

/*------------------------------------------------------------------

  RANDRECT.C --   Displays Random Rectangles

                (c) Charles Petzold, 1998

------------------------------------------------------------------*/

#include <windows.h>

#include <stdlib.h>// for the rand function


LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

void DrawRectangle (HWND) ;
```

```
int cxClient, cyClient ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                    PSTR szCmdLine, int iCmdShow)
{
   static TCHAR szAppName[] = TEXT ("RandRect") ;
   HWND         hwnd ;
   MSG          msg ;
   WNDCLASS     wndclass ;

   wndclass.style       = CS_HREDRAW | CS_VREDRAW ;
   wndclass.lpfnWndProc= WndProc ;
   wndclass.cbClsExtra   = 0 ;
   wndclass.cbWndExtra   = 0 ;
   wndclass.hInstance    = hInstance ;
   wndclass.hIcon        = LoadIcon (NULL, IDI_APPLICATION) ;
   wndclass.hCursor      = LoadCursor (NULL, IDC_ARROW) ;
   wndclass.hbrBackground= (HBRUSH) GetStockObject (WHITE_
   wndclass.lpszMenuName= NULL ;
```

```
wndclass.lpszClassName= szAppName ;
if (!RegisterClass (&wndclass))
{
      MessageBox (NULL, TEXT ("This program requires Windo
                        szAppName, MB_ICONERROR) ;
      return 0 ;
}

hwnd = CreateWindow (szAppName, TEXT ("Random Rectang
                  WS_OVERLAPPEDWINDOW,
                  CW_USEDEFAULT, CW_USEDEFAULT,
                  CW_USEDEFAULT, CW_USEDEFAULT,
                  NULL, NULL, hInstance, NULL) ;

ShowWindow (hwnd, iCmdShow) ;
UpdateWindow (hwnd) ;

while (TRUE)
{
```

```
        if (PeekMessage (&msg, NULL, 0, 0, PM_REMOVE))
        {
             if (msg.message == WM_QUIT)
               break ;
             TranslateMessage (&msg) ;
             DispatchMessage (&msg) ;
      }
        else
             DrawRectangle (hwnd) ;
   }
   return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT iMsg, WPARAM
{
   switch (iMsg)
   {
   case   WM_SIZE:
        cxClient = LOWORD (lParam) ;
```

```
        cyClient = HIWORD (lParam) ;
        return 0 ;

   case   WM_DESTROY:
        PostQuitMessage (0) ;
        return 0 ;
   }
   return DefWindowProc (hwnd, iMsg, wParam, lParam) ;
}
void DrawRectangle (HWND hwnd)
{
   HBRUSHhBrush ;
   HDC          hdc ;
   RECT         rect ;

   if (cxClient == 0 || cyClient == 0)
        return ;
   SetRect (&rect, rand () % cxClient, rand () % cyClient,
```

```
                    rand () % cxClient, rand () % cyClient) ;
    hBrush = CreateSolidBrush (
                RGB (rand () % 256, rand () % 256, rand () % 256)) ;
    hdc = GetDC (hwnd) ;
    FillRect (hdc, &rect, hBrush) ;
    ReleaseDC (hwnd, hdc) ;
    DeleteObject (hBrush) ;
}
```

SetRectFillRectCrand

GDIDeleteObject

WindowsHRGN

```
hRgn = CreateRectRgn (xLeft, yTop, xRight, yBottom) ;
```

```
hRgn = CreateRectRgnIndirect (&rect) ;
```

```
hRgn = CreateEllipticRgn (xLeft, yTop, xRight, yBottom) ;
```

```
hRgn = CreateEllipticRgnIndirect (&rect) ;
```

CreateRoundRectRgn

Polygon

```
hRgn = CreatePolygonRgn (&point, iCount, iPolyFillMode) ;
```

pointPOINTiCountiPolyFillModeALTERNATEWINDING
CreatePolyPolygonRgn

```
iRgnType = CombineRgn (hDestRgn, hSrcRgn1, hSrcRgn2, iCom
```

hSrcRgn1hSrcRgn2hDestRgnhDestRgnhDestRgn

iCombinehSrcRgn1hSrcRgn25-9

5-9

iCombine	
RGN_AND	
RGN_OR	

RGN_XOR	
RGN_DIFF	hSrcRgn1hSrcRgn2
RGN_COPY	hSrcRgn1hSrcRgn2

CombineRgniRgnTypeNULLREGIONSIMPLEREGION
COMPLEXREGIONERROR

```
FillRgn (hdc, hRgn, hBrush) ;

FrameRgn (hdc, hRgn, hBrush, xFrame, yFrame) ;

InvertRgn (hdc, hRgn) ;

PaintRgn (hdc, hRgn) ;
```

FillRgnFrameRgnInvertRgnFillRectFrameRectInvertRectFrameRgn
xFrameyFramePaintRgn

GDI

```
DeleteObject (hRgn) ;
```

InvalidateRectWM_PAINTInvalidateRectWM_PAINT

```
InvalidateRect (hwnd, NULL, TRUE) ;
```

GetUpdateRectValidateRectWM_PAINTPAINTSTRUCT
BeginPaint

WindowsInvalidateRectValidateRect

```
InvalidateRgn (hwnd, hRgn, bErase) ;
```

```
ValidateRgn (hwnd, hRgn) ;
```

WM_PAINT

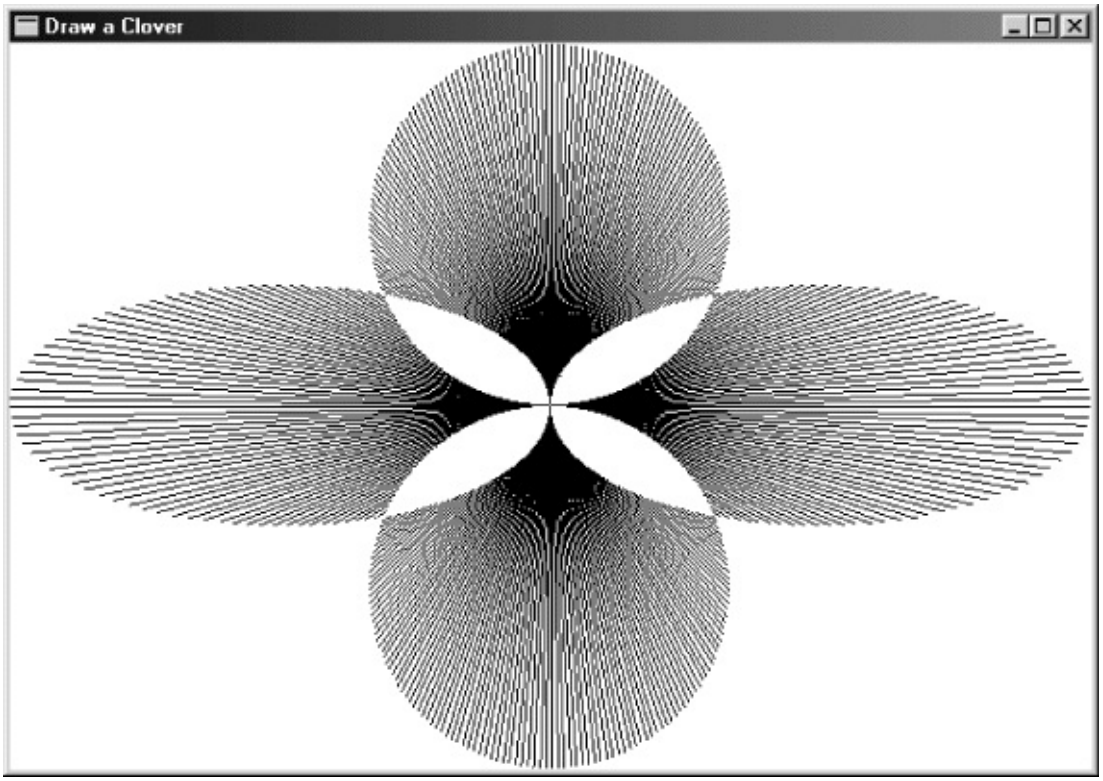
```
SelectObject (hdc, hRgn) ;
```

```
SelectClipRgn (hdc, hRgn) ;
```

GDIWindowsExcludeClipRectIntersectClipRect
OffsetClipRgn

CLOVER

CLOVER5-20



5-20　CLOVER

WindowsCLOVER5-8

```
5-8  CLOVER

CLOVER.C

/*------------------------------------------------------------------------

   CLOVER.C --   Clover Drawing Program Using Regions

                (c) Charles Petzold, 1998

------------------------------------------------------------------------*/

#include <windows.h>
```

```
#include <math.h>

#define TWO_PI (2.0 * 3.14159)

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                    PSTR szCmdLine, int iCmdShow)
{
   static TCHAR szAppName[] = TEXT ("Clover") ;
   HWND         hwnd ;
   MSG          msg ;
   WNDCLASS     wndclass ;


   wndclass.style       = CS_HREDRAW | CS_VREDRAW ;
   wndclass.lpfnWndProc= WndProc ;
   wndclass.cbClsExtra   = 0 ;
   wndclass.cbWndExtra   = 0 ;
   wndclass.hInstance    = hInstance ;
   wndclass.hIcon        = LoadIcon (NULL, IDI_APPLICATION) ;
   wndclass.hCursor      = LoadCursor (NULL, IDC_ARROW) ;
   wndclass.hbrBackground= (HBRUSH) GetStockObject (WHITE_
```

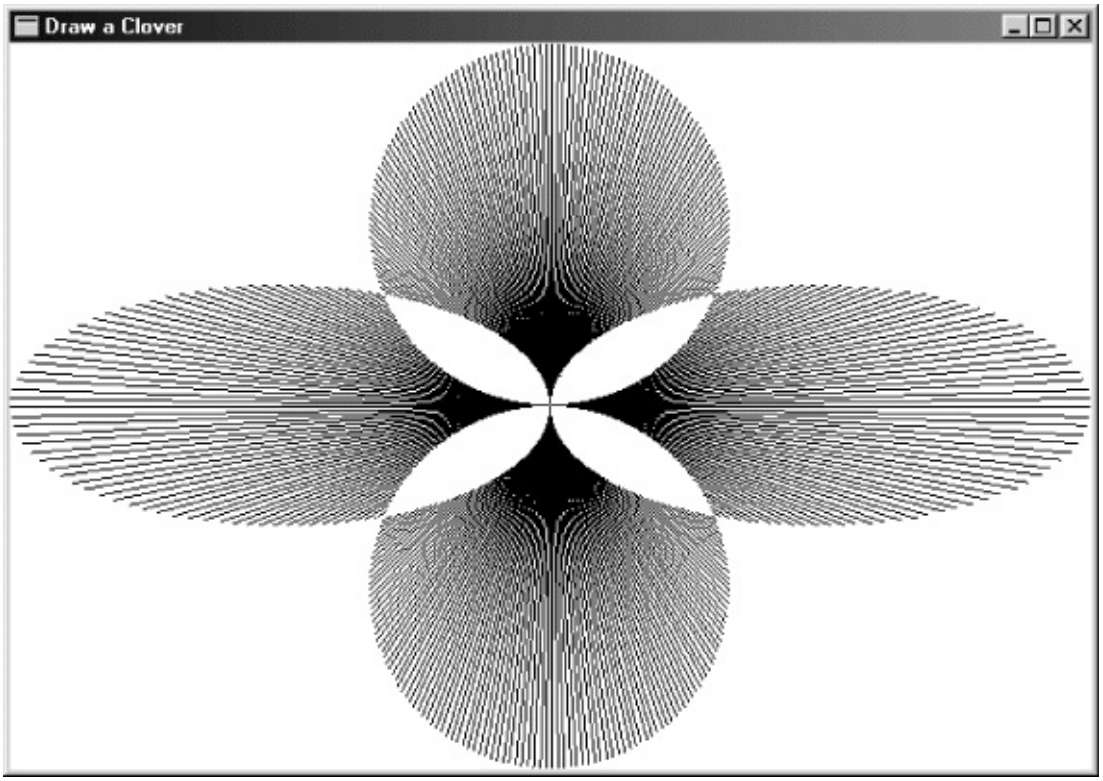
```
wndclass.lpszMenuName = NULL ;
wndclass.lpszClassName= szAppName ;

if (!RegisterClass (&wndclass))
{
      MessageBox (NULL, TEXT ("This program requires Windo
            szAppName, MB_ICONERROR) ;
      return 0 ;
}

hwnd = CreateWindow (szAppName, TEXT ("Draw a Clover"),
                  WS_OVERLAPPEDWINDOW,
                  CW_USEDEFAULT, CW_USEDEFAULT,
                  CW_USEDEFAULT, CW_USEDEFAULT,
                  NULL, NULL, hInstance, NULL) ;
ShowWindow (hwnd, iCmdShow) ;
UpdateWindow (hwnd) ;
```

```
   while (GetMessage (&msg, NULL, 0, 0))
   {
         TranslateMessage (&msg) ;
         DispatchMessage (&msg) ;
   }
   return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT iMsg, WPARAM
{
   static HRGN hRgnClip ;
   static int    cxClient, cyClient ;
   double fAngle, fRadius ;
   HCURSORhCursor ;
   HDC          hdc ;
   HRGN         hRgnTemp[6] ;
   int          i ;
   PAINTSTRUCT ps ;
   switch (iMsg)
```

```
   {
   case   WM_SIZE:
          cxClient      = LOWORD (lParam) ;
          cyClient      = HIWORD (lParam) ;
          hCursor= SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
          ShowCursor (TRUE) ;

          if (hRgnClip)
               DeleteObject (hRgnClip) ;

          hRgnTemp[0]   = CreateEllipticRgn (0, cyClient / 3,
                              cxClient / 2, 2 * cyClient / 3) ;
          hRgnTemp[1]   = CreateEllipticRgn (cxClient / 2, cyClient
                              cxClient, 2 * cyClient / 3) ;
          hRgnTemp[2]   = CreateEllipticRgn (cxClient / 3, 0,
                              2 * cxClient / 3, cyClient / 2) ;
          hRgnTemp[3]   = CreateEllipticRgn (cxClient / 3, cyClient
2 * cxClient / 3, cyClient) ;
          hRgnTemp[4]   = CreateRectRgn (0, 0, 1, 1) ;
```

```
        hRgnTemp[5]   = CreateRectRgn (0, 0, 1, 1) ;
        hRgnClip      = CreateRectRgn (0, 0, 1, 1) ;

        CombineRgn (hRgnTemp[4], hRgnTemp[0], hRgnTemp[1]
        CombineRgn (hRgnTemp[5], hRgnTemp[2], hRgnTemp[3]
        CombineRgn (hRgnClip,    hRgnTemp[4], hRgnTemp[5], R


        for (i = 0 ; i < 6 ; i++)
              DeleteObject (hRgnTemp[i]) ;

        SetCursor (hCursor) ;
        ShowCursor (FALSE) ;
        return 0 ;

case   WM_PAINT:
        hdc = BeginPaint (hwnd, &ps) ;

        SetViewportOrgEx (hdc, cxClient / 2, cyClient / 2, NULL) ;
     SelectClipRgn (hdc, hRgnClip) ;
```

```
            fRadius = _hypot (cxClient / 2.0, cyClient / 2.0) ;

            for (fAngle = 0.0 ; fAngle < TWO_PI ; fAngle += TWO_PI /
            {
                  MoveToEx (hdc, 0, 0, NULL) ;
                  LineTo (hdc,  (int) ( fRadius * cos (fAngle) + 0.5),
                        (int) (-fRadius * sin (fAngle) + 0.5)) ;
            }
            EndPaint (hwnd, &ps) ;
            return 0 ;
      case WM_DESTROY:
            DeleteObject (hRgnClip) ;
            PostQuitMessage (0) ;
            return 0 ;
      }
      return DefWindowProc (hwnd, iMsg, wParam, lParam) ;
}
```



CLOVERWM_SIZE

CLOVERhRgnTemp

```
hRgnTemp [4]= CreateRectRgn (0, 0, 1, 1) ;

hRgnTemp [5]= CreateRectRgn (0, 0, 1, 1) ;

hRgnClip      = CreateRectRgn (0, 0, 1, 1) ;
```

```
CombineRgn (hRgnTemp [4], hRgnTemp [0], hRgnTemp [1], RGN
```

```
CombineRgn (hRgnTemp [5], hRgnTemp [2], hRgnTemp [3], RGN
```

```

```

hRgnClip

```
CombineRgn (hRgnClip, hRgnTemp [4], hRgnTemp [5], RGN_XOR
```

RGN_XOR6

```
for (i = 0 ; i < 6 ; i++)
    DeleteObject (hRgnTemp [i]) ;
```

WM_PAINTWM_SIZE

```
SetViewportOrg (hdc, xClient / 2, yClient / 2) ;
SelectClipRgn (hdc, hRgnClip) ;
```

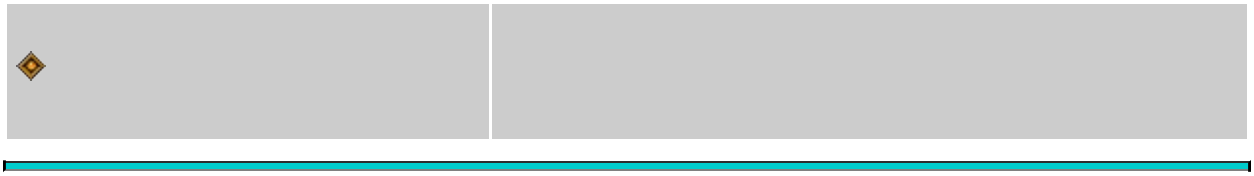
360fRadius

```
fRadius = hypot (xClient / 2.0, yClient / 2.0) ;
for (fAngle = 0.0 ; fAngle < TWO_PI ; fAngle += TWO_PI / 360)
{
    MoveToEx (hdc, 0, 0, NULL) ;
    LineTo (hdc, (int) ( fRadius * cos (fAngle) + 0.5),
                 (int) (-fRadius * sin (fAngle) + 0.5)) ;
}
```

WM_DESTROY

```
DeleteObject (hRgnClip) ;
```

MetaFile



Microsoft Windows 98WebPC

1874RemingtonHollerithPC

Windows

Windows

WindowsWindows

AltWindowsDefWindowProc

WindowsCtrlCtrl-S

WindowsWindowsWindows

Interface Services/Controls/Rich Edit Controls

Windows

Windows

MSGhwndDispatchMessage

NULLWindowsWindowsWindows

Windows

WM_SETFOCUSWM_KILLFOCUSWM_SETFOCUS
WM_KILLFOCUS

WindowsWindowsWindowsWindows
Windows

Windows

ACtrl
LockACtrlCtrl-AASCIIWindowsA(dead-character
key)ShiftCtrlAltàá

WindowsshiftInsertDeleteWindows

WindowsWM_KEYDOWNWM_SYSKEYDOWNWindows
WM_KEYUPWM_SYSKEYUP

6-1

WM_KEYDOWN	WM_KEYUP

WM_SYSKEYDOWN	WM_SYSKEYUP

downupWindowsWM_KEYDOWN
WM_SYSKEYDOWNWM_KEYUPWM_SYSKEYUP
GetMessageTime

WM_SYSKEYDOWNWM_SYSKEYUPSYSWindowsWindows
WM_SYSKEYDOWNWM_SYSKEYUPAltAlt-TabAlt-Esc
AltAlt-F4WM_SYSKEYUPWM_SYSKEYDOWN
DefWindowProcWindowsAlt
DefWindowProcWindows

DefWindowProcWindows

```
case      WM_SYSKEYDOWN:

case      WM_SYSKEYUP:

caseWM_SYSCHAR:

     return 0 ;
```

AltWM_SYSCHARAlt-TabAlt-Esc

WM_KEYDOWNWM_KEYUPAltWindows

wParamlParam

WM_KEYDOWNWM_KEYUPWM_SYSKEYDOWNWM_SYSKEYUP
wParam

DOSWindows

Windows(scan
21YWindowsIBM

WINUSER.HVK_6-2IBMWindows

6-2

		WINUSER.H		IBM
1	01	VK_LBUTTON		
2	02	VK_RBUTTON		
3	03	VK_CANCEL	ˇ	Ctrl-Break
4	04	VK_MBUTTON		

VK_CANCEL(Ctrl-Break)Windows

6-3--BackspaceTabEnterEscapeSpacebarWindowsWindows

6-3

		WINUSER.H		IBM

8	08	VK_BACK	ˇ	Backspace
9	09	VK_TAB	ˇ	Tab
12	0C	VK_CLEAR		Num Lock5
13	0D	VK_RETURN	ˇ	Enter
16	10	VK_SHIFT	ˇ	Shift
17	11	VK_CONTROL	ˇ	Ctrl
18	12	VK_MENU	ˇ	Alt
19	13	VK_PAUSE		Pause
20	14	VK_CAPITAL	ˇ	Caps Lock
27	1B	VK_ESCAPE	ˇ	Esc
32	20	VK_SPACE	ˇ	Spacebar

WindowsShiftCtrlAlt

6-4VK_INSERTVK_DELETE

		WINUSER.H		**IBM**
33	21	VK_PRIOR	ˇ	Page Up
34	22	VK_NEXT	ˇ	Page Down
35	23	VK_END	ˇ	End
36	24	VK_HOME	ˇ	Home
37	25	VK_LEFT	ˇ	
38	26	VK_UP	ˇ	
39	27	VK_RIGHT	ˇ	
40	28	VK_DOWN	ˇ	
41	29	VK_SELECT		
42	2A	VK_PRINT		
43	2B	VK_EXECUTE		

44	2C	VK_SNAPSHOT		Print Screen
45	2D	VK_INSERT	ˇ	Insert
46	2E	VK_DELETE	ˇ	Delete
47	2F	VK_HELP		

VK_PRIORVK_NEXTPrint
VK_SELECTVK_PRINTVK_EXECUTEVK_HELP

Windows

6-5

		WINUSER.H		**IBM**
48-57	30-39		ˇ	09
65-90	41-5A		ˇ	AZ

ASCIIWindowsASCII

6-6Microsoft Natural Keyboard

6-6

		WINUSER.H		IBM
91	5B	VK_LWIN		Windows
92	5C	VK_RWIN		Windows
93	5D	VK_APPS		Applications

WindowsVK_LWINVK_RWINWindowsMicrosoft
Windows NTWindows for Applicationsapplicatio

6-7

6-7

		WINUSER.H		IBM
96-105	60-69	VK_NUMPAD0VK_ NUMPAD9		NumLock09
106	6A	VK_MULTIPLY		*
107	6B	VK_ADD		+
108	6C	VK_SEPARATOR		

109	6D	VK_SUBTRACT		-
110	6E	VK_DECIMAL		.
111	6F	VK_DIVIDE		/

12Windows10246-8

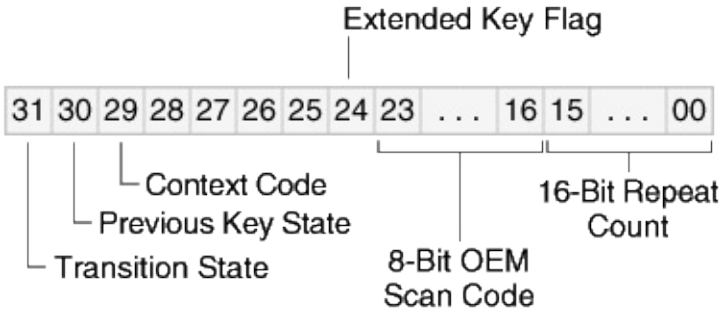
6-8

		WINUSER.H		**IBM**
112-121	70-79	VK_F1VK_F10	ˇ	F1F10
122-135	7A-87	VK_F11VK_F24		F11F24
144	90	VK_NUMLOCK		Num Lock
145	91	VK_SCROLL		Scroll Lock

/
Key Codes

lParam

WM_KEYDOWNWM_KEYUPWM_SYSKEYDOWN
WM_SYSKEYUPwParamlParamlParam3266-1



6-1 lParam6

1WindowsWM_KEYDOWN
WM_SYSKEYDOWNWM_KEYUPWM_SYSKEYUP1

1

OEM

OEMPCROM
ManufacturerIBMWindowsOEM

IBM1IBM101102AltCtrlInsertDelete
EnterNum Lock1Windows

ALT1WM_SYSKEYUPWM_SYSKEYDOWN1
WM_SYSKEYUPWM_KEYDOW0

- WM_SYSKEYUPWM_SYSKEYDOWNAlt0Windows
  WM_SYSKEYUPWM_SYSKEYDOWN

- ShiftCtrlAlt1

01WM_KEYUPWM_SYSKEYUP1WM_KEYDOWN
WM_SYSKEYDOWN011

01WM_KEYDOWNWM_SYSKEYDOWN0WM_KEYUP
WM_SYSKEYUP1

ShiftCtrlAltCaps

```
iState = GetKeyState (VK_SHIFT) ;
```

ShiftiStateCaps Lock

```
iState = GetKeyState (VK_CAPITAL) ;
```

1

GetKeyStateVK_SHIFTVK_CONTROLVK_MENUAlt
GetKeyStateShiftCtrlAltVK_LSHIFTVK_RSHIFT
VK_LCONTROLVK_RCONTROLVK_LMENUVK_RMENU
GetKeyStateGetAsyncKeyState

VK_LBUTTONVK_RBUTTONVK_MBUTTONWindows

GetKeyStateShift-TabTabWM_KEYDOWNGetKeyState

VK_SHIFTGetKeyStateTabShiftTabShiftTabShift

GetKeyStateF1

```
while (GetKeyState (VK_F1) >= 0) ;    // WRONG !!!
```

F1WM_KEYDOWNGetAsyncKeyState

WindowsWM_SYSKEYDOWNWM_SYSKEYUPWindows
WM_KEYDOWNWM_KEYUP

WindowsWM_KEYDOWNwParam0x33WM_KEYDOWN
3GetKeyStateShift#

InsertDeleteWM_KEYDOWN I

WindowsMS-DOSShiftCtrlAltWindowsMicrosoft
Windows

InsertDeleteWM_KEYDOWNGetKeyStateShiftCtrl
WindowsShiftCtrlCtrl

Windows

SYSMETS

SYSMETSHomeEndPage
Up ArrowDown Arrow

WM_VSCROLLWM_HSCROLLWM_KEYDOWN
WM_KEYDOWN

WM_KEYDOWNWM_VSCROLLWM_HSCROLLWndProc

WindowsSendMessage

SendMessage (hwnd, message, wParam, lParam) ;

SendMessageWindowshwndWindowsSendMessage

SYSMETSSendMessageWM_KEYDOWN

```
caseWM_KEYDOWN:
   switch (wParam)
  {
   case   VK_HOME:
         SendMessage (hwnd, WM_VSCROLL, SB_TOP, 0) ;
         break ;
   case   VK_END:
         SendMessage (hwnd, WM_VSCROLL, SB_BOTTOM, 0) ;
         break ;

   case   VK_PRIOR:
         SendMessage (hwnd, WM_VSCROLL, SB_PAGEUP, 0) ;
         break ;
```

SYSMETS3WM_VSCROLLSB_TOPSB_BOTTOM
HomeEnd6-1SYSENTS4

6-1　SYSMETS4

```
SYSMETS4.C

/*------------------------------------------------------------------------
  SYSMETS4.C -- System Metrics Display Program No. 4
                (c) Charles Petzold, 1998
------------------------------------------------------------------------*/

#include <windows.h>
#include "sysmets.h"
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                    PSTR szCmdLine, int iCmdShow)
{
   static TCHAR szAppName[]      = TEXT ("SysMets4") ;
   HWND                 hwnd ;
   MSG                  msg ;
   WNDCLASS             wndclass ;

   wndclass.style               = CS_HREDRAW | CS_VREDRAW ;
   wndclass.lpfnWndProc  = WndProc ;
```

```
wndclass.cbClsExtra          = 0 ;
wndclass.cbWndExtra          = 0 ;
wndclass.hInstance           = hInstance ;
wndclass.hIcon               = LoadIcon (NULL, IDI_APPLICATION)
wndclass.hCursor             = LoadCursor (NULL, IDC_ARROW) ;
wndclass.hbrBackground= (HBRUSH) GetStockObject (WHITE_
wndclass.lpszMenuName= NULL ;
wndclass.lpszClassName= szAppName ;

if (!RegisterClass (&wndclass))
{
      MessageBox (NULL, TEXT ("Program requires Windows NT
            szAppName, MB_ICONERROR) ;
      return 0 ;
}

hwnd = CreateWindow (szAppName, TEXT ("Get System Metr
                                    WS_OVERLAPPEDWINDOW | W
```

```
                              CW_USEDEFAULT, CW_USEDEFAULT,
                              CW_USEDEFAULT, CW_USEDEFAULT,
                              NULL, NULL, hInstance, NULL) ;

   ShowWindow (hwnd, iCmdShow) ;
   UpdateWindow (hwnd) ;

   while (GetMessage (&msg, NULL, 0, 0))
   {
         TranslateMessage (&msg) ;
         DispatchMessage (&msg) ;
   }
         return msg.wParam ;
}
LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
   static int  cxChar, cxCaps, cyChar, cxClient, cyClient, iMaxWid
   HDC                 hdc ;
   int                 i, x, y, iVertPos, iHorzPos, iPaintBeg, iPaintEnd ;
```

```
PAINTSTRUCT   ps ;
SCROLLINFO    si ;
TCHAR              szBuffer[10] ;
TEXTMETRIC    tm ;

switch (message)
{
case   WM_CREATE:
       hdc = GetDC (hwnd) ;

       GetTextMetrics (hdc, &tm) ;
       cxChar= tm.tmAveCharWidth ;
       cxCaps= (tm.tmPitchAndFamily & 1 ? 3 : 2) * cxChar / 2 ;
       cyChar= tm.tmHeight + tm.tmExternalLeading ;

       ReleaseDC (hwnd, hdc) ;

            // Save the width of the three columns
```

```
        iMaxWidth = 40 * cxChar + 22 * cxCaps ;
        return 0 ;

case   WM_SIZE:
        cxClient            = LOWORD (lParam) ;
        cyClient            = HIWORD (lParam) ;

             // Set vertical scroll bar range and page size

        si.cbSize     = sizeof (si) ;
        si.fMask      = SIF_RANGE | SIF_PAGE ;
        si.nMin       = 0 ;
        si.nMax       = NUMLINES - 1 ;
        si.nPage      = cyClient / cyChar ;
        SetScrollInfo (hwnd, SB_VERT, &si, TRUE) ;

                  // Set horizontal scroll bar range and page size

        si.cbSize     = sizeof (si) ;
        si.fMask      = SIF_RANGE | SIF_PAGE ;
        si.nMin       = 0 ;
```

```
        si.nMax       = 2 + iMaxWidth / cxChar ;
        si.nPage      = cxClient / cxChar ;
        SetScrollInfo (hwnd, SB_HORZ, &si, TRUE) ;
        return 0 ;

case WM_VSCROLL:
                // Get all the vertical scroll bar information
        si.cbSize     = sizeof (si) ;
        si.fMask      = SIF_ALL ;
        GetScrollInfo (hwnd, SB_VERT, &si) ;

                // Save the position for comparison later on

             iVertPos = si.nPos ;

             switch (LOWORD (wParam))
        {
case SB_TOP:
        si.nPos = si.nMin ;
        break ;
```

```
case SB_BOTTOM:
        si.nPos = si.nMax ;
        break ;

case SB_LINEUP:
        si.nPos -= 1 ;
        break ;

case SB_LINEDOWN:
        si.nPos += 1 ;
        break ;

case SB_PAGEUP:
        si.nPos -= si.nPage ;
        break ;

case SB_PAGEDOWN:
```

```
            si.nPos += si.nPage ;
            break ;

        case SB_THUMBTRACK:
            si.nPos = si.nTrackPos ;
            break ;

        default:
            break ;
        }
            // Set the position and then retrieve it.  Due to adjus
            //   by Windows it might not be the same as the valu

        si.fMask = SIF_POS ;
        SetScrollInfo (hwnd, SB_VERT, &si, TRUE) ;
        GetScrollInfo (hwnd, SB_VERT, &si) ;

            // If the position has changed, scroll the window and

        if (si.nPos != iVertPos)
```

```
        {
             ScrollWindow (hwnd, 0, cyChar * (iVertPos - si.nPos),
                                                    NULL, NULL)
             UpdateWindow (hwnd) ;
        }
        return 0 ;

case WM_HSCROLL:
                  // Get all the vertical scroll bar information
        si.cbSize           = sizeof (si) ;
        si.fMask            = SIF_ALL ;

                  // Save the position for comparison later on

             GetScrollInfo (hwnd, SB_HORZ, &si) ;
             iHorzPos      = si.nPos ;

        switch (LOWORD (wParam))
        {
case SB_LINELEFT:
```

```
        si.nPos -= 1 ;
        break ;

    case SB_LINERIGHT:
        si.nPos += 1 ;
        break ;

    case SB_PAGELEFT:
        si.nPos -= si.nPage ;
        break ;

    case SB_PAGERIGHT:
        si.nPos += si.nPage ;
        break ;

    case SB_THUMBPOSITION:
        si.nPos = si.nTrackPos ;
        break ;
```

```
        default:
        break ;
        }
                // Set the position and then retrieve it.  Due to a
                //   by Windows it might not be the same as the

        si.fMask = SIF_POS ;
        SetScrollInfo (hwnd, SB_HORZ, &si, TRUE) ;
        GetScrollInfo (hwnd, SB_HORZ, &si) ;

                // If the position has changed, scroll the window

        if (si.nPos != iHorzPos)
        {
             ScrollWindow (hwnd, cxChar * (iHorzPos - si.nPos), 0
             NULL, NULL) ;
        }
        return 0 ;

case WM_KEYDOWN:
```

```
        switch (wParam)
{
case VK_HOME:
        SendMessage (hwnd, WM_VSCROLL, SB_TOP, 0) ;
        break ;

case VK_END:
        SendMessage (hwnd, WM_VSCROLL, SB_BOTTOM, 0) ;
        break ;

case VK_PRIOR:
        SendMessage (hwnd, WM_VSCROLL, SB_PAGEUP, 0) ;
        break ;

case VK_NEXT:
        SendMessage (hwnd, WM_VSCROLL, SB_PAGEDOWN, 0) ;
        break ;
```

```
case VK_UP:
        SendMessage (hwnd, WM_VSCROLL, SB_LINEUP, 0) ;
        break ;

case VK_DOWN:
        SendMessage (hwnd, WM_VSCROLL, SB_LINEDOWN, 0) ;
        break ;


case VK_LEFT:
        SendMessage (hwnd, WM_HSCROLL, SB_PAGEUP, 0) ;
        break ;


case VK_RIGHT:
        SendMessage (hwnd, WM_HSCROLL, SB_PAGEDOWN, 0)
        break ;
        }
        return 0 ;
case WM_PAINT:
        hdc = BeginPaint (hwnd, &ps) ;
```

```
                // Get vertical scroll bar position

     si.cbSize           = sizeof (si) ;

     si.fMask            = SIF_POS ;

     GetScrollInfo (hwnd, SB_VERT, &si) ;

     iVertPos            = si.nPos ;

                // Get horizontal scroll bar position

     GetScrollInfo (hwnd, SB_HORZ, &si) ;

     iHorzPos            = si.nPos ;

                // Find painting limits

     iPaintBeg           = max (0, iVertPos + ps.rcPaint.top / cy

     iPaintEnd           = min (NUMLINES - 1,

     iVertPos + ps.rcPaint.bottom / cyChar) ;

     for (i = iPaintBeg ; i <= iPaintEnd ; i++)

     {

          x = cxChar * (1 - iHorzPos) ;

          y = cyChar * (i - iVertPos) ;
```

```
        TextOut (hdc, x, y,
        sysmetrics[i].szLabel,
        lstrlen (sysmetrics[i].szLabel)) ;

        TextOut (hdc, x + 22 * cxCaps, y,
        sysmetrics[i].szDesc,
        lstrlen (sysmetrics[i].szDesc)) ;

        SetTextAlign (hdc, TA_RIGHT | TA_TOP) ;

        TextOut (hdc, x + 22 * cxCaps + 40 * cxChar, y, szBuffer
      wsprintf (szBuffer, TEXT ("%5d"),
        GetSystemMetrics (sysmetrics[i].iIndex))) ;

        SetTextAlign (hdc, TA_LEFT | TA_TOP) ;
        }

        EndPaint (hwnd, &ps) ;
        return 0 ;
```

```
   case   WM_DESTROY:
          PostQuitMessage (0) ;
          return 0 ;
   }
          return DefWindowProc (hwnd, message, wParam, lParam
}
```

/Windows

```
while      (GetMessage (&msg, NULL, 0, 0))
{
   TranslateMessage (&msg) ;
   DispatchMessage (&msg) ;
}
```

WinMainGetMessagemsgDispatchMessage

TranslateMessageWM_KEYDOWNWM_SYSKEYDOWN
TranslateMessageGetMessage

6-9

6-9

WM_CHAR	WM_DEADCHAR
WM_SYSCHAR	WM_SYSDEADCHAR

WM_CHARWM_DEADCHARWM_KEYDOWNWM_SYSCHAR
WM_SYSDEADCHARWM_SYSKEYDOWN

WindowsWM_CHARlParamlParamwParamANSI
Unicode

08ANSI16UnicodeRegisterClassARegisterClassANSI
ANSIRegisterClassWRegisterClassUnicodeRegisterClass
UNICODERegisterClassWRegisterClassA

ANSIUnicodeWM_CHAR

```
(TCHAR) wParam
```

RegisterClassARegisterClassWANSIUnicodeUnicode

```
fUnicode = IsWindowUnicode (hwnd) ;
```

hwndUnicodefUnicodeTRUERegisterClassW

TranslateMessageWM_KEYDOWNWM_SYSKEYDOWNCaps LockA6-10

6-10

WM_KEYDOWN	A0x41
WM_CHAR	a0x61
WM_KEYUP	A0x41

ShiftAAShiftA6-11

6-11

WM_KEYDOWN	VK_SHIFT 0x10
WM_KEYDOWN	A0x41
WM_CHAR	A0x41

WM_KEYUP	A0x41
WM_KEYUP	VK_SHIFT0x10

Shift

AWM_KEYDOWN6-12

6-12

WM_KEYDOWN	A0x41
WM_CHAR	a0x61
WM_KEYDOWN	A0x41
WM_CHAR	a0x61
WM_KEYDOWN	A0x41
WM_CHAR	a0x61

WM_KEYDOWN	A0x41
WM_CHAR	a0x61
WM_KEYUP	A0x41

WM_KEYDOWN1WM_CHAR

Ctrl0x01Ctrl-A0x1ACtrl-ZASCII6-13

6-13

			ANSI C
Backspace	0x08	Ctrl-H	\b
Tab	0x09	Ctrl-I	\t
Ctrl-Enter	0x0A	Ctrl-J	\n
Enter	0x0D	Ctrl-M	\r
Esc	0x1B	Ctrl-[	

ANSI C

WindowsCtrl

WM_CHARDeleteInsertShiftCtrlAlt
WM_KEYDOWN

TabEnterBackspaceEscape6-13ASCIIWindows
WM_CHARWM_KEYDOWN

10WindowsTabEnterBackspaceEscapeWM_CHAR

```
case WM_CHAR:
        //
   switch (wParam)
   {
   case '\b':          // backspace
        //
        break ;
   case '\t':          // tab
        //
        break ;
   case '\n':          // linefeed
        //
        break ;
```

```
case '\r':          // carriage return
        //
        break ;

default:                    // character codes
        //
        break ;
}
return 0 ;
```

WindowsWM_DEADCHARWM_SYSDEADCHAR

U.S.U.S.+/=ShiftShift

wParamASCIIUnicodeWM_DEADCHARAWM_CHAR
wParamaANSI

WM_DEADCHARWM_CHARWindowss
WM_CHARwParamASCIIWM_DEADCHARwParamwParamsASCII

KEYVIEW1

WindowsWindows

UnicodeWindows NT U

KEYVIEW1

Windows6-2KEYVIEW1Windows8

6-2　KEYVIEW1

```
KEYVIEW1.C
/*------------------------------------------------------------------
   KEYVIEW1.C --Displays Keyboard and Character Messages
                        (c) Charles Petzold, 1998
------------------------------------------------------------------*/
#include <windows.h>
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                    PSTR szCmdLine, int iCmdShow)
{
   static TCHAR szAppName[]      = TEXT ("KeyView1") ;
   HWND                 hwnd ;
   MSG                  msg ;
   WNDCLASS             wndclass ;
```

```
wndclass.style                  = CS_HREDRAW | CS_VREDRAW
wndclass.lpfnWndProc            = WndProc ;
wndclass.cbClsExtra             = 0 ;
wndclass.cbWndExtra             = 0 ;
wndclass.hInstance                  = hInstance ;
wndclass.hIcon                  = LoadIcon (NULL, IDI_APPLICATI
wndclass.hCursor                    = LoadCursor (NULL, IDC_ARRO
wndclass.hbrBackground                  = (HBRUSH) GetStockObjec
wndclass.lpszMenuName           = NULL ;
wndclass.lpszClassName          = szAppName ;

if (!RegisterClass (&wndclass))
{
     MessageBox (NULL, TEXT ("This program requires Windo
                         szAppName, MB_ICONERROR) ;
     return 0 ;
}

hwnd = CreateWindow (szAppName, TEXT ("Keyboard Messag
```

```
                                    WS_OVERLAPPEDWINDOW,
                                    CW_USEDEFAULT, CW_USEDEFAU
                                    CW_USEDEFAULT, CW_USEDEFAU
                                    NULL, NULL, hInstance, NULL) ;

   ShowWindow (hwnd, iCmdShow) ;
   UpdateWindow (hwnd) ;

   while (GetMessage (&msg, NULL, 0, 0))
   {
         TranslateMessage (&msg) ;
         DispatchMessage (&msg) ;
   }
         return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
   static int   cxClientMax, cyClientMax, cxClient, cyClient, cxCha
```

```
static int   cLinesMax, cLines ;
static PMSG  pmsg ;
static RECT  rectScroll ;
static TCHAR szTop[] = TEXT ("Message Key  Char ")
                                    TEXT ("Repeat Scan Ext ALT
static TCHAR szUnd[] = TEXT ("_______       ___      ____     ")
                                    TEXT ("______ ____ ___ ___ __

static TCHAR * szFormat[2] = {
                TEXT ("%-13s %3d %-15s%c%6u %4d %3s %3s
                TEXT ("%-13s  0x%04X%1s%c %6u %4d %3s %
      static TCHAR * szYes  = TEXT ("Yes") ;
      static TCHAR * szNo   = TEXT ("No") ;
      static TCHAR * szDown = TEXT ("Down") ;
      static TCHAR * szUp   = TEXT ("Up") ;

      static TCHAR * szMessage [] = {
      TEXT ("WM_KEYDOWN"),  TEXT ("WM_KEYUP"),
      TEXT ("WM_CHAR"),     TEXT ("WM_DEADCHAR"),
      TEXT ("WM_SYSKEYDOWN"),TEXT ("WM_SYSKEYUP"),
```

```
        TEXT ("WM_SYSCHAR"),  TEXT ("WM_SYSDEADCHAR") } ;
        HDC                 hdc ;
        int                 i, iType ;
        PAINTSTRUCT         ps ;
        TCHAR               szBuffer[128], szKeyName [32] ;
        TEXTMETRIC          tm ;

switch (message)
{
case WM_CREATE:
case WM_DISPLAYCHANGE:
                    // Get maximum size of client area
        cxClientMax = GetSystemMetrics (SM_CXMAXIMIZED) ;
        cyClientMax = GetSystemMetrics (SM_CYMAXIMIZED) ;

                    // Get character size for fixed-pitch font
        hdc = GetDC (hwnd) ;
        SelectObject (hdc, GetStockObject (SYSTEM_FIXED_FONT
        GetTextMetrics (hdc, &tm) ;
```

```
        cxChar = tm.tmAveCharWidth ;
        cyChar = tm.tmHeight ;

        ReleaseDC (hwnd, hdc) ;
                    // Allocate memory for display lines
        if (pmsg)
              free (pmsg) ;
              cLinesMax = cyClientMax / cyChar ;
              pmsg = malloc (cLinesMax * sizeof (MSG)) ;
              cLines = 0 ;
              // fall through
case WM_SIZE:
        if (message == WM_SIZE)
        {
              cxClient = LOWORD (lParam) ;
              cyClient = HIWORD (lParam) ;
        }
                          // Calculate scrolling rectangle
        rectScroll.left      = 0 ;
```

```
        rectScroll.right      = cxClient ;
        rectScroll.top        = cyChar ;
        rectScroll.bottom     = cyChar * (cyClient / cyChar) ;

        InvalidateRect (hwnd, NULL, TRUE) ;
        return 0 ;


case WM_KEYDOWN:
case WM_KEYUP:
case WM_CHAR:
case WM_DEADCHAR:
case WM_SYSKEYDOWN:
case WM_SYSKEYUP:
case WM_SYSCHAR:
case WM_SYSDEADCHAR:
                        // Rearrange storage array
        for (i = cLinesMax - 1 ; i > 0 ; i--)
        {
                pmsg[i] = pmsg[i - 1] ;
```

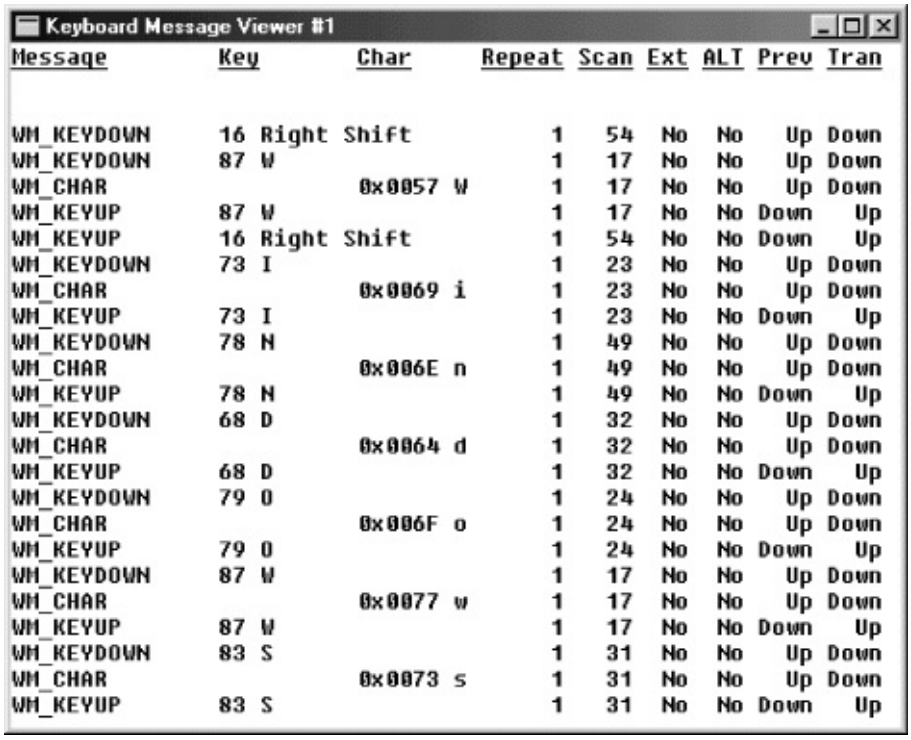
```
        }
                        // Store new message
        pmsg[0].hwnd = hwnd ;
        pmsg[0].message = message ;
        pmsg[0].wParam = wParam ;
        pmsg[0].lParam = lParam ;

        cLines = min (cLines + 1, cLinesMax) ;
                        // Scroll up the display
        ScrollWindow (hwnd, 0, -cyChar, &rectScroll, &rectScroll)
        break ;             // i.e., call DefWindowProc so Sys messa
case WM_PAINT:
        hdc = BeginPaint (hwnd, &ps) ;
        SelectObject (hdc, GetStockObject (SYSTEM_FIXED_FONT
        SetBkMode (hdc, TRANSPARENT) ;
        TextOut (hdc, 0, 0, szTop, lstrlen (szTop)) ;
        TextOut (hdc, 0, 0, szUnd, lstrlen (szUnd)) ;

        for (i = 0 ; i < min (cLines, cyClient / cyChar - 1) ; i++)
```

```
        {
             iType =       pmsg[i].message == WM_CHAR ||
                           pmsg[i].message == WM_SYSCHAR ||
                           pmsg[i].message == WM_DEADCHAR ||
                           pmsg[i].message == WM_SYSDEADCHAR

             GetKeyNameText (pmsg[i].lParam, szKeyName,
                                 sizeof (szKeyName) / sizeof (TCHAR))

        TextOut (hdc, 0, (cyClient / cyChar - 1 - i) * cyChar, szBuf
        wsprintf (szBuffer, szFormat [iType],
        szMessage [pmsg[i].message - WM_KEYFIRST],
        pmsg[i].wParam,
        (PTSTR) (iType ? TEXT (" ") : szKeyName),
        (TCHAR) (iType ? pmsg[i].wParam : ' '),
        LOWORD (pmsg[i].lParam),
        HIWORD (pmsg[i].lParam) & 0xFF,
                    0x01000000 & pmsg[i].lParam ? szYes  : szNo,
                    0x20000000 & pmsg[i].lParam ? szYes  : szNo,
                    0x40000000 & pmsg[i].lParam ? szDown : szUp
```

```
                    0x80000000 & pmsg[i].lParam ? szUp   : szDow
        }
        EndPaint (hwnd, &ps) ;
        return 0 ;

   case   WM_DESTROY:
        PostQuitMessage (0) ;
        return 0 ;
   }
   return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

KEYVIEW1MSGKEYVIEW1WM_DISPLAYCHANGE
KEYVIEW1Cmalloc

6-2WindowsKEYVIEW1GetKeyNameTextChar
lParam



6-2 KEYVIEW1

KEYVIEW1 GetStockObjectSelect

```
SelectObject (hdc, GetStockObject (SYSTEM_FIXED_FONT)) ;
```

KEYVIEW1szTopszUndWM_PAINT
WindowsWindowsszUndszTop

```
SetBkMode (hdc, TRANSPARENT) ;
```

Windows

EN

UNICODEKEYVIEW1UnicodeKEYVIEW1RELEASE
Windowsabcde WM_CHARASCII0x610x620x630x6
abcde

KEYVIEW1=aeiou=WM_DEADCHARWM_CHAR
0xE10xE90xED0xF30xFAáéíóú

abcdeWM_CHAR0xE10xE20xF80xE40xE5á
T ? ? ?Ω ?/p>

abcdeWM_CHAR0xF40xE80xF10xE20xF3

GDIGDI

PCMicrosoft Developer Networ
KEYLOOK1abcdeASCII0x610x620x630x640x65
abcdeASCII

WindowsabcdeWM_CHAR0xE10xE20xF80xE40xE5
Windowsalphabetapsideltaepsilongamma
WindowscpsigammagNadine
Software for Windows 95 and Windows NT587

WindowsKEYVIEW1=aeiouWM_CHAR0xE10xE9
0xED0xF30xFAWindowsïáéíóú

abcde0xF40xE80xF10xE20xF3Windows

Windows

WindowsKEYVIEW1

WindowsIME
KEYVIEW1KEYVIEW1WM_CHAR128Nadine

KEYLOOK1WindowsWindowsWindowsWindows

KEYLOOK1

WindowsWindows 3.1TrueTyp

TrueTypeTrueTypeTrueTypeWindows
WYSIWYG

Windows

SYSTEM_FONTGetStockObjectKEYVIEW1SYSTEM_FIXED_FONT
GetStockObjectOEM_FIXED_FONT

SystemFixedSysTerminalCreateFontCreateFontIndirectWindows
FONTSWindowsWindows96
dpi6-14

6-14

GetStockObject			
SYSTEM_FONT	System	VGASYS.FON	8514SYS.FON
SYSTEM_FIXED_FONT	FixedSys	VGAFIX.FON	8514FIX.FON
OEM_FIXED_FONT	Terminal	VGAOEM.FON	8514OEM.FON

VGAVideo Graphics ArrayIBM1987IBM64
Windows96 dp
dpiWindows85148514IBM19871024×768

WindowsWindows Exp

WindowsMS S
SSERIFE.FON96 dpi81012141824GetStockObject
DEFAULT_GUI_FONTWindows

GetStockObjectANSI_FIXED_FONTANSI_VAR_FONT
DEVICE_DEFAULT_FONTWindowsSTOKFONT6-3

6-3 STOKFONT

```
STOKFONT.C

/*------------------------------------------------------------------------

   STOKFONT.C -- Stock Font Objects

                     (c) Charles Petzold, 1998

------------------------------------------------------------------------*/

#include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns

                    PSTR szCmdLine, int iCmdShow)

{

   static TCHAR szAppName[] = TEXT ("StokFont") ;
```

```
HWND                hwnd ;
MSG                 msg ;
WNDCLASS            wndclass ;

wndclass.style              = CS_HREDRAW | CS_VREDRAW ;
wndclass.lpfnWndProc  = WndProc ;
wndclass.cbClsExtra         = 0 ;
wndclass.cbWndExtra           = 0 ;
wndclass.hInstance          = hInstance ;
wndclass.hIcon              = LoadIcon (NULL, IDI_APPLICATION)
wndclass.hCursor            = LoadCursor (NULL, IDC_ARROW) ;
wndclass.hbrBackground= (HBRUSH) GetStockObject (WHITE_
wndclass.lpszMenuName = NULL ;
wndclass.lpszClassName= szAppName ;

if (!RegisterClass (&wndclass))
{
      MessageBox (  NULL, TEXT ("Program requires Windows N
```

```
                                        szAppName, MB_ICONERRO
        return 0 ;
    }

    hwnd = CreateWindow ( szAppName, TEXT ("Stock Fonts"),
                                        WS_OVERLAPPEDWINDOW |
                                         CW_USEDEFAULT, CW_USED
                                         CW_USEDEFAULT, CW_USED
                                         NULL, NULL, hInstance, NUL

    ShowWindow (hwnd, iCmdShow) ;
    UpdateWindow (hwnd) ;
    while (GetMessage (&msg, NULL, 0, 0))
    {
        TranslateMessage (&msg) ;
        DispatchMessage (&msg) ;
    }
    return msg.wParam ;
}
```

```
LRESULT CALLBACK WndProc ( HWND hwnd, UINT message, WPA
{
   static struct
   {
         int     idStockFont ;
         TCHAR * szStockFont ;
   }
   stockfont [] = { OEM_FIXED_FONT,           "OEM_FIXED_FONT
                              ANSI_FIXED_FONT,      "ANSI_FIXED_FO
                              ANSI_VAR_FONT,        "ANSI_VAR_FON
                              SYSTEM_FONT,          "SYSTEM_FONT"
                              DEVICE_DEFAULT_FONT,"DEVICE_DEFA
                              SYSTEM_FIXED_FONT,    "SYSTEM_FIXE
                              DEFAULT_GUI_FONT,     "DEFAULT_GUI

   static int  iFont, cFonts = sizeof stockfont / sizeof stockfont[0]
   HDC                  hdc ;
   int                  i, x, y, cxGrid, cyGrid ;
```

```
PAINTSTRUCT   ps ;
TCHAR              szFaceName [LF_FACESIZE], szBuffer [LF_FA
TEXTMETRIC  tm ;
switch (message)
{
case   WM_CREATE:
       SetScrollRange (hwnd, SB_VERT, 0, cFonts - 1, TRUE) ;
       return 0 ;

case   WM_DISPLAYCHANGE:
       InvalidateRect (hwnd, NULL, TRUE) ;
       return 0 ;

case   WM_VSCROLL:
       switch (LOWORD (wParam))
       {
       case SB_TOP:       iFont = 0 ;                break ;
case SB_BOTTOM:         iFont = cFonts - 1 ;      break ;
       case SB_LINEUP:
       case SB_PAGEUP:   iFont -= 1 ;                 break ;
```

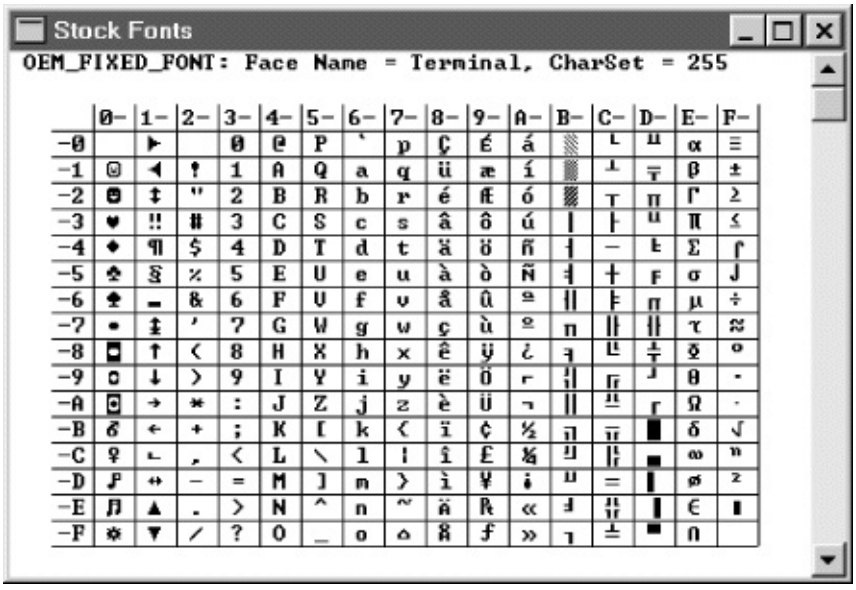
```
case SB_LINEDOWN:
case SB_PAGEDOWN:           iFont += 1 ;                    brea
case SB_THUMBPOSITION:iFont = HIWORD (wParam) ;     brea
        }
        iFont = max (0, min (cFonts - 1, iFont)) ;
        SetScrollPos (hwnd, SB_VERT, iFont, TRUE) ;
        InvalidateRect (hwnd, NULL, TRUE) ;
        return 0 ;

case   WM_KEYDOWN:
        switch (wParam)
        {
        case VK_HOME: SendMessage (hwnd, WM_VSCROLL, SB_
        case VK_END:  SendMessage (hwnd, WM_VSCROLL, SB_E
        case VK_PRIOR:
        case VK_LEFT:
        case VK_UP:   SendMessage (hwnd, WM_VSCROLL, SB_LI
        case VK_NEXT:
        case VK_RIGHT:
```

```
        case VK_DOWN: SendMessage (hwnd, WM_VSCROLL, SB_
        }
        return 0 ;

case   WM_PAINT:
        hdc = BeginPaint (hwnd, &ps) ;

        SelectObject (hdc, GetStockObject (stockfont[iFont].idSto
        GetTextFace (hdc, LF_FACESIZE, szFaceName) ;
        GetTextMetrics (hdc, &tm) ;
       cxGrid = max (3 * tm.tmAveCharWidth, 2 * tm.tmMaxCha
        cyGrid = tm.tmHeight + 3 ;

        TextOut (hdc, 0, 0, szBuffer,
        wsprintf (    szBuffer, TEXT (" %s: Face Name = %s, Char
                          stockfont[iFont].szStockFont,
                          szFaceName, tm.tmCharSet)) ;

SetTextAlign (hdc, TA_TOP | TA_CENTER) ;
        // vertical and horizontal lines
```

```
        for (i = 0 ; i < 17 ; i++)
        {
             MoveToEx (hdc, (i + 2) * cxGrid,  2 * cyGrid, NULL) ;
            LineTo   (hdc, (i + 2) * cxGrid, 19 * cyGrid) ;

             MoveToEx (hdc,      cxGrid, (i + 3) * cyGrid, NULL) ;
             LineTo   (hdc, 18 * cxGrid, (i + 3) * cyGrid) ;
        }
                    // vertical and horizontal headings

        for (i = 0 ; i < 16 ; i++)
        {
        TextOut (hdc, (2 * i + 5) * cxGrid / 2, 2 * cyGrid + 2, szBu
        wsprintf (szBuffer, TEXT ("%X-"), i)) ;

        TextOut (hdc, 3 * cxGrid / 2, (i + 3) * cyGrid + 2, szBuffe
        wsprintf (szBuffer, TEXT ("-%X"), i)) ;
        }
                    // characters

        for (y = 0 ; y < 16 ; y++)
```

```
        for (x = 0 ; x < 16 ; x++)
        {
        TextOut (hdc, (2 * x + 5) * cxGrid / 2,
                                                (y + 3) * cyGrid + 2, sz
        wsprintf (szBuffer, TEXT ("%c"), 16 * x + y)) ;
      }

        EndPaint (hwnd, &ps) ;
        return 0 ;

  case   WM_DESTROY:
        PostQuitMessage (0) ;
        return 0 ;
  }
  return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

WindowsSTOKFONTOEM_FIXED_FONTGetStockObject6-3



6-3　WindowsOEM_FIXED_FONT

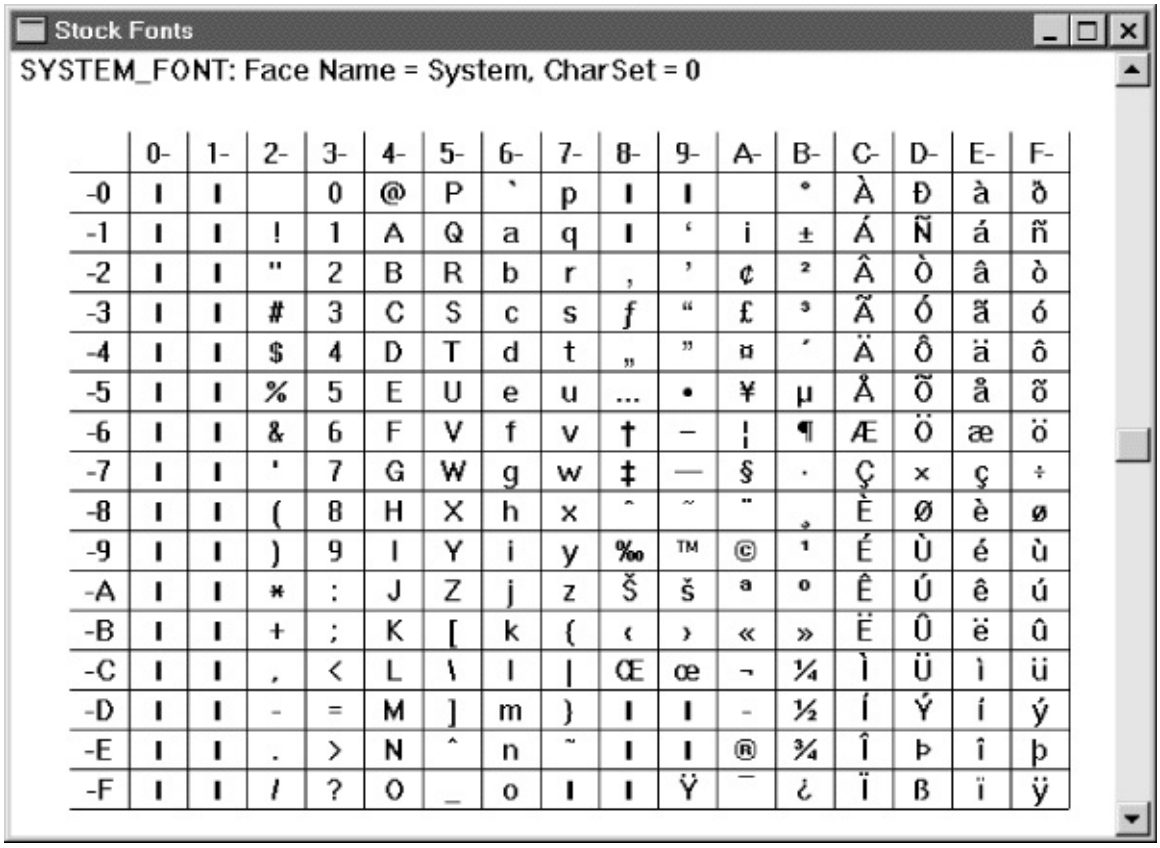
ASCIIASCII70x200x7EIBMIBM
ASCIIMS-DOSMS-DOS

WindowsWindowsWindows48IBM

WindowsIBMMS-DOSMS-DOSWindowsWindowsIBM
OEMIBM

WindowsOEMMS-DOS

IBMWindowsANSIAmerican
ISOInternational Standards　OrganizationISO8859Latin
European12526-4ANSIWindows



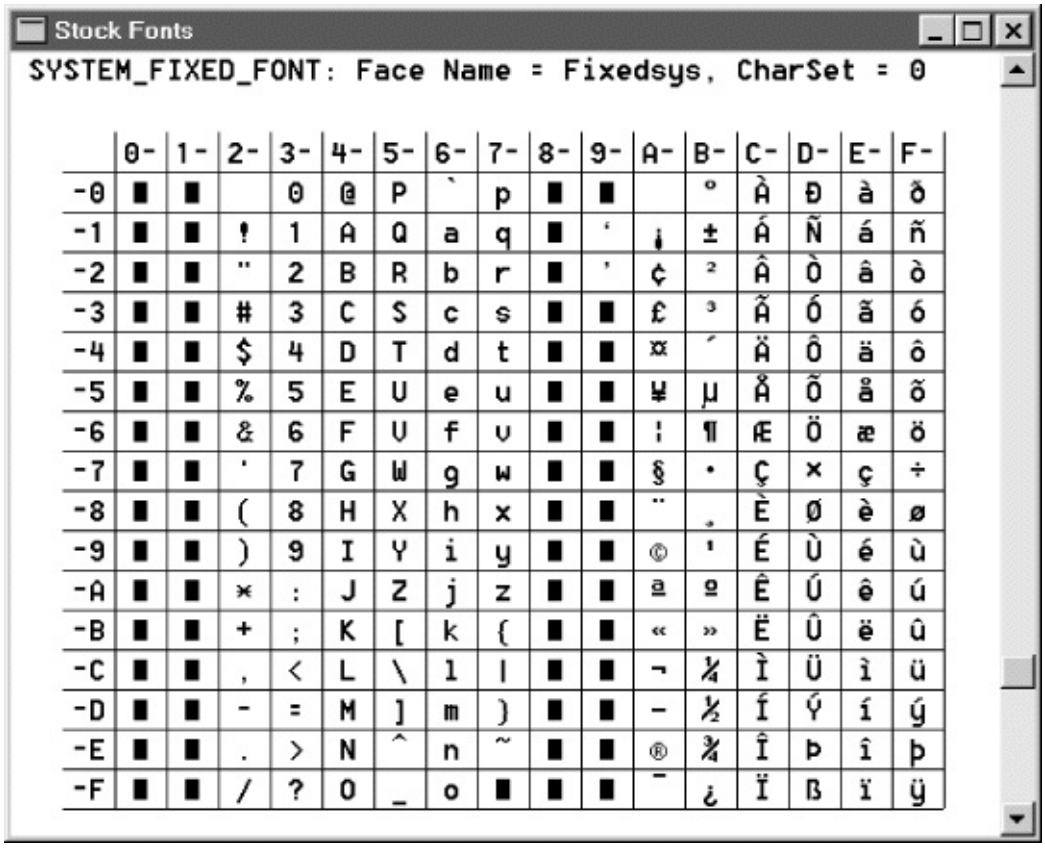
	0-	1-	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	▮	▮		0	@	P	`	p	▮	▮		°	À	Ð	à	ð
-1	▮	▮	!	1	A	Q	a	q	▮	‘	¡	±	Á	Ñ	á	ñ
-2	▮	▮	"	2	B	R	b	r	‚	’	¢	²	Â	Ò	â	ò
-3	▮	▮	#	3	C	S	c	s	ƒ	“	£	³	Ã	Ó	ã	ó
-4	▮	▮	$	4	D	T	d	t	„	”	¤	´	Ä	Ô	ä	ô
-5	▮	▮	%	5	E	U	e	u	…	•	¥	µ	Å	Õ	å	õ
-6	▮	▮	&	6	F	V	f	v	†	–	¦	¶	Æ	Ö	æ	ö
-7	▮	▮	'	7	G	W	g	w	‡	—	§	·	Ç	×	ç	÷
-8	▮	▮	(	8	H	X	h	x	ˆ	˜	¨	¸	È	Ø	è	ø
-9	▮	▮	)	9	I	Y	i	y	‰	™	©	¹	É	Ù	é	ù
-A	▮	▮	*	:	J	Z	j	z	Š	š	ª	º	Ê	Ú	ê	ú
-B	▮	▮	+	;	K	[	k	{	‹	›	«	»	Ë	Û	ë	û
-C	▮	▮	,	<	L	\	l	|	Œ	œ	¬	¼	Ì	Ü	ì	ü
-D	▮	▮	-	=	M	]	m	}	▮	▮	-	½	Í	Ý	í	ý
-E	▮	▮	.	>	N	^	n	~	▮	▮	®	¾	Î	Þ	î	þ
-F	▮	▮	/	?	O	_	o	▮	▮	Ÿ	¯	¿	Ï	ß	ï	ÿ

6-4 WindowsSYSTEM_FONT

0x200x7EASCIIASCII0x000x1F0x7F

0xC00xFFANSIWindows640xA0WW

ANSI0x800x9F6-5



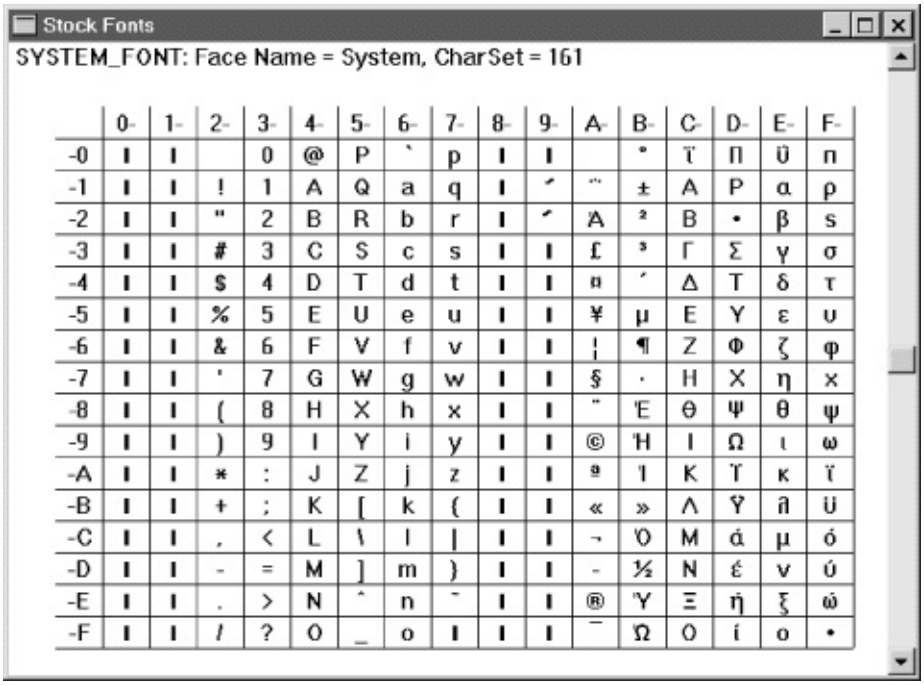
	0-	1-	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0				0	@	P	`	p				°	À	Ð	à	ð
-1			!	1	A	Q	a	q		‘	¡	±	Á	Ñ	á	ñ
-2			"	2	B	R	b	r		’	¢	²	Â	Ò	â	ò
-3			#	3	C	S	c	s			£	³	Ã	Ó	ã	ó
-4			$	4	D	T	d	t			¤	´	Ä	Ô	ä	ô
-5			%	5	E	U	e	u			¥	µ	Å	Õ	å	õ
-6			&	6	F	V	f	v			¦	¶	Æ	Ö	æ	ö
-7			'	7	G	W	g	w			§	·	Ç	×	ç	÷
-8			(	8	H	X	h	x			¨	¸	È	Ø	è	ø
-9			)	9	I	Y	i	y			©	¹	É	Ù	é	ù
-A			*	:	J	Z	j	z			ª	º	Ê	Ú	ê	ú
-B			+	;	K	[	k	{			«	»	Ë	Û	ë	û
-C			,	<	L	\	l	|			¬	¼	Ì	Ü	ì	ü
-D			-	=	M	]	m	}			-	½	Í	Ý	í	ý
-E			.	>	N	^	n	~			®	¾	Î	Þ	î	þ
-F			/	?	O	_	o				¯	¿	Ï	ß	ï	ÿ

6-5 WindowsSYSTEM_FIXED_FONT

Unicode0x00000x007FASCII0x00800x009F0x00000x001F0x00A0 0x00FFWindowsANSI

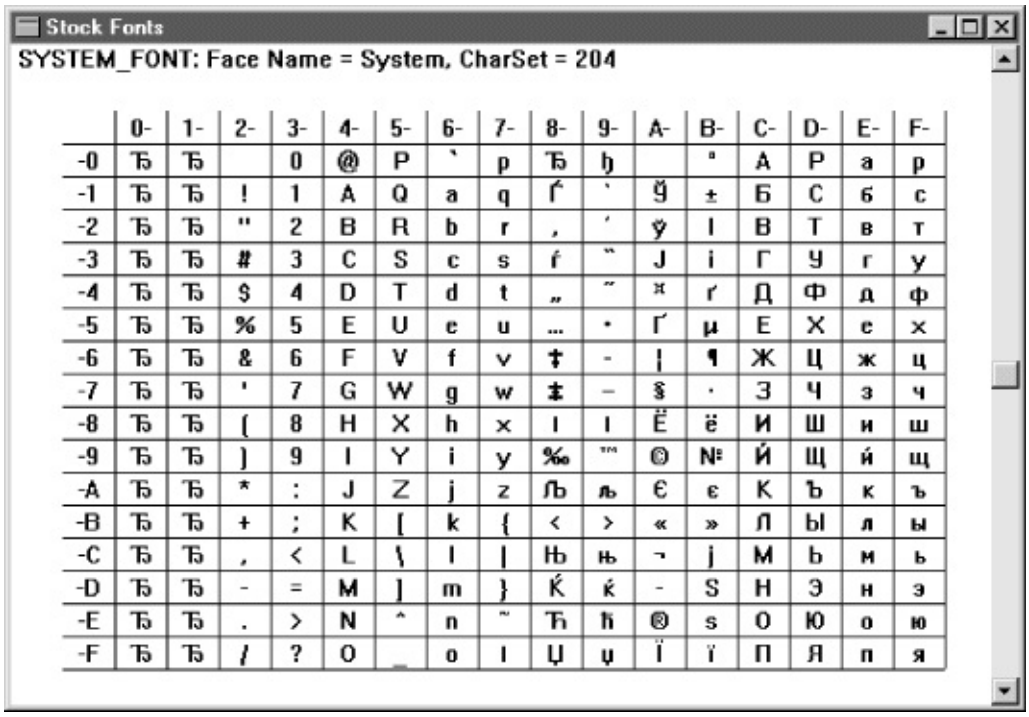
WindowsSYSTEM_FONTSYSTEM_FIXED_FONTGetStockObjectANSI WindowsANSI

WindowsSYSTEM_FONT6-6



6-6 WindowsSYSTEM_FONT

SYSTEM_FIXED_FONT0xC00xFFWindows6-7

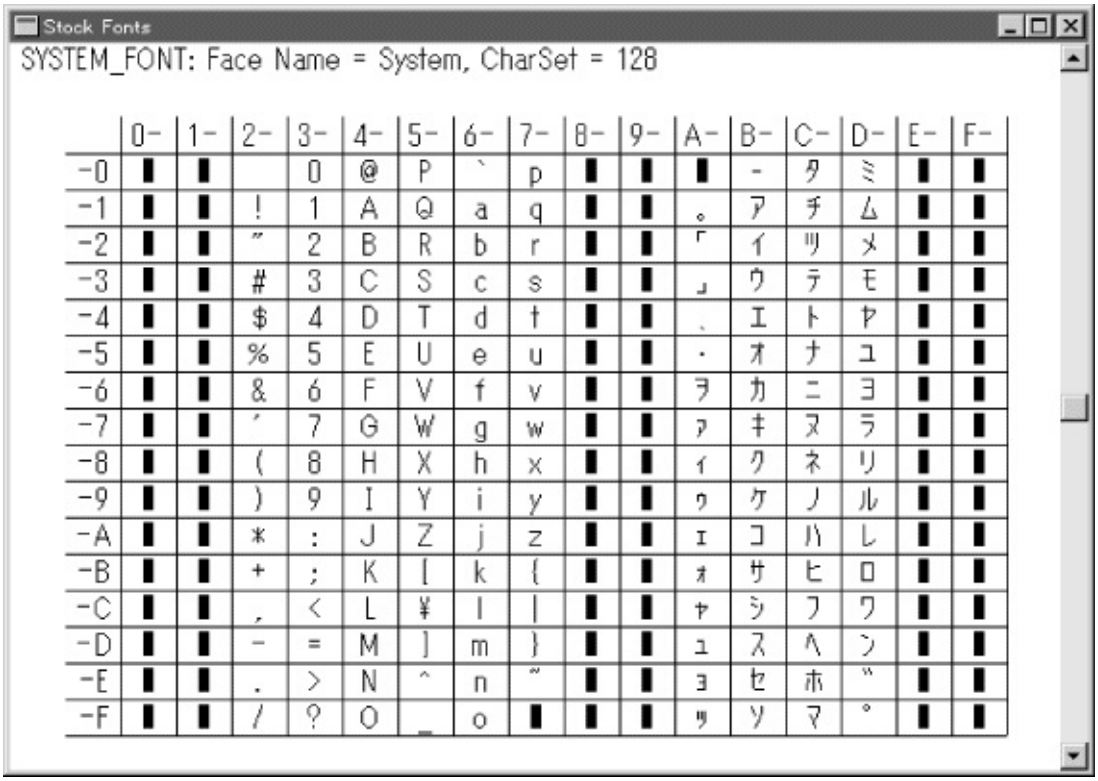


	0-	1-	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	Ђ	Ђ		0	@	P	`	p	Ђ	ђ		°	А	Р	а	р
-1	Ђ	Ђ	!	1	A	Q	a	q	Ґ	`	Ў	±	Б	С	б	с
-2	Ђ	Ђ	"	2	B	R	b	r	,	'	ў	І	В	Т	в	т
-3	Ђ	Ђ	#	3	C	S	c	s	ґ	"	J	і	Г	У	г	у
-4	Ђ	Ђ	$	4	D	T	d	t	„	~	¤	ґ	Д	Ф	д	ф
-5	Ђ	Ђ	%	5	E	U	e	u	…	•	Ґ	µ	Е	Х	е	х
-6	Ђ	Ђ	&	6	F	V	f	v	†	-	¦	¶	Ж	Ц	ж	ц
-7	Ђ	Ђ	'	7	G	W	g	w	‡	–	§	·	З	Ч	з	ч
-8	Ђ	Ђ	(	8	H	X	h	x	І	І	Ё	ё	И	Ш	и	ш
-9	Ђ	Ђ	)	9	I	Y	i	y	‰	™	©	№	Й	Щ	й	щ
-A	Ђ	Ђ	*	:	J	Z	j	z	Љ	љ	Є	є	К	Ъ	к	ъ
-B	Ђ	Ђ	+	;	K	[	k	{	<	>	«	»	Л	Ы	л	ы
-C	Ђ	Ђ	,	<	L	\	l	|	Њ	њ	¬	j	М	Ь	м	ь
-D	Ђ	Ђ	-	=	M	]	m	}	Ќ	ќ	-	S	Н	Э	н	э
-E	Ђ	Ђ	.	>	N	^	n	~	Ћ	ћ	®	s	О	Ю	о	ю
-F	Ђ	Ђ	/	?	O	_	o	I	Џ	џ	Ї	ї	П	Я	п	я

6-7　WindowsSYSTEM_FONT

0xC00xFF

6-8WindowsSYSTEM_FONT0xA50xDF



6-8　WindowsSYSTEM_FONT

6-8DBCSShift-JISJISJapanese
0x810x9F0xE00xFF0x400xFCNadine

KEYVIEW1abcdeWindowsWindowsWM_CHAR0xE1
0xE20xF80xE40xE5Windows

abcdeWindowsWindowsWM_CHAR0xF40xE80xF10xE20xF3
Windowsфисвy

=aeiouWindowsWindowsWM_CHAR0xE10xE9
0xED0xF30xFAWindowsamp;nbsp;áéíóú

WindowsWM_CHAR

Unicode

Windows　　NTUnicodeUNICODEKEYVIEW1Windows

UnicodeKEYVIEW1DEBUG

UNICODEKeyView1RegisterClassWRegisterClassAWndProc16
8WM_CHAR168

Windows NTUnicodeKEYVIEW1

Windows NTUnicodeKEYVIEW1Unicode0xFF256
UnicodeWindowsANSI

abcdeWM_CHARUnicode0x03B1
0x03B50xFFUnicoded
SYSTEM_FIXED_FONT256

abcdeKEYVIEW1WM_CHARUnicode0x04440x04380x0441
0x04320x0443φисвy

UnicodeKEYVIEW1UnicodeKEYVIEW1UnicodeKEYVIEW1
UnicodeUnicodeUnicode

UnicodeUnicodeKEYVIEW1

WM_CHAR168UnicodeKEYVIEW180xE1á
Unicode16a0x00E10x03B10x0431

UnicodeTextOutW16UnicodeTextOutA816GDI

Windows NTUnicodeKEYVIEW1GDI0x00000x00FF0x00FF
256

Windows NTWindows
Windows NTWindows

UnicodeKEYVIEW1Windows NTIME

TrueType

Windows2568WindowsSYSTEM_FONT
SYSTEM_FIXED_FONTTrueType.TTC

TrueType256TrueType256Windows
TrueType256 / **Windows**

Windows 98TrueTypeANSITrueType

UnicodeGetStockObject

CreateFontCreateFontIndirectCreatePenCreateBrushCreateFont14
CreateFontIndirectLOGFONTLOGFONT14CreateFont
CreateFont0

KEYVIEW1CreateFont13FIXED_PITCHCreateFont9
IDIDWINGDI.H

#define ANSI_CHARSET	0	// 1252 Latin 1 (ANSI)
#define DEFAULT_CHARSET	1	
#define SYMBOL_CHARSET	2	
#define MAC_CHARSET	77	
#define SHIFTJIS_CHARSET	128	// 932 (DBCS,)
#define HANGEUL_CHARSET	129	// 949 (DBCS,)
#define HANGUL_CHARSET	129	// " "

#define JOHAB_CHARSET	130	// 1361 (DBCS,)
#define GB2312_CHARSET	134	// 936 (DBCS,)
#define CHINESEBIG5_CHARSET	136	// 950 (DBCS,)
#define GREEK_CHARSET	161	// 1253
#define TURKISH_CHARSET	162	// 1254 Latin 5 ()
#define VIETNAMESE_CHARSET	163	// 1258
#define HEBREW_CHARSET	177	// 1255
#define ARABIC_CHARSET	178	// 1256
#define BALTIC_CHARSET	186	// 1257
#define RUSSIAN_CHARSET	204	// 1251 ()
#define THAI_CHARSET	222	// 874
#define EASTEUROPE_CHARSET	238	// 1250 Latin 2 ()
#define OEM_CHARSET	255	//

WindowsIDIDIDWindowsID1LOGFONTWindows
MS-DOSIDOEM_CHARSETMS-DOS

STOKFONTCharSetWindowsID0ANSI_CHARSET
255OEM_CHARSETWindows161GREEK_CHARSET
204RUSSIAN_CHARSET128SHIFTJIS_CHARSET

DBCSWindowsWindowsDBCSID

CreateFontHFONTSelectObjectDeleteObject

WM_INPUTLANGCHANGEWindowsWM_INPUTLANGCHANGE
wParamID

6-4KEYVIEW2

6-4　KEYVIEW2

```
KEYVIEW2.C

/*------------------------------------------------------------------------

  KEYVIEW2.C -- Displays Keyboard and Character Messages

                        (c) Charles Petzold, 1998

------------------------------------------------------------------------*/

#include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns

                    PSTR szCmdLine, int iCmdShow)

{
```

```
static TCHAR szAppName[] = TEXT ("KeyView2") ;
HWND         hwnd ;
MSG          msg ;
WNDCLASS     wndclass ;

wndclass.style         = CS_HREDRAW | CS_VREDRAW ;
wndclass.lpfnWndProc   = WndProc ;
wndclass.cbClsExtra    = 0 ;
wndclass.cbWndExtra    = 0 ;
wndclass.hInstance     = hInstance ;
wndclass.hIcon         = LoadIcon (NULL, IDI_APPLICATION)
wndclass.hCursor       = LoadCursor (NULL, IDC_ARROW)
wndclass.hbrBackground= (HBRUSH) GetStockObject (WHITE_
wndclass.lpszMenuName= NULL ;
wndclass.lpszClassName= szAppName ;

if (!RegisterClass (&wndclass))
{
MessageBox (NULL, TEXT ("This program requires Windows NT
```

```
                                        szAppName, MB_ICONERRO
        return 0 ;
}

hwnd = CreateWindow (szAppName, TEXT ("Keyboard Messag
                                    WS_OVERLAPPEDWINDOW,
                                    CW_USEDEFAULT, CW_USEDEFAU
                                    CW_USEDEFAULT, CW_USEDEFAU
                                    NULL, NULL, hInstance, NULL) ;

ShowWindow (hwnd, iCmdShow) ;
UpdateWindow (hwnd) ;

while (GetMessage (&msg, NULL, 0, 0))
      {
            TranslateMessage (&msg) ;
            DispatchMessage (&msg) ;
      }
return msg.wParam ;
```

```
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
   static DWORD dwCharSet = DEFAULT_CHARSET ;
   static int   cxClientMax, cyClientMax, cxClient, cyClient, cxCha
   static int   cLinesMax, cLines ;
   static PMSG  pmsg ;
   static RECT  rectScroll ;
   static TCHAR szTop[] =       TEXT ("Message   Key   Char   ")
                                TEXT ("Repeat Scan Ext ALT
   static TCHAR szUnd[] =       TEXT ("_______  ___  ____   ")
                                TEXT ("______ ____ ___ ___ __

   static TCHAR * szFormat[2] = {
               TEXT ("%-13s %3d %-15s%c%6u %4d %3s %3s %4s
               TEXT ("%-13s   0x%04X%1s%c %6u %4d %3s %3s %

   static TCHAR * szYes  = TEXT ("Yes") ;
   static TCHAR * szNo          = TEXT ("No") ;
```

```
static TCHAR * szDown= TEXT ("Down") ;
static TCHAR * szUp          = TEXT ("Up") ;

static TCHAR * szMessage [] = {
                    TEXT ("WM_KEYDOWN"),    TEXT ("WM_KEYUP")
                    TEXT ("WM_CHAR"),       TEXT ("WM_DEADCHAR
                    TEXT ("WM_SYSKEYDOWN"), TEXT ("WM_SYSKE
                    TEXT ("WM_SYSCHAR"),    TEXT ("WM_SYSDEAD
HDC         hdc ;
int         i, iType ;
PAINTSTRUCT  ps ;
TCHAR       szBuffer[128], szKeyName [32] ;
TEXTMETRIC   tm ;

switch (message)
{
case   WM_INPUTLANGCHANGE:
       dwCharSet = wParam ;
   // fall through
```

```
case WM_CREATE:
case WM_DISPLAYCHANGE:
        // Get maximum size of client area
    cxClientMax = GetSystemMetrics (SM_CXMAXIMIZED) ;
    cyClientMax = GetSystemMetrics (SM_CYMAXIMIZED) ;

        // Get character size for fixed-pitch font
    hdc = GetDC (hwnd) ;
    SelectObject (hdc, CreateFont (0, 0, 0, 0, 0, 0, 0, 0,
                                   dwCharSet, 0, 0, 0, FIXED_PITCH,
    GetTextMetrics (hdc, &tm) ;
    cxChar = tm.tmAveCharWidth ;
    cyChar = tm.tmHeight ;

    DeleteObject (SelectObject (hdc, GetStockObject (SYSTE
    ReleaseDC (hwnd, hdc) ;

        // Allocate memory for display lines
    if (pmsg)
            free (pmsg) ;
```

```
        cLinesMax = cyClientMax / cyChar ;
        pmsg = malloc (cLinesMax * sizeof (MSG)) ;
        cLines = 0 ;
             // fall through
case   WM_SIZE:
      if (message == WM_SIZE)
        {
                  cxClient            = LOWORD (lParam) ;
                  cyClient            = HIWORD (lParam) ;
        }
                  // Calculate scrolling rectangle

        rectScroll.left       = 0 ;
        rectScroll.right      = cxClient ;
        rectScroll.top        = cyChar ;
        rectScroll.bottom     = cyChar * (cyClient / cyChar) ;

        InvalidateRect (hwnd, NULL, TRUE) ;

        if (message == WM_INPUTLANGCHANGE)
```

```
                return TRUE ;
        return 0 ;

case WM_KEYDOWN:
case WM_KEYUP:
case WM_CHAR:
case WM_DEADCHAR:
case WM_SYSKEYDOWN:
case WM_SYSKEYUP:
case WM_SYSCHAR:
case WM_SYSDEADCHAR:
            // Rearrange storage array
        for (i = cLinesMax - 1 ; i > 0 ; i--)
        {
                pmsg[i] = pmsg[i - 1] ;
       }
                // Store new message
        pmsg[0].hwnd = hwnd ;
        pmsg[0].message = message ;
```

```
        pmsg[0].wParam = wParam ;
        pmsg[0].lParam = lParam ;

        cLines = min (cLines + 1, cLinesMax) ;
                    // Scroll up the display
        ScrollWindow (hwnd, 0, -cyChar, &rectScroll, &rectScroll)
        break ;       // ie, call DefWindowProc so Sys messages wo

case   WM_PAINT:
        hdc = BeginPaint (hwnd, &ps) ;

        SelectObject (hdc, CreateFont (0, 0, 0, 0, 0, 0, 0, 0,
                                                dwCharSet, 0, 0, 0, FIXE
        SetBkMode (hdc, TRANSPARENT) ;
TextOut (hdc, 0, 0, szTop, lstrlen (szTop)) ;
TextOut (hdc, 0, 0, szUnd, lstrlen (szUnd)) ;

for (i = 0 ; i < min (cLines, cyClient / cyChar - 1) ; i++)
{
        iType =pmsg[i].message == WM_CHAR ||
```

```
                pmsg[i].message == WM_SYSCHAR ||
                pmsg[i].message == WM_DEADCHAR ||
                pmsg[i].message == WM_SYSDEADCHAR ;

        GetKeyNameText (pmsg[i].lParam, szKeyName,
                                    sizeof (szKeyName) / sizeof (TCH

        TextOut (hdc, 0, (cyClient / cyChar - 1 - i) * cyChar, szBuf
                wsprintf (    szBuffer, szFormat [iType],
                                        szMessage [pmsg[i].messag
WM_KEYFIRST],
                                        pmsg[i].wParam,
                                        (PTSTR) (iType ? TEXT (" ") :
                                       (TCHAR) (iType ? pmsg[i].wP
                                        LOWORD (pmsg[i].lParam),
                                        HIWORD (pmsg[i].lParam) &
                                      0x01000000 & pmsg[i].lPara
                                        0x20000000 & pmsg[i].lPar
                                        0x40000000 & pmsg[i].lPar
```

```
                                        0x80000000 & pmsg[i].lPar
        }
        DeleteObject (SelectObject (hdc, GetStockObject (SYSTE
        EndPaint (hwnd, &ps) ;
        return 0 ;

   case   WM_DESTROY:
        PostQuitMessage (0) ;
        return 0 ;
   }
   return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

KEYVIEW2KEYVIEW2KEYVIEW2
IDKEYVIEW2

/Platform SDK/Windows
/Platform SDK/Windows Base Services/General Library/String Manipulation

WindowsWindows

- CreateCaret
- SetCaretPos
- ShowCaret
- HideCaret
- DestroyCaret

GetCaretPosGetCaretBlinkTimeSetCaretBlinkTime

WindowsWindows

WM_CREATEWM_DESTROY

WM_SETFOCUSWM_KILLFOCUSWM_SETFOCUS
WM_KILLFOCUSWM_KILLFOCUSWM_SETFOCUS
WM_SETFOCUSWM_KILLFOCUS

WM_SETFOCUSCreateCaretWM_KILLFOCUSDestroyCaret

CreateCaretShowCaretWM_PAINTHideCaret
ShowCaretHideCaretHideCaretShowCaretShowCaret

TYPER

6-5TYPERTYPERIEscape

6-5 TYPER

```
TYPER.C

/*---------------------------------------------------------------------
```

```
 TYPER.C --      Typing Program

                          (c) Charles Petzold, 1998

----------------------------------------------------------------------*/

#include <windows.h>

#define BUFFER(x,y) *(pBuffer + y * cxBuffer + x)

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
          PSTR szCmdLine, int iCmdShow)
{
   static TCHAR  szAppName[] = TEXT ("Typer") ;
   HWND                hwnd ;
   MSG                 msg ;
   WNDCLASS            wndclass ;

   wndclass.style                    = CS_HREDRAW | CS_VREDI
   wndclass.lpfnWndProc              = WndProc ;
   wndclass.cbClsExtra               = 0 ;
```

```
wndclass.cbWndExtra                    = 0 ;
wndclass.hInstance                     = hInstance ;
wndclass.hIcon                         = LoadIcon (NULL, IDI_APPI
wndclass.hCursor                       = LoadCursor (NULL, IDC_
wndclass.hbrBackground                 = (HBRUSH) GetStockObjec
wndclass.lpszMenuName                  = NULL ;
wndclass.lpszClassName                 = szAppName ;
if (!RegisterClass (&wndclass))
{
MessageBox (  NULL, TEXT ("This program requires Windows N
                    szAppName, MB_ICONERROR) ;
return 0 ;
}
hwnd = CreateWindow (szAppName, TEXT ("Typing Program")
                                       WS_OVERLAPPEDWINDOW,
                                       CW_USEDEFAULT, CW_USED
                                       CW_USEDEFAULT, CW_USED
                                      NULL, NULL, hInstance, NUL
```

```
   ShowWindow (hwnd, iCmdShow) ;
   UpdateWindow (hwnd) ;

   while (GetMessage (&msg, NULL, 0, 0))
   {
               TranslateMessage (&msg) ;
               DispatchMessage (&msg) ;
   }
   return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
   static DWORD        dwCharSet = DEFAULT_CHARSET ;
   static int          cxChar, cyChar, cxClient, cyClient, cxBuffer, c
                           xCaret, yCaret ;
   static TCHAR *pBuffer = NULL ;
   HDC                 hdc ;
```

```
int                     x, y, i ;
PAINTSTRUCT             ps ;
TEXTMETRIC              tm ;

switch (message)
{
case   WM_INPUTLANGCHANGE:
       dwCharSet = wParam ;
         // fall through
case   WM_CREATE:
       hdc = GetDC (hwnd) ;
       SelectObject (hdc, CreateFont (0, 0, 0, 0, 0, 0, 0, 0,
                                      dwCharSet, 0, 0, 0, FIXE

       GetTextMetrics (hdc, &tm) ;
       cxChar = tm.tmAveCharWidth ;
       cyChar = tm.tmHeight ;

       DeleteObject (SelectObject (hdc, GetStockObject (SYSTE
```

```
        ReleaseDC (hwnd, hdc) ;
        // fall through
case   WM_SIZE:
             // obtain window size in pixels

        if (message == WM_SIZE)
        {
                    cxClient = LOWORD (lParam) ;
                    cyClient = HIWORD (lParam) ;
        }
             // calculate window size in characters

        cxBuffer = max (1, cxClient / cxChar) ;
        cyBuffer = max (1, cyClient / cyChar) ;

             // allocate memory for buffer and clear it

        if (pBuffer != NULL)
                    free (pBuffer) ;
```

```
        pBuffer = (TCHAR *) malloc (cxBuffer * cyBuffer * sizeof

        for (y = 0 ; y < cyBuffer ; y++)
                for (x = 0 ; x < cxBuffer ; x++)
                        BUFFER(x,y) = ' ' ;

             // set caret to upper left corner

        xCaret = 0 ;
        yCaret = 0 ;

        if (hwnd == GetFocus ())
                SetCaretPos (xCaret * cxChar, yCaret * cyChar)

        InvalidateRect (hwnd, NULL, TRUE) ;
        return 0 ;

case   WM_SETFOCUS:
                // create and show the caret
```

```
        CreateCaret (hwnd, NULL, cxChar, cyChar) ;
        SetCaretPos (xCaret * cxChar, yCaret * cyChar) ;
       ShowCaret (hwnd) ;
        return 0 ;

case WM_KILLFOCUS:
                // hide and destroy the caret
        HideCaret (hwnd) ;
        DestroyCaret () ;
        return 0 ;

case   WM_KEYDOWN:
        switch (wParam)
        {
        case VK_HOME:
                xCaret = 0 ;
                break ;

                case   VK_END:
```

```
                    xCaret = cxBuffer - 1 ;
                    break ;

            case   VK_PRIOR:
                    yCaret = 0 ;
                    break ;

            case   VK_NEXT:
                    yCaret = cyBuffer - 1 ;
                    break ;

            case   VK_LEFT:
                    xCaret = max (xCaret - 1, 0) ;
                    break ;

        case   VK_RIGHT:
            xCaret = min (xCaret + 1, cxBuffer - 1) ;
            break ;
```

```
            case   VK_UP:
                  yCaret = max (yCaret - 1, 0) ;
                   break ;

            case   VK_DOWN:
                   yCaret = min (yCaret + 1, cyBuffer - 1) ;
                   break ;

            case   VK_DELETE:
                   for (x = xCaret ; x < cxBuffer - 1 ; x++)
                              BUFFER (x, yCaret) = BUFFER (x + 1,

                   BUFFER (cxBuffer - 1, yCaret) = ' ' ;

                   HideCaret (hwnd) ;
                   hdc = GetDC (hwnd) ;

                   SelectObject (hdc, CreateFont (0, 0, 0, 0, 0, 0, 0
```

```
                                        dwCharSet, 0, 0, 0,FIXE
                TextOut (hdc, xCaret * cxChar, yCaret * cyChar,
                                        & BUFFER (xCaret, yCaret),
                                        cxBuffer - xCaret) ;

                 DeleteObject (SelectObject (hdc, GetStockObje
                 ReleaseDC (hwnd, hdc) ;
                ShowCaret (hwnd) ;
                break ;
        }
        SetCaretPos (xCaret * cxChar, yCaret * cyChar) ;
        return 0 ;

case   WM_CHAR:
        for (i = 0 ; i < (int) LOWORD (lParam) ; i++)
        {
             switch (wParam)
                   {
             case '\b':                                 // backspace
```

```
                    if (xCaret > 0)
                         {
                              xCaret-- ;
                             SendMessage (hwnd, WM_KEYDO
                         }
                    break ;

case '\t':                                   // tab
               do
                    {
                         SendMessage (hwnd, WM_CHAR, ' ', 1
                    }
               while (xCaret % 8 != 0) ;
               break ;
 case '\n':                                    // line feed
               if (++yCaret == cyBuffer)
                         yCaret = 0 ;
               break ;
```

```
        case '\r':                                  // carriage return
                    xCaret = 0 ;

                    if (++yCaret == cyBuffer)
                                yCaret = 0 ;
                    break ;

        case '\x1B':                                // escape
                    for (y = 0 ; y < cyBuffer ; y++)
                    for (x = 0 ; x < cxBuffer ; x++)
                                BUFFER (x, y) = ' ' ;

                    xCaret = 0 ;
                    yCaret = 0 ;

                    InvalidateRect (hwnd, NULL, FALSE) ;
                    break ;
```

```
default:                                        // character cod
     BUFFER (xCaret, yCaret) = (TCHAR) wParam ;


     HideCaret (hwnd) ;
     hdc = GetDC (hwnd) ;


     SelectObject (hdc, CreateFont (0, 0, 0, 0, 0, 0, 0, 0,
                                    dwCharSet, 0, 0, 0, FIXED_P
     TextOut (hdc, xCaret * cxChar, yCaret * cyChar,
                         & BUFFER (xCaret, yCaret), 1) ;
     DeleteObject (
          SelectObject (hdc, GetStockObject (SYSTEM_FO
     ReleaseDC (hwnd, hdc) ;
     ShowCaret (hwnd) ;

    if (++xCaret == cxBuffer)
          {
               xCaret = 0 ;
               if (++yCaret == cyBuffer)
```

```
                                    yCaret = 0 ;

                    }

               break ;

          }

}


          SetCaretPos (xCaret * cxChar, yCaret * cyChar) ;

          return 0 ;


case   WM_PAINT:

          hdc = BeginPaint (hwnd, &ps) ;


          SelectObject (hdc, CreateFont (0, 0, 0, 0, 0, 0, 0, 0,

                                         dwCharSet, 0, 0, 0, FIXED_PITCH,

          for (y = 0 ; y < cyBuffer ; y++)

               TextOut (hdc, 0, y * cyChar, & BUFFER(0,y), cxBuffer

          DeleteObject (SelectObject (hdc, GetStockObject (SYSTE

          EndPaint (hwnd, &ps) ;
```

```
        return 0 ;


   case   WM_DESTROY:
        PostQuitMessage (0) ;
        return 0 ;
   }
   return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

TYPERWM_CREATEWM_KEYDOWNWM_CHAR
WM_PAINTGetStockObjectSelectObject

WM_SIZETYPERcxBuffercyBuffermalloccxBuffercyBuffer
sizeofTCHAR128Unicode

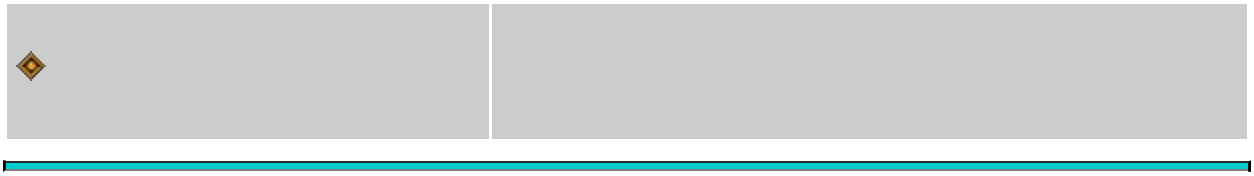
xCaretyCaretWM_SETFOCUSTYPERCreateCaretSetCaretPos
ShowCaretWM_KILLFOCUSTYPERHideCaretDestroyCaret

WM_KEYDOWNHomeEndPage
TYPER

WM_CHARBackspaceTabLinefeedCtrl-EnterEnterEscape
WM_CHARlParamWM_KEYDOWNBackspaceTab
SendMessageBackspaceDeleteTab

WM_PAINTTYPERDeleteWM_KEYDOWNWM_CHAR
TYPER

TYPERKEYVIEW2WindowsTYPER



PC

WindowsWindowsappletWindows

WindowsWindowsWindows

Windows 98Windows

GetSystemMetrics

```
fMouse = GetSystemMetrics (SM_MOUSEPRESENT) ;
```

fMouseTRUE00Windows
Windows NT

```
cButtons = GetSystemMetrics (SM_CMOUSEBUTTONS) ;
```

0Windows 982

WindowsGetSystemMetricsSM_SWAPBUTTON

WindowsSystemParametersInfo

WindowsWindows

WindowsIDC_ARROWWINUSER.HIDC_CROSS
BLOKOUTIDC_WAIT

```
wndclass.hCursor = LoadCursor (NULL, IDC_ARROW) ;
```

- **Clicking**
- **Double-clicking**
- **Dragging**

WindowsLBUTTONMBUTTONRBUTTON

(Mouse)

mousemicemicemouseAmerican
Dictionary of the English Language

Wired stylePrinciples of English Usage in the Digital AgeHardWired,
1996mouse1964Doug Engelba

Microsoft Manual of Style for Technical Publicationsmicemouse
mouse devicesmouse"People
much as keyboard""Pople use mice almost as much as keyboards"

WindowsWindows2111
Windows

WM_MOUSEMOVE

		()
WM_LBUTTONDOWN	WM_LBUTTONUP	WM_LBUTTONDBLCLK
WM_MBUTTONDOWN	WM_MBUTTONUP	WM_MBUTTONDBLCLK
WM_RBUTTONDOWN	WM_RBUTTONUP	WM_RBUTTONDBLCLK

MBUTTONRBUTTONDBLCLK

lParamxyLOWORDHIWORD

```
x = LOWORD (lParam) ;

y = HIWORD (lParam) ;
```

wParamShiftCtrlWINUSER.HwParamMK

MK_LBUTTON	
MK_MBUTTON	
MK_RBUTTON	

MK_SHIFT	Shift
MK_CONTROL	Ctrl

WM_LBUTTONDOWN

```
wparam & MK_SHIFT
```

TRUE0Shift

WindowsWM_MOUSEMOVEWM_MOUSEMOVE
WindowsWM_MOUSEMOVECONNECTWM_MOUSEMOVE

WindowsWM_LBUTTONDOWNWM_LBUTTONDOWN
WM_LBUTTONDOWNWM_LBUTTONUP
WM_LBUTTONDOWNWM_LBUTTONUP

-
- Windows

7-1CONNECTWindows

7-1　CONNECT

```
CONNECT.C

/*------------------------------------------------------------------------
```

```
   CONNECT.C -- Connect-the-Dots Mouse Demo Program

                        (c) Charles Petzold, 1998

----------------------------------------------------------------------*/

#include <windows.h>

#define MAXPOINTS 1000

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns

                                PSTR szCmdLine, int iCmdShow)

{

   static TCHAR  szAppName[] = TEXT ("Connect") ;

   HWND                    hwnd ;

   MSG                     msg ;

   WNDCLASS                wndclass ;


   wndclass.style                      = CS_HREDRAW | CS_VRED

   wndclass.lpfnWndProc             = WndProc ;

   wndclass.cbClsExtra                    = 0 ;

   wndclass.cbWndExtra                    = 0 ;
```

```
wndclass.hInstance                      = hInstance ;
wndclass.hIcon                         = LoadIcon (NULL, IDI_APPI
wndclass.hCursor                        = LoadCursor (NULL, IDC_
wndclass.hbrBackground                  = (HBRUSH) GetStockO
wndclass.lpszMenuName                   = NULL ;
wndclass.lpszClassName = szAppName ;
if (!RegisterClass (&wndclass))
{
     MessageBox (NULL, TEXT ("Program requires Windows NT
                              szAppName, MB_ICONERROR) ;
     return 0 ;
}

hwnd = CreateWindow (szAppName, TEXT ("Connect-the-Poin
                                    WS_OVERLAPPEDWINDOW,
                                    CW_USEDEFAULT, CW_USED
                                    CW_USEDEFAULT, CW_USED
                                    NULL, NULL, hInstance, NUL
```

```
   ShowWindow (hwnd, iCmdShow) ;
   UpdateWindow (hwnd) ;

   while (GetMessage (&msg, NULL, 0, 0))
   {
         TranslateMessage (&msg) ;
         DispatchMessage (&msg) ;
   }
return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
   static POINT pt[MAXPOINTS] ;
   static int          iCount ;
   HDC                       hdc ;
   in                        i, j ;
   PAINTSTRUCT  ps ;
```

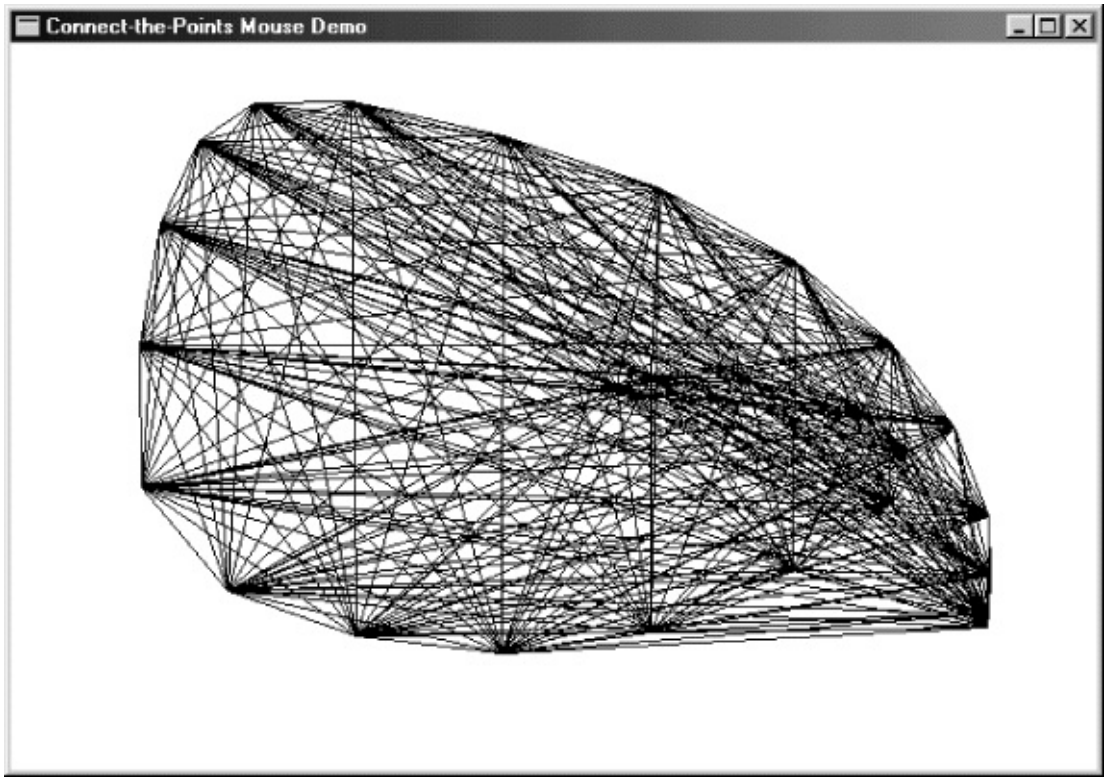
```
switch (message)
{
case   WM_LBUTTONDOWN:
       iCount = 0 ;
       InvalidateRect (hwnd, NULL, TRUE) ;
       return 0 ;

case   WM_MOUSEMOVE:
       if (wParam & MK_LBUTTON && iCount < 1000)
       {
                  pt[iCount  ].x = LOWORD (lParam) ;
                  pt[iCount++].y = HIWORD (lParam) ;

                  hdc = GetDC (hwnd) ;
                  SetPixel (hdc, LOWORD (lParam), HIWORD (lPar
                  ReleaseDC (hwnd, hdc) ;
       }
                        return 0 ;
```

```
        case   WM_LBUTTONUP:
             InvalidateRect (hwnd, NULL, FALSE) ;
             return 0 ;

        case   WM_PAINT:
             hdc = BeginPaint (hwnd, &ps) ;

             SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
             ShowCursor (TRUE) ;

             for (i = 0 ; i < iCount - 1 ; i++)
                          for (j = i + 1 ; j < iCount ; j++)
                           {
                        MoveToEx (hdc, pt[i].x, pt[i].y, NULL) ;
                        LineTo   (hdc, pt[j].x, pt[j].y) ;
          }

             ShowCursor (FALSE) ;
             SetCursor (LoadCursor (NULL, IDC_ARROW)) ;
```

```
            EndPaint (hwnd, &ps) ;
            return 0 ;

        case   WM_DESTROY:
            PostQuitMessage (0) ;
            return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

CONNECT

- **WM_LBUTTONDOWN**CONNECT
- **WM_MOUSEMOVE**CONNECT
- **WM_LBUTTONUPCONNECT** 7-1



7-1 CONNECT

CONNECTCONNECT

CONNECT(GDI) SetPixelWM_MOUSEMC
MoveToExLineTo

CONNECTWM_LBUTTONUPCONNECT

CONNECT1000PCONNECTP × (P -
Windows 98CONNECT

CONNECTWM_PAINTSetCursorCONNECTShowCursorTRUE
FALSE

Shift

CONNECTWM_MOUSEMOVEwParamMK_LBUTTONAND wParamShiftShiftCtrl

```
if (wParam & MK_SHIFT)
{
   if (wParam & MK_CONTROL)
         {
         //ShiftCtrl
   }
   else
   {
                  //Shift
         }
   {
   else
   {
         if (wParam & MK_CONTROL)
         {
                  //Ctrl
         }
```

```
        else
        {
                //ShiftCtrl
        }
}
```

Shift

```
caseWM_LBUTTONDOWN:
   if (!(wParam & MK_SHIFT))
   {
                //
                return 0 ;
   }
                // Fall through
case WM_RBUTTONDOWN:
   //
   return 0 ;
```

WindowsGetKeyState VK_LBUTTONVK_RBUTTON VK_MBUTTONVK_SHIFTVK_CONTROLShiftGetKeyStateShift GetKeyStateShiftGetKeyStateGetKeyState

```
while (GetKeyState (VK_LBUTTON) >= 0) ;  // WRONG !!!
```

GetKeyState

RegisterClassCS_DBLCLKS

```
wndclass.style = CS_HREDRAW | CS_VREDRAW | CS_DBLCLKS ;
```

CS_DBLCLKS

WM_LBUTTONDOWN

WM_LBUTTONUP

WM_LBUTTONDOWN

WM_LBUTTONUP

WindowsGetMessageTimeWM_LBUTTONDOWN

CS_DBLCLKS

WM_LBUTTONDOWN

WM_LBUTTONUP

WM_LBUTTONDBLCLK

WM_LBUTTONUP

WM_LBUTTONDBLCLKWM_LBUTTONDOWN

WM_LBUTTONDBLCLKWindows

Windows ExplorerWindows

10Windows

DefWindowProcWindowsWM_SYSKEYDOWN
WM_SYSKEYUPWM_SYSCHAR

NCWM_NCMOUSEMOVE7-2

7-2

WM_NCLBUTTONDOWN	WM_NCLBUTTONUP	WM_NCLBUTTONDB
WM_NCMBUTTONDOWN	WM_NCMBUTTONUP	WM_NCMBUTTONDI
WM_NCRBUTTONDOWN	WM_NCRBUTTONUP	WM_NCRBUTTONDB

wParamlParamwParamlParamwParamWINUSER.HHTHT

lParamwordxwordyxy0xy7-2

Windows

```
ScreenToClient (hwnd, &pt) ;

ClientToScreen (hwnd, &pt) ;
```

ptPOINTxy

2120WM_NCHITTESTlParamxywParam

WindowsDefWindowProcWindowsWM_NCHITTEST
WM_NCHITTESTDefWindowProcwParamwParam

HTCLIENT HTNOWHERE HTTRANSPARENT HTERROR	DefWindowProc

DefWindowProcWM_NCHITTESTHTCLIENTWindows

WM_SYSKEYDOWN

```
case WM_NCHITTEST:
  return (LRESULT) HTNOWHERE ;
```

WindowsWM_NCHITTESTWindowsWindows
WM_NCHITTESTDefWindowProcHTSYSMENUWindowswParam
HTSYSMENUWM_NCLBUTTONDBLCLK

DefWindowProcDefWindowProcwParamHTSYSMENU
WM_NCLBUTTONDBLCLKwParamSC_CLOSEWM_SYSCOMMAND
WM_SYSCOMMANDCloseDefWindowProc
DefWindowProcWM_CLOSE

WM_CLOSEDefWindowProcDestroyWindowWM_CLOSE
DestroyWindowWM_DESTROYWM_DESTROY

```
caseWM_DESTROY:
```

```
    PostQuitMessage (0) ;

    return 0 ;
```

PostQuitMessageWindowsWM_QUITGetMessage0

Windows ExplorerWindows

DefWindowProcWM_NCHITTESTxylParam

list view

cxClientcyClientcxColWidthcyChar

```
iNumInCol = cyClient / cyChar ;
```

lParamcxMousecyMouse

```
iColumn = cxMouse / cxColWidth ;
```

```
iFromTop = cyMouse / cyChar ;
```

szFileNames

```
iIndex = iColumn * iNumInCol + iFromTop ;
```

iIndex

7-2CHECKER15×525XX

7-2　CHECKER1

```
CHECKER1.C
/*------------------------------------------------------------------------
  CHECKER1.C --      Mouse Hit-Test Demo Program No. 1
                                                (c) Charles Petzold, 1998
------------------------------------------------------------------------*/
#include <windows.h>
#define DIVISIONS 5
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
{      static TCHAR      szAppName[] = TEXT ("Checker1") ;
          HWND         hwnd ;
          MSG          msg ;
          WNDCLASS     wndclass ;
```

```
     wndclass.style                  = CS_HREDRAW | CS_V
     wndclass.lpfnWndProc              = WndProc ;
     wndclass.cbClsExtra               = 0 ;
     wndclass.cbWndExtra               = 0 ;
     wndclass.hInstance                = hInstance ;
     wndclass.hIcon                    = LoadIcon (NULL, IDI_
     wndclass.hCursor                  = LoadCursor (NULL,
     wndclass.hbrBackground            = (HBRUSH) GetStockO
     wndclass.lpszMenuName             = NULL ;
     wndclass.lpszClassName            = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (  NULL, TEXT ("Program requires Windo
                                          szAppName, MB_IC
          return 0 ;
     }
     hwnd = CreateWindow (szAppName, TEXT ("Checker1 Mo
               WS_OVERLAPPEDWINDOW,
```

```
                CW_USEDEFAULT, CW_USEDEFAULT,
                CW_USEDEFAULT, CW_USEDEFAULT,
               NULL, NULL, hInstance, NULL) ;

  ShowWindow (hwnd, iCmdShow) ;
        UpdateWindow (hwnd) ;

        while (GetMessage (&msg, NULL, 0, 0))
        {
                TranslateMessage (&msg) ;
                DispatchMessage (&msg) ;
        }
  return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
        static BOOL   fState[DIVISIONS][DIVISIONS] ;
        static int          cxBlock, cyBlock ;
```

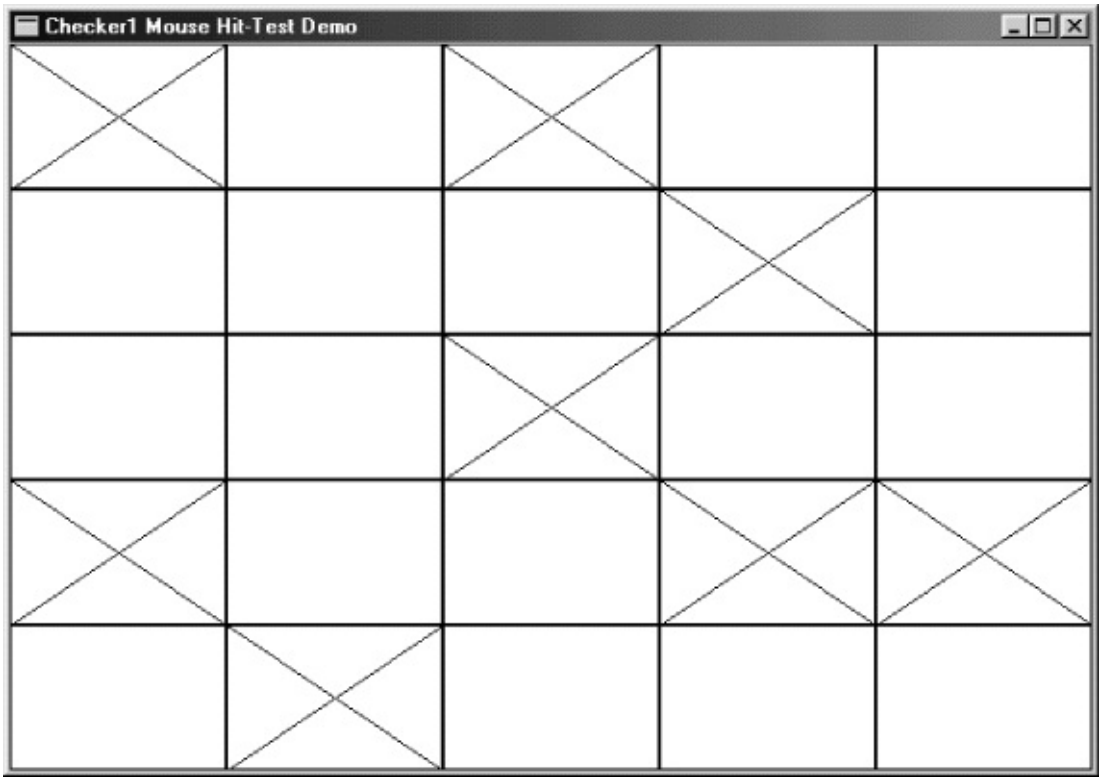
```
     HDC                         hdc ;
     int                        x, y ;
     PAINTSTRUCT          ps ;
    RECT                         rect ;

     switch (message)
{
     case   WM_SIZE :
          cxBlock = LOWORD (lParam) / DIVISIONS ;
          cyBlock = HIWORD (lParam) / DIVISIONS ;
          return 0 ;

     case WM_LBUTTONDOWN :
          x = LOWORD (lParam) / cxBlock ;
          y = HIWORD (lParam) / cyBlock ;

          if (x < DIVISIONS && y < DIVISIONS)
          {
```

```
                    fState [x][y] ^= 1 ;
                    rect.left      = x * cxBlock ;
                    rect.top       = y * cyBlock ;
                    rect.right     = (x + 1) * cxBloc
                    rect.bottom    = (y + 1) * cyBlock

                    InvalidateRect (hwnd, &rect, FALSE) ;
          }
          else
                    MessageBeep (0) ;
          return 0 ;

     case   WM_PAINT :
          hdc = BeginPaint (hwnd, &ps) ;

          for (x = 0 ; x < DIVISIONS ; x++)
          for (y = 0 ; y < DIVISIONS ; y++)
          {
                    Rectangle (hdc, x * cxBlock, y * cyBlock,
```

```
                    (x + 1) * cxBlock, (y + 1) * cyBlock) ;

               if (fState [x][y])
               {
               MoveToEx      (hdc, x * cxBlock,  y * cyBlock, NU
               LineTo(hdc, (x+1) * cxBlock, (y+1) * cyBlock) ;
          MoveToEx (hdc, x * cxBlock, (y+1) * cyBlock, NULL)
          LineTo (hdc, (x+1) * cxBlock, y  * cyBlock) ;
                    }
          }
          EndPaint (hwnd,&ps);
     return 0 ;

    case   WM_DESTROY :
          PostQuitMessage (0) ;
          return 0 ;
    }
    return DefWindowProc (hwnd, message, wParam, lParam)
```

```
}
```

7-3CHECKER125cxBlockcyBlockWM_LBUTTONDOWN
fStateWM_PAINT



7-3 CHECKER1

5CHECKER1MessageBeep

CHECKER1WM_PAINTGDIRectanglefStateCHECKER1
MoveToExLineToWM_PAINTCHECKER1RECT
WM_LBUTTONDOWNIntersectRectps.rcPaint

CHECKER1 SYS

WindowsWindows0-1

```
ShowCursor (TRUE) ;
```

```
ShowCursor (FALSE) ;
```

ShowCursorShowCursor

Windows

```
GetCursorPos (&pt) ;
```

ptPOINTxyPOINT

```
SetCursorPos (x, y) ;
```

xyhwndScreenToClientClientToScreen

GetCursorPos lParamGetCursorPoslPa

SpacebarEnter

WM_KEYDOWNlParam

CHECKER

7-3CHECKER2CHECKER125Home
SpacebarEnterX

7-3 CHECKER2

```
CHECKER2.C

/*------------------------------------------------------------------------
  CHECKER2.C --   Mouse Hit-Test Demo Program No. 2
                                (c) Charles Petzold, 1998
------------------------------------------------------------------------*/

#include <windows.h>

#define DIVISIONS 5

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                            PSTR szCmdLine, int iCmdShow)
{
        static TCHAR szAppName[] = TEXT ("Checker2") ;
        HWND                 hwnd ;
        MSG                  msg ;
        WNDCLASS             wndclass ;

        wndclass.style                          = CS_HREDRAW |
```

```
        wndclass.lpfnWndProc                    = WndProc ;
        wndclass.cbClsExtra                     = 0 ;
        wndclass.cbWndExtra                     = 0 ;
        wndclass.hInstance                      = hInstance ;
        wndclass.hIcon                          = LoadIcon (NULL
wndclass.hCursor                                = LoadCursor (NU
        wndclass.hbrBackground                  = (HBRUSH) GetS
        wndclass.lpszMenuName                   = NULL ;
        wndclass.lpszClassName                  = szAppName ;

 if (!RegisterClass (&wndclass))
        {
             MessageBox (  NULL, TEXT ("Program requires Windo
                                     szAppName, MB_ICONERROR) ;
            return 0 ;
        }
        hwnd = CreateWindow ( szAppName, TEXT ("Checker2 M
                WS_OVERLAPPEDWINDOW,
                CW_USEDEFAULT, CW_USEDEFAULT,
```

```
                CW_USEDEFAULT, CW_USEDEFAULT,
                NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
     static BOOL   fState[DIVISIONS][DIVISIONS] ;
     static int    cxBlock, cyBlock ;
     HDC           hdc ;
```

```
int                         x, y ;
PAINTSTRUCT                 ps ;
POINT                       point ;
RECT                        rect ;

switch (message)
{
case   WM_SIZE :
     cxBlock = LOWORD (lParam) / DIVISIONS ;
     cyBlock = HIWORD (lParam) / DIVISIONS ;
     return 0 ;

case   WM_SETFOCUS :
     ShowCursor (TRUE) ;
     return 0 ;

case   WM_KILLFOCUS :
     ShowCursor (FALSE) ;
```

```
            return 0 ;

        case   WM_KEYDOWN :
            GetCursorPos (&point) ;
            ScreenToClient (hwnd, &point) ;
            x = max (0, min (DIVISIONS - 1, point.x / cxBlock)) ;
            y = max (0, min (DIVISIONS - 1, point.y / cyBlock)) ;

            switch (wParam)
            {
                case VK_UP :
                                y-- ;
                                break ;

                case   VK_DOWN :
                                y++ ;
                                break ;

                case   VK_LEFT :
```

```
                    x-- ;
                    break ;

            case   VK_RIGHT :
                    x++ ;
                    break ;

            case   VK_HOME :
                    x = y = 0 ;
                    break ;

            case   VK_END :
                    x = y = DIVISIONS - 1 ;
                    break ;

            case   VK_RETURN :
            case   VK_SPACE :
                    SendMessage (hwnd, WM_LBUTT
```

```
                    MAKELONG (x * cxBlock, y * cyBlock)) ;

                              break ;

          }

           x = (x + DIVISIONS) % DIVISIONS ;

                y = (y + DIVISIONS) % DIVISIONS ;


                point.x = x * cxBlock + cxBlock / 2 ;

                point.y = y * cyBlock + cyBlock / 2 ;


                ClientToScreen (hwnd, &point) ;

                SetCursorPos (point.x, point.y) ;

                return 0 ;

     case   WM_LBUTTONDOWN :

      x = LOWORD (lParam) / cxBlock ;

      y = HIWORD (lParam) / cyBlock ;


     if (x < DIVISIONS && y < DIVISIONS)

          {

                              fState[x][y] ^= 1 ;
```

```
                                rect.left     = x * cxBlock ;
                                rect.top      = y * cyBlock ;
                                rect.right    = (x + 1) * cxBlock ;
                                rect.bottom   = (y + 1) * cyBlock

                                InvalidateRect (hwnd, &rect, FALS
     }
            else
                                MessageBeep (0) ;
            return 0 ;
      case   WM_PAINT :
       hdc = BeginPaint (hwnd, &ps) ;

          for (x = 0 ; x < DIVISIONS ; x++)
            for (y = 0 ; y < DIVISIONS ; y++)
            {
                        Rectangle (hdc, x * cxBlock, y * cyBlock,
```

```
                    (x + 1) * cxBlock, (y + 1) * cyBlock) ;


                    if (fState [x][y])
                    {
            MoveToEx (hdc,  x   *cxBlock,  y   *cyBlock, NULL) ;
                      LineTo   (hdc, (x+1)*cxBlock, (y+1)*cyBloc
                      MoveToEx (hdc,  x   *cxBlock, (y+1)*cyBlo
            LineTo   (hdc, (x+1)*cxBlock,  y   *cyBlock) ;
                    }
           }
           EndPaint (hwnd, &ps) ;
           return 0 ;

  case   WM_DESTROY :
           PostQuitMessage (0) ;
           return 0 ;
 }
       return DefWindowProc (hwnd, message, wParam, lParam)
}
```

CHECKER2WM_KEYDOWNGetCursorPosScreenToClient
xy5×5xyminmax04

CHECKER2xyEnterSpacebarCHECKER2SendMessage
WM_LBUTTONDOWN SYSMETSWM_KEYI
ClientToScreenSetCursorPos

Windows

lParam

CHECKER

CHECKER

7-4CHECKER325

7-4 CHECKER3

```
CHECKER3.C

/*------------------------------------------------------------------------
  CHECKER3.C --   Mouse Hit-Test Demo Program No. 3
                                        (c) Charles Petzold, 1998
------------------------------------------------------------------------*/

#include <windows.h>

#define DIVISIONS 5
```

```
LRESULT CALLBACK WndProc   (HWND, UINT, WPARAM, LPARAM)
LRESULT CALLBACK ChildWndProc (HWND, UINT, WPARAM, LPAR
TCHAR szChildClass[]                                = TEXT ("Checker3
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                          PSTR szCmdLine, int iCmdSh
{
        static TCHAR  szAppName[]   = TEXT ("Checker3") ;
        HWND                        hwnd ;
        MSG                         msg ;
        WNDCLASS                    wndclass ;

        wndclass.style                          = CS_HREDRAW |
        wndclass.lpfnWndProc                      = WndProc ;
        wndclass.cbClsExtra                     = 0 ;
        wndclass.cbWndExtra                       = 0 ;
        wndclass.hInstance                      = hInstance ;
        wndclass.hIcon                          = LoadIcon (NULL
        wndclass.hCursor                        = LoadCursor (N
```

```
wndclass.hbrBackground                  = (HBRUSH) GetStockO
     wndclass.lpszMenuName                    = NULL ;
     wndclass.lpszClassName = szAppName ;
     if (!RegisterClass (&wndclass))
     {
      MessageBox (  NULL, TEXT ("Program requires Windows N
                                                szAppName, MB_IC
    return 0 ;
     }

wndclass.lpfnWndProc                    = ChildWndProc ;
     wndclass.cbWndExtra                = sizeof (long) ;
     wndclass.hIcon                   = NULL ;
     wndclass.lpszClassName             = szChildClass ;

   RegisterClass (&wndclass) ;
     hwnd = CreateWindow (szAppName, TEXT ("Checker3 Mo
               WS_OVERLAPPEDWINDOW,
              CW_USEDEFAULT, CW_USEDEFAULT,
```

```
                    CW_USEDEFAULT, CW_USEDEFAULT,
                    NULL, NULL, hInstance, NULL) ;

          ShowWindow (hwnd, iCmdShow) ;
          UpdateWindow (hwnd) ;

          while (GetMessage (&msg, NULL, 0, 0))
          {
                    TranslateMessage (&msg) ;
                    DispatchMessage (&msg) ;
          }
        return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
          static HWND   hwndChild[DIVISIONS][DIVISIONS] ;
          int                    cxBlock, cyBlock, x, y ;
```

```
     switch (message)
{
     case   WM_CREATE :
          for (x = 0 ; x < DIVISIONS ; x++)
                for (y = 0 ; y < DIVISIONS ; y++)
           hwndChild[x][y] = CreateWindow (szChildClass, NUL
           WS_CHILDWINDOW | WS_VISIBLE,
           0, 0, 0, 0,
          hwnd, (HMENU) (y << 8 | x),
           (HINSTANCE) GetWindowLong (hwnd, GWL_HINSTAN
                                                NULL) ;
          return 0 ;

     case   WM_SIZE :
          cxBlock = LOWORD (lParam) / DIVISIONS ;
          cyBlock = HIWORD (lParam) / DIVISIONS ;
          for (x = 0 ; x < DIVISIONS ; x++)
                          for (y = 0 ; y < DIVISIONS ; y++)
```

```
                                        MoveWindow (  hwndChild[x
                                    x * cxBlock, y * cyBlock,
                                      cxBlock, cyBlock, TRUE) ;
                return 0 ;

        case   WM_LBUTTONDOWN :
                MessageBeep (0) ;
                return 0 ;


        case   WM_DESTROY :
                PostQuitMessage (0) ;
                return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam)
}

LRESULT CALLBACK ChildWndProc (HWND hwnd, UINT message,
           WPARAM wParam, LPARAM lParam)
{
        HDC                                hdc ;
```

```
     PAINTSTRUCT                    ps ;
     RECT                           rect ;

     switch (message)
     {
     case   WM_CREATE :
          SetWindowLong (hwnd, 0, 0) ;       // on/off flag
          return 0 ;

     case   WM_LBUTTONDOWN :
          SetWindowLong (hwnd, 0, 1 ^ GetWindowLong (hwn
          InvalidateRect (hwnd, NULL, FALSE) ;
        return 0 ;

     case   WM_PAINT :
          hdc = BeginPaint (hwnd, &ps) ;

          GetClientRect (hwnd, &rect) ;
```

```
            Rectangle (hdc, 0, 0, rect.right, rect.bottom) ;


            if (GetWindowLong (hwnd, 0))
            {
                        MoveToEx (hdc, 0,         0, NULL) ;
                         LineTo   (hdc, rect.right, rect.bottom) ;
                         MoveToEx (hdc, 0,         rect.bottom, N
                         LineTo   (hdc, rect.right, 0) ;
            }

           EndPaint (hwnd, &ps) ;
            return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam)
}
```

CHECKER3WndProcChildWndProcWndProcChildWndProc25
CALLBACK

WindowsRegisterClassCHECKER3Checker3
Checker3_Child

CHECKER3WinMainCHECKER3wndclassChecker3_Child

- pfnWndProcChildWndProc
- cbWndExtra4sizeof (long)Windows4
- CHECKER3hIconNULL
- pszClassNameChecker3_Child

WinMainCreateWindowChecker3WndProcWM_CREATE
CreateWindow 2525Checker3_Child7-3WinMainCreateWindow25
WndProcCreateWindow

7-3

	Checker3	Checker3_Child
	Checker3...	NULL
	WS_OVERLAPPEDWINDOW	WS_CHILDWINDOW | WS_VISIBLE
	CW_USEDEFAULT	0
	CW_USEDEFAULT	0

	CW_USEDEFAULT	0
	CW_USEDEFAULT	0
	NULL	hwnd
/ ID	NULL	(HMENU) (y << 8 | x)
	hInstance	(HINSTANCE) GetWindowLong (hwnd, GWL_HINSTANCE)
	NULL	NULL

CHECKER3WndProcNULLCreateWindow

NULLIDID

CreateWindowWinMainWinMain
GetWindowLongWindowshInstanceGetWindowLonghInstance

hwndChildWndProcWM_SIZE25MoveWindowMoveWindow

ChildWndProc25ChildWndProchwndChildWndProcWM_CREATE
2525SetWindowWord0cbWndExtraChildWndProcX
XWM_LBUTTONDOWN0110WM_PAINT

CHECKER3C.EXECHECKER1CHECKER3CHECKER1
CHECKER3WM_LBUTTONDOWN

CHECKER3CHECKERCHECKER2Spacebar

WindowsSpacebarEnter

CHECKER4.C7-5

7-5　　CHECKER4

```
CHECKER4.C
/*---------------------------------------------------------------------
 CHECKER4.C --   Mouse Hit-Test Demo Program No. 4
                                (c) Charles Petzold, 1998
---------------------------------------------------------------------*/
#include <windows.h>
#define DIVISIONS 5

LRESULT CALLBACK WndProc   (HWND, UINT, WPARAM, LPARAM)
LRESULT CALLBACK ChildWndProc (HWND, UINT, WPARAM, LPAR
int   idFocus = 0 ;
TCHAR szChildClass[] = TEXT ("Checker4_Child") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                          PSTR szCmdLine, int iCmdSh
```

```
{
        static TCHAR szAppName[] = TEXT ("Checker4") ;
         HWND                        hwnd ;
         MSG                        msg ;
         WNDCLASS                    wndclass ;

         wndclass.style                          = CS_HREDRAW |
         wndclass.lpfnWndProc                      = WndProc ;
         wndclass.cbClsExtra                       = 0 ;
         wndclass.cbWndExtra                       = 0 ;
         wndclass.hInstance                        = hInstance ;
         wndclass.hIcon                    = LoadIcon (NULL, IDI_
   wndclass.hCursor                        = LoadCursor (NULL,
       wndclass.hbrBackground                  = (HBRUSH) GetSt
         wndclass.lpszMenuName                   = NULL ;
         wndclass.lpszClassName                  = szAppName ;

         if (!RegisterClass (&wndclass))
         {
```

```
        MessageBox (NULL, TEXT ("Program requires Window
                                                szAppName, MB_IC
        return 0 ;
    }

    wndclass.lpfnWndProc                    = ChildWndPr
    wndclass.cbWndExtra                     = sizeof (long
    wndclass.hIcon                          = NULL ;
    wndclass.lpszClassName                  = szChildClass ;

    RegisterClass (&wndclass) ;
    hwnd = CreateWindow (szAppName, TEXT ("Checker4 Mo
              WS_OVERLAPPEDWINDOW,
              CW_USEDEFAULT, CW_USEDEFAULT,
              CW_USEDEFAULT, CW_USEDEFAULT,
              NULL, NULL, hInstance, NULL) ;
    ShowWindow (hwnd, iCmdShow) ;
    UpdateWindow (hwnd) ;
```

```
        while (GetMessage (&msg, NULL, 0, 0))
        {
                TranslateMessage (&msg) ;
                 DispatchMessage (&msg) ;
        }
       return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
   static HWND   hwndChild[DIVISIONS][DIVISIONS] ;
        int                     cxBlock, cyBlock, x, y ;

        switch (message)
        {
        case WM_CREATE :
              for (x = 0 ; x < DIVISIONS ; x++)
                              for (y = 0 ; y < DIVISIONS ; y++)
```

```
          hwndChild[x][y] = CreateWindow (szChildClass, NULL,
                    WS_CHILDWINDOW | WS_VISIBLE,
                     0, 0, 0, 0,
                     hwnd, (HMENU) (y << 8 | x),
                      HINSTANCE) GetWindowLong (hwnd, GWL_HINST
                                        NULL) ;
               return 0 ;
          case   WM_SIZE :
               cxBlock = LOWORD (lParam) / DIVISIONS ;
               cyBlock = HIWORD (lParam) / DIVISIONS ;

               for (x = 0 ; x < DIVISIONS ; x++)
                              for (y = 0 ; y < DIVISIONS ; y++)
                         MoveWindow (  hwndChild[x][y],
                           x * cxBlock, y * cyBlock,
                             cxBlock, cyBlock, TRUE) ;
               return 0 ;

          case   WM_LBUTTONDOWN :
```

```
            MessageBeep (0) ;
            return 0 ;

            // On set-focus message, set focus to child window
        case   WM_SETFOCUS:
            SetFocus (GetDlgItem (hwnd, idFocus)) ;
            return 0 ;

         // On key-down message, possibly change the focus winc

        case   WM_KEYDOWN:
            x = idFocus & 0xFF ;
            y = idFocus >> 8 ;

            switch (wParam)
            {
    case VK_UP:          y-- ;                                  break
        case VK_DOWN:             y++ ;
        case VK_LEFT:             x-- ;                           break
    case VK_RIGHT:            x++ ;                  break ;
```

```
     case VK_HOME:                x = y = 0 ;                    brea
     case VK_END:                 x = y = DIVISIONS - 1 ;        bre
               default:                          return 0 ;

               }

               x = (x + DIVISIONS) % DIVISIONS ;

               y = (y + DIVISIONS) % DIVISIONS ;

               idFocus = y << 8 | x ;

               SetFocus (GetDlgItem (hwnd, idFocus)) ;

               return 0 ;

          case   WM_DESTROY :

               PostQuitMessage (0) ;

               return 0 ;

          }

          return DefWindowProc (hwnd, message, wParam, lParam)
}

LRESULT CALLBACK ChildWndProc (HWND hwnd, UINT message,
```

```
            WPARAM wParam, LPARAM lParam)
{
        HDC                             hdc ;
        PAINTSTRUCT                     ps ;
        RECT                             rect ;

  switch (message)
   {
        case   WM_CREATE :
             SetWindowLong (hwnd, 0, 0) ;       // on/off flag
             return 0 ;

        case   WM_KEYDOWN:
                        // Send most key presses to the parent win

             if (wParam != VK_RETURN && wParam != VK_SPACE)
             {
                  SendMessage (GetParent (hwnd), message, wPa
                  return 0 ;
```

```
            }
                        // For Return and Space, fall through to togg


        case   WM_LBUTTONDOWN :
            SetWindowLong (hwnd, 0, 1 ^ GetWindowLong (hwn
            SetFocus (hwnd) ;
            InvalidateRect (hwnd, NULL, FALSE) ;
           return 0 ;

            // For focus messages, invalidate the window for repa


        case   WM_SETFOCUS:
            idFocus = GetWindowLong (hwnd, GWL_ID) ;

            // Fall through

        case   WM_KILLFOCUS:
            InvalidateRect (hwnd, NULL, TRUE) ;
            return 0 ;
```

```
    case     WM_PAINT :
             hdc = BeginPaint (hwnd, &ps) ;

             GetClientRect (hwnd, &rect) ;
             Rectangle (hdc, 0, 0, rect.right, rect.bottom) ;

                        // Draw the "x" mark

            if (GetWindowLong (hwnd, 0))
             {
                        MoveToEx (hdc, 0,          0, NULL) ;
                        LineTo   (hdc, rect.right, rect.bottom) ;
                        MoveToEx (hdc, 0,          rect.bottom, NULL)
                        LineTo   (hdc, rect.right, 0) ;
             }
                        // Draw the "focus" rectangle

             if (hwnd == GetFocus ())
             {
```

```
                    rect.left   += rect.right / 10 ;
                    rect.right  -= rect.left ;
                    rect.top    += rect.bottom / 10 ;
                    rect.bottom -= rect.top ;

                    SelectObject (hdc, GetStockObject (NULL_B
                    SelectObject (hdc, CreatePen (PS_DASH, 0,
                    Rectangle (hdc, rect.left, rect.top, rect.righ
                    DeleteObject (SelectObject (hdc, GetStock(
               }

               EndPaint (hwnd, &ps) ;
               return 0 ;
          }
          return DefWindowProc (hwnd, message, wParam, lParam)
}
```

IDIDCreateWindowCHECKER3IDxyID

```
idChild = GetWindowLong (hwndChild, GWL_ID) ;
```

```
idChild = GetDlgCtrlID (hwndChild) ;
```

ID

```
hwndChild = GetDlgItem (hwndParent, idChild) ;
```

CHECKER4idFocusIDCHECKER4WM_SETFOCUS

```
SetFocus (GetDlgItem (hwnd, idFocus)) ;
```

ChildWndProcWM_SETFOCUSWM_KILLFOCUSWM_SETFOCUS idFocusIDWM_PAINTWM_PAINTPS_DASH

ChildWndProcWM_KEYDOWNSpacebarEnterWM_KEYDOWN WM_LBUTTONDOWN

CHECKER2xySetFocus

BLOKOUT17-6

7-6 BLOKOUT1

BLOKOUT1.C

```
/*------------------------------------------------------------------------
```

```
  BLOKOUT1.C --   Mouse Button Demo Program

                              (c) Charles Petzold, 1998

----------------------------------------------------------------------*/

#include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                         PSTR szCmdLine, int iCmdSh
{
          static TCHAR szAppName[] = TEXT ("BlokOut1") ;
          HWND                         hwnd ;
          MSG                          msg ;
          WNDCLASS                     wndclass ;

          wndclass.style                              = CS_HREDRAW |
          wndclass.lpfnWndProc                          = WndProc ;
          wndclass.cbClsExtra                         = 0 ;
          wndclass.cbWndExtra                           = 0 ;
          wndclass.hInstance                          = hInstance ;
```

```
        wndclass.hIcon                          = LoadIcon (NULL
      wndclass.hCursor                          = LoadCursor (NU
        wndclass.hbrBackground                  = (HBRUSH) GetS
wndclass.lpszMenuName                   = NULL ;
        wndclass.lpszClassName                  = szAppName ;

        if (!RegisterClass (&wndclass))
 {
              MessageBox (  NULL, TEXT ("Program requires Windo
                                                  szAppName, MB_IC
         return 0 ;
        }

        hwnd = CreateWindow (szAppName, TEXT ("Mouse Butto
               WS_OVERLAPPEDWINDOW,
                CW_USEDEFAULT, CW_USEDEFAULT,
                CW_USEDEFAULT, CW_USEDEFAULT,
                NULL, NULL, hInstance, NULL) ;
```

```
        ShowWindow (hwnd, iCmdShow) ;
        UpdateWindow (hwnd) ;

        while (GetMessage (&msg, NULL, 0, 0))
        {
             TranslateMessage (&msg) ;
             DispatchMessage (&msg) ;
        }
        return msg.wParam ;
}

void DrawBoxOutline (HWND hwnd, POINT ptBeg, POINT ptEnd)
{
        HDC hdc ;
        hdc = GetDC (hwnd) ;
        SetROP2 (hdc, R2_NOT) ;
        SelectObject (hdc, GetStockObject (NULL_BRUSH)) ;
        Rectangle (hdc, ptBeg.x, ptBeg.y, ptEnd.x, ptEnd.y) ;
```

```
        ReleaseDC (hwnd, hdc) ;
}
LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
        static BOOL   fBlocking, fValidBox ;
        static POINT  ptBeg, ptEnd, ptBoxBeg, ptBoxEnd ;
        HDC                         hdc ;
        PAINTSTRUCT  ps ;
        switch (message)
        {
        case   WM_LBUTTONDOWN :
             ptBeg.x = ptEnd.x = LOWORD (lParam) ;
             ptBeg.y = ptEnd.y = HIWORD (lParam) ;

             DrawBoxOutline (hwnd, ptBeg, ptEnd) ;

             SetCursor (LoadCursor (NULL, IDC_CROSS)) ;
```

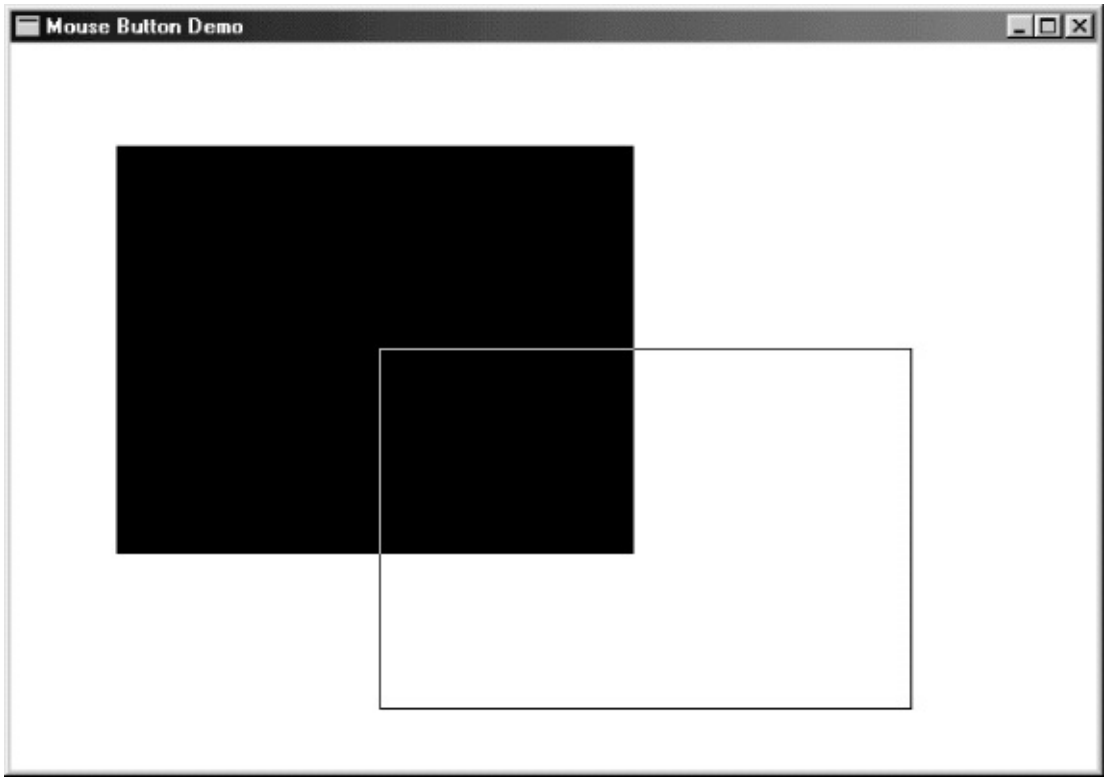
```
            fBlocking = TRUE ;
            return 0 ;

        case   WM_MOUSEMOVE :
            if (fBlocking)
            {
                        SetCursor (LoadCursor (NULL, IDC_CROSS)


                        DrawBoxOutline (hwnd, ptBeg, ptEnd) ;


                        ptEnd.x = LOWORD (lParam) ;
                        ptEnd.y = HIWORD (lParam) ;


                        DrawBoxOutline (hwnd, ptBeg, ptEnd) ;
            }
            return 0 ;


        case   WM_LBUTTONUP :
            if (fBlocking)
```

```
            {
                        DrawBoxOutline (hwnd, ptBeg, ptEnd) ;

                        ptBoxBeg                        = ptBeg ;
                        ptBoxEnd.x                       = LOWORD (lP
                        ptBoxEnd.y                       = HIWORD (lPa

                        SetCursor (LoadCursor (NULL, IDC_ARROW
                        fBlocking                       = FALSE ;
                        fValidBox                      = TRUE ;

                        InvalidateRect (hwnd, NULL, TRUE) ;
            }
            return 0 ;

      case   WM_CHAR :
            if (fBlocking & wParam == '\x1B')      // i.e., Escape
            {
                        DrawBoxOutline (hwnd, ptBeg, ptEnd) ;
```

```
                    SetCursor (LoadCursor (NULL, IDC_ARROW))

                    fBlocking = FALSE ;
          }
          return 0 ;

     case   WM_PAINT :
          hdc = BeginPaint (hwnd, &ps) ;

     if (fValidBox)
          {
                         SelectObject (hdc, GetStockObject (BL
                         Rectangle     (      hdc, ptBoxBeg.x, ptB
                                                ptBoxEnd.x, ptB
          }

          if (fBlocking)
          {
```

```
                    SetROP2 (hdc, R2_NOT) ;
                    SelectObject (hdc, GetStockObject (NULL_B
                    Rectangle (hdc, ptBeg.x, ptBeg.y, ptEnd.x,
               }

               EndPaint (hwnd, &ps) ;
               return 0 ;

          case   WM_DESTROY :
               PostQuitMessage (0) ;
              return 0 ;
     }
          return DefWindowProc (hwnd, message, wParam, lParam)
}
```

Windows7-4



7-4 BLOKOUT1

BLOKOUT1WM_MOUSEMOVEBLOKOUT1
WM_BUTTONUPBLOKOUT1

BLOKOUT1

```
SetCapture (hwnd) ;
```

WindowshwndlParamxy

```
ReleaseCapture () ;
```

32WindowsWindows

BLOKOUT2

BLOKOUT27-7

7-7　BLOKOUT2

```
BLOKOUT2.C
/*------------------------------------------------------------------------
  BLOKOUT2.C --         Mouse Button & Capture Demo Program
                                (c) Charles Petzold, 1998
------------------------------------------------------------------------*/
#include <windows.h>
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                PSTR szCmdLine, int iCmdShow)
{
```

```
        static TCHAR szAppName[] = TEXT ("BlokOut2") ;
        HWND                 hwnd ;
        MSG                  msg ;
        WNDCLASS             wndclass ;

wndclass.style                  = CS_HREDRAW | CS_VREDR
        wndclass.lpfnWndProc            = WndProc ;
        wndclass.cbClsExtra             = 0 ;
        wndclass.cbWndExtra             = 0 ;
        wndclass.hInstance              = hInstance ;
        wndclass.hIcon                  = LoadIcon (NULL, IDI_
wndclass.hCursor                        = LoadCursor (NULL,
        wndclass.hbrBackground          = (HBRUSH) GetStockO
        wndclass.lpszMenuName  = NULL ;
        wndclass.lpszClassName          = szAppName ;

 if (!RegisterClass (&wndclass))
        {
       MessageBox (  NULL, TEXT ("Program requires Windows N
```

```
                                    szAppName, MB_ICONERROR) ;
        return 0 ;
}

        hwnd = CreateWindow (szAppName, TEXT ("Mouse Butto
                             WS_OVERLAPPEDWINDOW,
                CW_USEDEFAULT, CW_USEDEFAULT,
                CW_USEDEFAULT, CW_USEDEFAULT,
                NULL, NULL, hInstance, NULL) ;

        ShowWindow (hwnd, iCmdShow) ;
        UpdateWindow (hwnd) ;

        while (GetMessage (&msg, NULL, 0, 0))
        {
                        TranslateMessage (&msg) ;
                        DispatchMessage (&msg) ;
        }
        return msg.wParam ;
```

```
}

void DrawBoxOutline (HWND hwnd, POINT ptBeg, POINT ptEnd)
{
          HDC hdc ;
          hdc = GetDC (hwnd) ;
          SetROP2 (hdc, R2_NOT) ;
          SelectObject (hdc, GetStockObject (NULL_BRUSH)) ;
          Rectangle (hdc, ptBeg.x, ptBeg.y, ptEnd.x, ptEnd.y) ;

          ReleaseDC (hwnd, hdc) ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
          static BOOL  fBlocking, fValidBox ;
          static POINT ptBeg, ptEnd, ptBoxBeg, ptBoxEnd ;
          HDC          hdc ;
          PAINTSTRUCT  ps ;
```

```
switch (message)
{
case   WM_LBUTTONDOWN :
      ptBeg.x = ptEnd.x = LOWORD (lParam) ;
      ptBeg.y = ptEnd.y = HIWORD (lParam) ;

      DrawBoxOutline (hwnd, ptBeg, ptEnd) ;

      SetCapture (hwnd) ;
      SetCursor (LoadCursor (NULL, IDC_CROSS)) ;

      fBlocking = TRUE ;
      return 0 ;

case   WM_MOUSEMOVE :
      if (fBlocking)
      {
                  SetCursor (LoadCursor (NULL, IDC_CROSS)
```

```
                    DrawBoxOutline (hwnd, ptBeg, ptEnd) ;

                    ptEnd.x = LOWORD (lParam) ;
                    ptEnd.y = HIWORD (lParam) ;

                    DrawBoxOutline (hwnd, ptBeg, ptEnd) ;
          }
          return 0 ;

     case   WM_LBUTTONUP :
          if (fBlocking)
          {
                    DrawBoxOutline (hwnd, ptBeg, ptEnd) ;

                    ptBoxBeg                = ptBeg ;
                    ptBoxEnd.x              = LOWORD (lParam
                    ptBoxEnd.y              = HIWORD (lParam
```

```
                    ReleaseCapture () ;
                    SetCursor (LoadCursor (NULL, IDC_ARROW

                    fBlocking               = FALSE ;
                    fValidBox               = TRUE ;

                InvalidateRect (hwnd, NULL, TRUE) ;
          }
          return 0 ;

     case   WM_CHAR :
          if (fBlocking & wParam == '\x1B')      // i.e., Escape
          {
                    DrawBoxOutline (hwnd, ptBeg, ptEnd) ;
                    ReleaseCapture () ;
                    SetCursor (LoadCursor (NULL, IDC_ARROW

                    fBlocking = FALSE ;
          }
```

```
            return 0 ;

        case   WM_PAINT :
            hdc = BeginPaint (hwnd, &ps) ;

            if (fValidBox)
        {
                        SelectObject (hdc, GetStockObject (BLACK_
                        Rectangle (hdc, ptBoxBeg.x, ptBoxBeg.y,
                        ptBoxEnd.x, ptBoxEnd.y) ;
            }

                  if (fBlocking)
            {
                        SetROP2 (hdc, R2_NOT) ;
                       SelectObject (hdc, GetStockObject (NULL_B
                        Rectangle (hdc, ptBeg.x, ptBeg.y, ptEnd.x,
            }
```

```
            EndPaint (hwnd, &ps) ;
            return 0 ;

        case   WM_DESTROY :
            PostQuitMessage (0) ;
            return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

BLOKOUT2BLOKOUT1WM_LBUTTONDOWNSetCapture
WM_LBUTTONDOWNWM_CHARReleaseCapture

WM_MOUSEMOVE

Microsoft IntelliMouseWM_MOUSEWHEEL
Microsoft WordMicrosoft Internet Explorer

SYSMETS4SYSMETS7-8

7-8 SYSMETS4

SYSMETS.C

```
/*------------------------------------------------------------------------
SYSMETS.C -- Final System Metrics Display Program
                        (c) Charles Petzold, 1998
------------------------------------------------------------------------*/
#include <windows.h>
#include "sysmets.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                    PSTR szCmdLine, int iCmdSh
{
     static TCHAR szAppName[] = TEXT ("SysMets") ;
     HWND         hwnd ;
     MSG          msg ;
     WNDCLASS     wndclass ;
    wndclass.style                  = CS_HREDRAW | C
     wndclass.lpfnWndProc              = WndProc ;
     wndclass.cbClsExtra               = 0 ;
     wndclass.cbWndExtra               = 0 ;
```

```
        wndclass.hInstance                  = hInstance ;
        wndclass.hIcon                      = LoadIcon (NULL,
        wndclass.hCursor                    = LoadCursor (N
        wndclass.hbrBackground              = (HBRUSH) GetS
        wndclass.lpszMenuName               = NULL ;
        wndclass.lpszClassName              = szAppName ;

if (!RegisterClass (&wndclass))
        {
         MessageBox (  NULL, TEXT ("Program requires Windows I
                                             szAppName, MB_IC
              return 0 ;
        }

 hwnd = CreateWindow (szAppName, TEXT ("Get System Metr
                   WS_OVERLAPPEDWINDOW | WS_VSCROLL | WS_
                    CW_USEDEFAULT, CW_USEDEFAULT,
                   CW_USEDEFAULT, CW_USEDEFAULT,
                   NULL, NULL, hInstance, NULL) ;
```

```
        ShowWindow (hwnd, iCmdShow) ;
             UpdateWindow (hwnd) ;

  while (GetMessage (&msg, NULL, 0, 0))
        {
             TranslateMessage (&msg) ;
             DispatchMessage (&msg) ;
   }
   return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
   static int    cxChar, cxCaps, cyChar, cxClient, cyClient, iMaxWi
   static       int          iDeltaPerLine, iAccumDelta ;        // for m
             HDC        hdc ;
             int        i, x, y, iVertPos, iHorzPos, iPaintBeg, iPaintEn
             PAINTSTRUCT          ps ;
```

```
        SCROLLINFO          si ;
        TCHAR               szBuffer[10] ;
        TEXTMETRIC          tm ;
        ULONG               ulScrollLines ;       // for mouse
    switch (message)
    {
    case   WM_CREATE:
        hdc = GetDC (hwnd) ;

        GetTextMetrics (hdc, &tm) ;
        cxChar = tm.tmAveCharWidth ;
        cxCaps = (tm.tmPitchAndFamily & 1 ? 3 : 2) * cxChar
        cyChar = tm.tmHeight + tm.tmExternalLeading ;

       ReleaseDC (hwnd, hdc) ;

                                // Save the width of the three colu

        iMaxWidth = 40 * cxChar + 22 * cxCaps ;
```

```
                                // Fall through for mouse wheel in

        case   WM_SETTINGCHANGE:
               SystemParametersInfo (SPI_GETWHEELSCROLLLINES

               // ulScrollLines usually equals 3 or 0 (for no scrolling
               // WHEEL_DELTA equals 120, so iDeltaPerLine will be

               if (ulScrollLines)
                                iDeltaPerLine = WHEEL_DELTA / ulScro
               else
                                iDeltaPerLine = 0 ;

         return 0 ;

        case   WM_SIZE:
               cxClient = LOWORD (lParam) ;
        cyClient = HIWORD (lParam) ;

                                // Set vertical scroll bar range and
```

```
        si.cbSize                   = sizeof (si) ;
        si.fMask                    = SIF_RANGE | SIF_PAGE ;
        si.nMin                     = 0 ;
        si.nMax                     = NUMLINES - 1 ;
        si.nPage                    = cyClient / cyChar ;
        SetScrollInfo (hwnd, SB_VERT, &si, TRUE) ;

                                // Set horizontal scroll bar range a

        si.cbSize                   = sizeof (si) ;
        si.fMask                    = SIF_RANGE | SIF_PAGE ;
       si.nMin                      = 0 ;
        si.nMax                     = 2 + iMaxWidth / cxChar
       si.nPage                     = cxClient / cxChar ;
SetScrollInfo (hwnd, SB_HORZ, &si, TRUE) ;
        return 0 ;

case WM_VSCROLL:
                                // Get all the vertical scroll bar inf
```

```
        si.cbSize = sizeof (si) ;
        si.fMask  = SIF_ALL ;
        GetScrollInfo (hwnd, SB_VERT, &si) ;

                                // Save the position for comparisc
        iVertPos = si.nPos ;
    switch (LOWORD (wParam))
        {
        case   SB_TOP:
                        si.nPos = si.nMin ;
                        break ;

        case   SB_BOTTOM:
                        si.nPos = si.nMax ;
                        break ;

        case SB_LINEUP:
                        si.nPos -= 1 ;
                        break ;
```

```
        case   SB_LINEDOWN:
                        si.nPos += 1 ;
                        break ;

        case SB_PAGEUP:
                        si.nPos -= si.nPage ;
                        break ;

        case   SB_PAGEDOWN:
                        si.nPos += si.nPage ;
                        break ;

        case   SB_THUMBTRACK:
                       si.nPos = si.nTrackPos ;
                        break ;
        default:
                        break ;
        }
```

```
                    // Set the position and then retrieve it.
            //   by Windows it may not be the same as the v

        si.fMask = SIF_POS ;
        SetScrollInfo (hwnd, SB_VERT, &si, TRUE) ;
        GetScrollInfo (hwnd, SB_VERT, &si) ;

                    // If the position has changed, scroll th

        if (si.nPos != iVertPos)
        {
                ScrollWindow (hwnd, 0, cyChar * (iVertPos -
                    NULL, NULL) ;
                UpdateWindow (hwnd) ;
       }
        return 0 ;

    case   WM_HSCROLL:
                    // Get all the vertical scroll bar informa
```

```
                si.cbSize = sizeof (si) ;
                si.fMask  = SIF_ALL ;

                                // Save the position for comparison lat

                GetScrollInfo (hwnd, SB_HORZ, &si) ;
                iHorzPos = si.nPos ;

                switch (LOWORD (wParam))
                {
                case   SB_LINELEFT:
                                si.nPos -= 1 ;
                                break ;

                case   SB_LINERIGHT:
                                si.nPos += 1 ;
                                break ;

                case   SB_PAGELEFT:
                                si.nPos -= si.nPage ;
                                break ;
```

```
        case   SB_PAGERIGHT:
                        si.nPos += si.nPage ;
                        break ;

        case   SB_THUMBPOSITION:
                        si.nPos = si.nTrackPos ;
                       break ;

        default:
                     break ;
        }
                    // Set the position and then retrieve it.  Due
         //   by Windows it may not be the same as the value

    si.fMask = SIF_POS ;
        SetScrollInfo (hwnd, SB_HORZ, &si, TRUE) ;
        GetScrollInfo (hwnd, SB_HORZ, &si) ;
```

```
                    // If the position has changed, scroll the wi

          if (si.nPos != iHorzPos)
          {
                    ScrollWindow (hwnd, cxChar * (iHorzPo
                              NULL, NULL) ;
          }
          return 0 ;

     case   WM_KEYDOWN :
          switch (wParam)
          {
          case   VK_HOME :
                    SendMessage (hwnd, WM_VSCROLL, S
                    break ;

          case   VK_END :
                    SendMessage (hwnd, WM_VSCROLL, S
                    break ;
```

```
            case   VK_PRIOR :
                        SendMessage (hwnd, WM_VSCROLL, S
                        break ;

          case   VK_NEXT :
                       SendMessage (hwnd, WM_VSCROLL, SE
                        break ;
            case   VK_UP :
                        SendMessage (hwnd, WM_VSCROLL, S
                        break ;

            case   VK_DOWN :
                       SendMessage (hwnd, WM_VSCROLL, SE
                        break ;

            case   VK_LEFT :
                        SendMessage (hwnd, WM_HSCROLL, S
                        break ;
```

```
        case   VK_RIGHT :
                        SendMessage (hwnd, WM_HSCROLL, SB
                         break ;
        }
        return 0 ;

    case   WM_MOUSEWHEEL:
        if (iDeltaPerLine == 0)
                        break ;

        iAccumDelta += (short) HIWORD (wParam) ;     // 120

        while (iAccumDelta >= iDeltaPerLine)
        {
                        SendMessage (hwnd, WM_VSCROLL, S
                        iAccumDelta -= iDeltaPerLine ;
        }

        while (iAccumDelta <= -iDeltaPerLine)
```

```
            {
                        SendMessage (hwnd, WM_VSCROLL, SE
                        iAccumDelta += iDeltaPerLine ;
            }

            return 0 ;

      case   WM_PAINT :
            hdc = BeginPaint (hwnd, &ps) ;

                        // Get vertical scroll bar position

                  si.cbSize            = sizeof (si) ;
                  si.fMask                  = SIF_POS ;
                  GetScrollInfo (hwnd, SB_VERT, &si) ;
                  iVertPos                  = si.nPos ;

                        // Get horizontal scroll bar position

                  GetScrollInfo (hwnd, SB_HORZ, &si) ;
                  iHorzPos = si.nPos ;
```

```
                    // Find painting limits

          iPaintBeg = max (0, iVertPos + ps.rcPaint.top / c
          iPaintEnd = min (NUMLINES - 1,
     iVertPos + ps.rcPaint.bottom / cyChar) ;

          for (i = iPaintBeg ; i <= iPaintEnd ; i++)
          {
                         x = cxChar * (1 - iHorzPos) ;
                          y = cyChar * (i - iVertPos) ;

                         TextOut (     hdc, x, y,
          sysmetrics[i].szLabel,
           lstrlen (sysmetrics[i].szLabel)) ;

                         TextOut (     hdc, x + 22 * cxCaps, y,
          sysmetrics[i].szDesc,
          lstrlen (sysmetrics[i].szDesc)) ;
```

```
                        SetTextAlign (hdc, TA_RIGHT | TA_TOP)

            TextOut  (    hdc, x + 22 * cxCaps + 40 * cxChar, y, s
          wsprintf (szBuffer, TEXT ("%5d"),
          GetSystemMetrics (sysmetrics[i].iIndex))) ;

                        SetTextAlign (hdc, TA_LEFT | TA_TOP) ;
            }

            EndPaint (hwnd, &ps) ;
            return 0 ;

        case   WM_DESTROY :
            PostQuitMessage (0) ;
            return 0 ;
  }
        return DefWindowProc (hwnd, message, wParam, lParam)
}
```

WindowsWM_MOUSEWHEELlParamwParam

ShiftCtrl

wParamdelta120-120120-120
delta40-40WM_MOUSEWHEEL

SYSMETSWM_CREATEWM_SETTINGCHANGE
SPI_GETWHEELSCROLLLINESSystemParametersInfoWHEEL_DELTA
deltaWHEEL_DELTAWINUSER.HWHEEL_DELTA120
SystemParametersInfo3delta40SYSMETSiDeltaPerLine

WM_MOUSEWHEELSYSMETSdeltaiAccumDeltaiAccumDelta
iDeltaPerLine-iDeltaPerLinSYSMETSSB_LINEUPSB_LINEDOWN
WM_VSCROLLWM_VSCROLLiAccumDeltaiDeltaPerLine
deltadelta

Windows

Microsoft WindowsWindows10WindowsWM_TIMER

WindowsWindowsWindows

- Windows 98WindowsWM_TIMER
-
- Windows
- 30
-
- CD

SetTimerWindowsSetTimer14,294,967,29550Windows
WM_TIMER1000WindowsWM_TIMER

KillTimerWM_TIMERKillTimerKillTimerWM_TIMER
KillTimerWM_TIMER

WindowsPCROM BIOSWindowsMS-DOStimer
54.91518.2IBM PC4.772720

WindowsBIOSWindowsWindowstimer
WindowsWM_TIMER

WindowsWM_TIMERWM_TIMERWindows

Windows 98PC55Microsoft Win

WindowsWindows 9818.2Windows
SetTimertick100054.92518.207tick18tick98955tick
WM_TIMER

WM_TIMER

WM_TIMERWM_TIMERSetTimer10001000989
WM_TIMERWM_TIMERWindowsWM_TIMER
WM_PAINT

WM_TIMERWM_PAINTWindowsWM_TIMERWM_TIMER
WM_TIMERWM_TIMER

WM_TIMERWM_TIMER

WM_TIMERtick

WinMainWM_CREATESetTimerWinMainWM_DESTROYKillTimer
SetTimer

WindowsWM_TIMERSetTimer

```
SetTimer (hwnd, 1, uiMsecInterval, NULL) ;
```

WM_TIMERID013260,000WindowsWM_TIMER

```
KillTimer (hwnd, 1) ;
```

WM_TIMERWM_TIMERSetTimerIDWM_DESTROY

WM_TIMERwParamID1lParam0IDwParamWM_TIMER
#defineID

```
#define TIMER_SEC 1

#define TIMER_MIN 2
```

SetTimer

```
SetTimer (hwnd, TIMER_SEC, 1000, NULL) ;

SetTimer (hwnd, TIMER_MIN, 60000, NULL) ;
```

WM_TIMER

```
caseWM_TIMER:
   switch (wParam)
   {
   case TIMER_SEC:
         //
              break ;
   case TIMER_MIN:
         //
```

```
        break ;
   }
return 0 ;
```

SetTimer100060

8-1BEEPER11WM_TIMERMessageBeep
MessageBeepMessageBoxPCMB_ICONMessageBeepMessageBox

BEEPER1WM_CREATEWM_TIMERBEEPER1MessageBeep
bFlipFlopWM_PAINTWM_PAINTBEEPER1GetClientRectRECT
FillRect

8-1 BEEPER1

```
BEEPER1.C
/*------------------------------------------------------------------------
   BEEPER1.C  -- Timer Demo Program No. 1
                             (c) Charles Petzold, 1998
------------------------------------------------------------------------*/
#include <windows.h>

#define ID_TIMER    1
```

```
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                    PSTR szCmdLine, int iCmdShow)
{
   static TCHAR szAppName[] = TEXT ("Beeper1") ;
   HWND         hwnd ;
   MSG          msg ;
   WNDCLASS     wndclass ;

   wndclass.style         = CS_HREDRAW | CS_VREDR
   wndclass.lpfnWndProc   = WndProc ;
   wndclass.cbClsExtra    = 0 ;
   wndclass.cbWndExtra    = 0 ;
   wndclass.hInstance     = hInstance ;
        wndclass.hIcon         = LoadIcon (NULL, IDI_APPL
        wndclass.hCursor       = LoadCursor (NULL, IDC_A
        wndclass.hbrBackground = (HBRUSH) GetStockO
        wndclass.lpszMenuName  = NULL ;
        wndclass.lpszClassName = szAppName ;
```

```
      if (!RegisterClass (&wndclass))
{
       MessageBox (  NULL, TEXT ("Program requires Windows I
                                  szAppName, MB_ICONERROR) ;
       return 0 ;
      }

      hwnd = CreateWindow (szAppName, TEXT ("Beeper1 Tim
                        WS_OVERLAPPEDWINDOW,
            CW_USEDEFAULT, CW_USEDEFAULT,
           CW_USEDEFAULT, CW_USEDEFAULT,
                  NULL, NULL, hInstance, NULL) ;

ShowWindow (hwnd, iCmdShow) ;
UpdateWindow (hwnd) ;

while (GetMessage (&msg, NULL, 0, 0))
{
```

```
            TranslateMessage (&msg) ;
            DispatchMessage (&msg) ;
    }
        return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
    static BOOL   fFlipFlop = FALSE ;
    HBRUSH              hBrush ;
    HDC                     hdc ;
    PAINTSTRUCT ps ;
    RECT       rc ;

    switch (message)
    {
    case   WM_CREATE:
        SetTimer (hwnd, ID_TIMER, 1000, NULL) ;
            return 0 ;
```

```
case   WM_TIMER :
      MessageBeep (-1) ;
      fFlipFlop = !fFlipFlop ;
      InvalidateRect (hwnd, NULL, FALSE) ;
      return 0 ;

case   WM_PAINT :
      hdc = BeginPaint (hwnd, &ps) ;

      GetClientRect (hwnd, &rc) ;
      hBrush = CreateSolidBrush (fFlipFlop ? RGB(255,0,0)
      FillRect (hdc, &rc, hBrush) ;

 EndPaint (hwnd, &ps) ;
DeleteObject (hBrush) ;
      return 0 ;

case   WM_DESTROY :
```

```
            KillTimer (hwnd, ID_TIMER) ;

            PostQuitMessage (0) ;

        return 0 ;

        }

        return DefWindowProc (hwnd, message, wParam, lParam)
}
```

BEEPER1WM_TIMERBEEPER1WM_TIMERWindows

BEEPER1WindowsBEEPER1
BEEPER1WM_TIMER1

Windows

WM_TIMERWindows

callbackWindowsWindowsWindowscallback
WindowsWindows

SetTimercallbackWindowsCreateDialogDialogBox
WindowsEnumChildWindowEnumFontsEnumObjectsEnumProps
EnumWindowcallbackGrayStringLineDDASetWindowHookEx
callback

callbackCALLBACKWindowscallbackcallbackcallbackcallback
callbackWindows

callbackTimerProcWM_TIMER

```
VOID CALLBACK TimerProc (  HWND hwnd, UINT message, UINT
```

```
{
        WM_TIMER
}
```

TimerProchwndSetTimerWindowsWM_TIMERTimerProcWM_TIMER iTimerIDIDdwTimerGetTickCountWindows

BEEPER1SetTimer

```
SetTimer (hwnd, iTimerID, iMsecInterval, NULL) ;
```

callbackWM_TIMERSetTimercallback

```
SetTimer (hwnd, iTimerID, iMsecInterval, TimerProc) ;
```

WindowsTimerProcWndProc8-2BEEPER2BEEPER1 TimerProcWndProc

```
8-2  BEEPER2

BEEPER2.C

/*------------------------------------------------------------------------

  BEEPER2.C -- Timer Demo Program No. 2

                          (c) Charles Petzold, 1998

------------------------------------------------------------------------*/

#include <windows.h>
```

```
#define ID_TIMER    1

LRESULT     CALLBACK            WndProc   (HWND, UINT, WPARAM
VOID    CALLBACK   TimerProc (HWND, UINT, UINT,   DWORD ) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                        PSTR szCmdLine, int iCmdSh
{
  static char   szAppName[]         = "Beeper2" ;
         HWND                        hwnd ;
         MSG                        msg ;
         WNDCLASS                   wndclass ;

         wndclass.style                          = CS_HREDRAW |
         wndclass.lpfnWndProc                       = WndProc ;
        wndclass.cbClsExtra                       = 0 ;
         wndclass.cbWndExtra                       = 0 ;
         wndclass.hInstance                        = hInstance ;
         wndclass.hIcon                          = LoadIcon (NULL
```

```
     wndclass.hCursor                        = LoadCursor (N
    wndclass.hbrBackground                  = (HBRUSH) GetSt
     wndclass.lpszMenuName                   = NULL ;
     wndclass.lpszClassName                  = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (  NULL, TEXT ("Program requires Windo
                                              szAppName, MB_IC
          return 0 ;
     }

     hwnd = CreateWindow ( szAppName, "Beeper2 Timer De
                                        WS_OVERLAPPEDWINDOW,
                    CW_USEDEFAULT, CW_USEDEFAULT,
                CW_USEDEFAULT, CW_USEDEFAULT,
               NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
```

```
        UpdateWindow (hwnd) ;

        while (GetMessage (&msg, NULL, 0, 0))
        {
             TranslateMessage (&msg) ;
             DispatchMessage (&msg) ;
        }
        return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
  switch (message)
   {
        case   WM_CREATE:
             SetTimer (hwnd, ID_TIMER, 1000, TimerProc) ;
            return 0 ;

        case   WM_DESTROY:
```

```
            KillTimer (hwnd, ID_TIMER) ;
            PostQuitMessage (0) ;
            return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam)
}

VOID CALLBACK TimerProc (HWND hwnd, UINT message, UINT iT
{
        static BOOL fFlipFlop = FALSE ;
        HBRUSH                  hBrush ;
        HDC                     hdc ;
        RECT                    rc ;

        MessageBeep (-1) ;
        fFlipFlop = !fFlipFlop ;

        GetClientRect (hwnd, &rc) ;
       hdc = GetDC (hwnd) ;
```

```
    hBrush = CreateSolidBrush (fFlipFlop ? RGB(255,0,0) : RGB(0,



    FillRect (hdc, &rc, hBrush) ;
   ReleaseDC (hwnd, hdc) ;
   DeleteObject (hBrush) ;
}
```

SetTimerhwndNULLIDID

```
iTimerID = SetTimer (NULL, 0, wMsecInterval, TimerProc) ;
```

SetTimeriTimerIDNULL

KillTimerNULLIDSetTimer

```
KillTimer (NULL, iTimerID) ;
```

TimerProchwndNULLSetTimerID

Windows

8-3DIGCLOCKLED7

```
8-3  DIGCLOCK

DIGCLOCK.C

/*------------------------------------------------------------------------
  DIGCLOCK.C --         Digital Clock
                                (c) Charles Petzold, 1998
------------------------------------------------------------------------*/

#include <windows.h>

#define ID_TIMER    1

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                          PSTR szCmdLine, int iCmdSh
{
   static TCHAR szAppName[] = TEXT ("DigClock") ;
          HWND                          hwnd ;
          MSG                           msg ;
          WNDCLASS                      wndclass ;

          wndclass.style                              = CS_HREDRAW |
```

```
     wndclass.lpfnWndProc                   = WndProc ;
     wndclass.cbClsExtra                    = 0 ;
     wndclass.cbWndExtra                    = 0 ;
    wndclass.hInstance                      = hInstance ;
     wndclass.hIcon                         = LoadIcon (NULL,
     wndclass.hCursor                       = LoadCursor (N
     wndclass.hbrBackground                 = (HBRUSH) GetS
    wndclass.lpszMenuName                   = NULL ;
     wndclass.lpszClassName                 = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (  NULL, TEXT ("Program requires Windo
                                          szAppName, MB_IC
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, TEXT ("Digital Clock
               WS_OVERLAPPEDWINDOW,
               CW_USEDEFAULT, CW_USEDEFAULT,
```

```
                CW_USEDEFAULT, CW_USEDEFAULT,
                NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
          {
               TranslateMessage (&msg) ;
               DispatchMessage (&msg) ;
          }
     return msg.wParam ;
     }

void DisplayDigit (HDC hdc, int iNumber)
{
     static BOOL  fSevenSegment [10][7] = {
               1, 1,    1,    0,    1,    1,    1,         // 0
               0, 0,    1,    0,    0,    1,    0,         // 1
               1, 0,    1,    1,    1,    0,    1,         // 2
```

```
                1, 0,    1,    1,    0,    1,    1,          // 3
                0, 1,    1,    1,    0,    1,    0,          // 4
                1, 1,    0,    1,    0,    1,    1,          // 5
               1, 1,    0,    1,    1,    1,    1,          // 6
                1, 0,    1,    0,    0,    1,    0,          // 7
                1, 1,    1,    1,    1,    1,    1,          // 8
                1, 1,    1,    1,    0,    1,    1 } ;       // 9
      static POINT ptSegment [7][6] = {
                        7,  6,  11,   2,  31,  2, 35,  6,  31, 10, 11, 1
            6,  7,  10, 11, 10, 31, 6,   35, 2,  31, 2,  11,
            36, 7,  40, 11, 40, 31, 36,  35, 32, 31, 32, 11,
            7 , 36, 11, 32, 31, 32, 35,  36, 31, 40, 11, 40,
            6 , 37, 10, 41, 10, 61, 6,   65, 2,  61, 2,  41,
            36, 37, 40, 41, 40, 61, 36,  65, 32, 61, 32, 41,
            7 , 66, 11, 62, 31, 62, 35,  66, 31, 70, 11, 70 } ;
int         iSeg ;
      for (iSeg = 0 ; iSeg < 7 ; iSeg++)
            if (fSevenSegment [iNumber][iSeg])
```

```
                                Polygon (hdc, ptSegment [iSeg], 6) ;
}

void DisplayTwoDigits (HDC hdc, int iNumber, BOOL fSuppress)
{
          if (!fSuppress || (iNumber / 10 != 0))
               DisplayDigit (hdc, iNumber / 10) ;
          OffsetWindowOrgEx (hdc, -42, 0, NULL) ;
          DisplayDigit (hdc, iNumber % 10) ;
         OffsetWindowOrgEx (hdc, -42, 0, NULL) ;
}

void DisplayColon (HDC hdc)
{
          POINT ptColon [2][4] = {      2,     21,    6,     17,    10,    2
                              2,51,    6,     47,    10,    51,    6,     55 } ;

    Polygon (hdc, ptColon [0], 4) ;
    Polygon (hdc, ptColon [1], 4) ;

    OffsetWindowOrgEx (hdc, -12, 0, NULL) ;
```

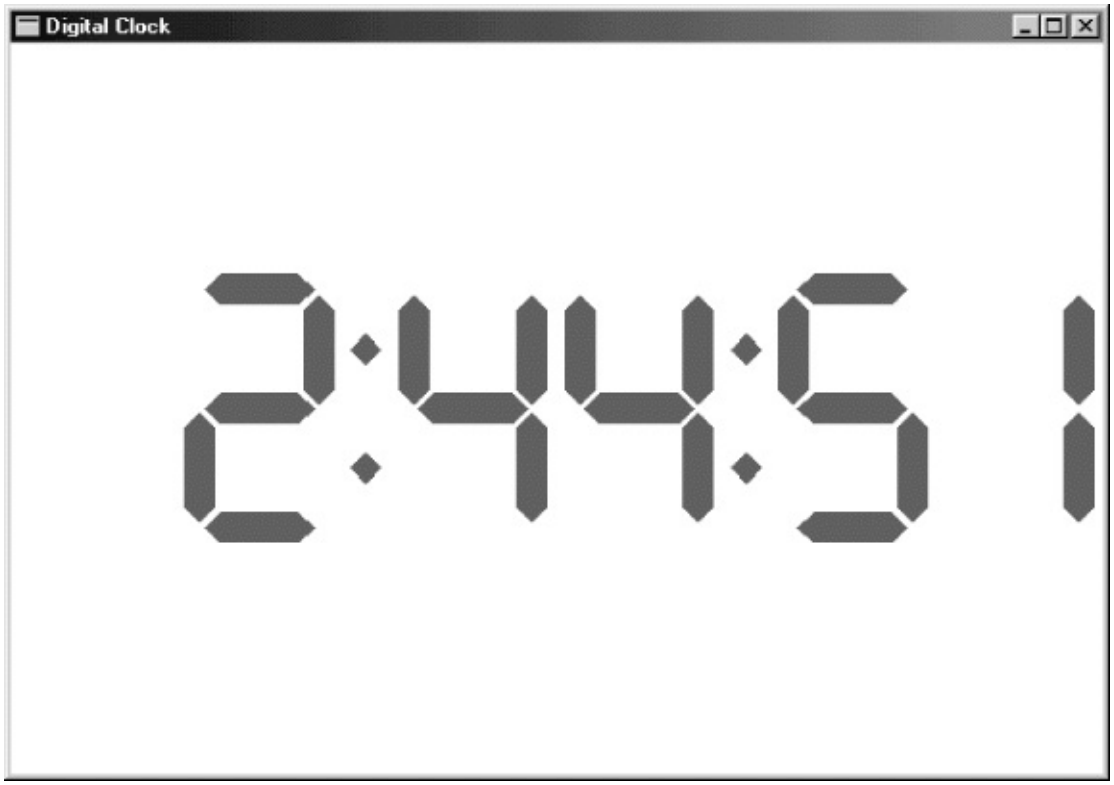
```
}

void DisplayTime (HDC hdc, BOOL f24Hour, BOOL fSuppress)
{
   SYSTEMTIME st ;
   GetLocalTime (&st) ;
   if (f24Hour)
         DisplayTwoDigits (hdc, st.wHour, fSuppress) ;
   else
   DisplayTwoDigits (hdc, (st.wHour %= 12) ? st.wHour : 12, fSup
   DisplayColon (hdc) ;
   DisplayTwoDigits (hdc, st.wMinute, FALSE) ;
   DisplayColon (hdc) ;
   DisplayTwoDigits (hdc, st.wSecond, FALSE) ;
}
LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
   static BOOL          f24Hour, fSuppress ;
   static HBRUSH        hBrushRed ;
```

```
static int                  cxClient, cyClient ;
HDC                                 hdc ;
PAINTSTRUCT   ps ;
TCHAR                           szBuffer [2] ;

switch (message)
      {
      case   WM_CREATE:
           hBrushRed = CreateSolidBrush (RGB (255, 0, 0)) ;
          SetTimer (hwnd, ID_TIMER, 1000, NULL) ;// fall throug

case   WM_SETTINGCHANGE:
       GetLocaleInfo (LOCALE_USER_DEFAULT, LOCALE_ITIME, s
       f24Hour = (szBuffer[0] == '1') ;

       GetLocaleInfo (LOCALE_USER_DEFAULT, LOCALE_ITLZERO
       fSuppress = (szBuffer[0] == '0') ;

       InvalidateRect (hwnd, NULL, TRUE) ;
       return 0 ;
```

```
 case   WM_SIZE:
        cxClient = LOWORD (lParam) ;
             cyClient = HIWORD (lParam) ;
             return 0 ;

       case   WM_TIMER:
             InvalidateRect (hwnd, NULL, TRUE) ;
            return 0 ;

case   WM_PAINT:
             hdc = BeginPaint (hwnd, &ps) ;

             SetMapMode (hdc, MM_ISOTROPIC) ;
             SetWindowExtEx (hdc, 276, 72, NULL) ;
             SetViewportExtEx (hdc, cxClient, cyClient, NULL) ;

             SetWindowOrgEx (hdc, 138, 36, NULL) ;
             SetViewportOrgEx (hdc, cxClient / 2, cyClient / 2, NUL
             SelectObject (hdc, GetStockObject (NULL_PEN)) ;
             SelectObject (hdc, hBrushRed) ;
```

```
            DisplayTime (hdc, f24Hour, fSuppress) ;

            EndPaint (hwnd, &ps) ;
            return 0 ;

        case   WM_DESTROY:
            KillTimer (hwnd, ID_TIMER) ;
            DeleteObject (hBrushRed) ;
            PostQuitMessage (0) ;
            return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

DIGCLOCK8-1



8-1 DIGCLOCK

8-1DIGCLOCKWM_CREATEWM_DESTROYWM_CREATE
DIGCLOCKWM_DESTROYGetLocaleInfo

WM_TIMERDIGCLOCKInvalidateRect

WM_TIMERWM_PAINTDIGCLOCKWM_PAINTMM_ISOTROPIC
DIGCLOCKSetWindowExtEx27672

DIGCLOCK138,36cxClient
DIGCLOCK0, 0

WM_PAINTNULL_PENDIGCLOCKDisplayTime

DisplayTimeWindowsGetLocalTimeSYSTEMTIMEWINBASE.H

```
typedef struct _SYSTEMTIME
{
        WORD   wYear ;
        WORD   wMonth ;
        WORD   wDayOfWeek ;
        WORD   wDay ;
        WORD   wHour ;
        WORD   wMinute ;
        WORD   wSecond ;
       WORD   wMilliseconds ;
}
SYSTEMTIME, * PSYSTEMTIME ;
```

SYSTEMTIME1100wDay1

SYSTEMTIMEGetLocalTimeGetSystemTimeGetSystemTimeCoordinated Universal TimeUTCGetLocalTime

WindowsSetLocalTimeSetSystemTime/Platform SDK/Windows Services/General Library/Time

DIGCLOCK7Polygon

DIGCLOCKDisplayDigitfSevenSegment7BOOL091

0776ptSegmentPOINT7

```
for (iSeg = 0 ; iSeg < 7 ; iSeg++)
        if (   fSevenSegment [iNumber][iSeg])
             Polygon (hdc, ptSegment [iSeg], 6) ;
```

DisplayColon421262276SetWindowExtEx

DisplayTimeDisplayTimeDisplayTwoDigitsDisplayTwoDigits
DisplayDigitOffsetWindowOrgEx42DisplayColon12

12240

DIGCLOCKWindowsGetDateFormatGetTimeFormat/Platform
SDK/Windows Base Services/General Library/String Manipulation/String
Manipulation Reference/String Manipulation Functions/Platform
SDK/Windows Base Services/International Features/National Language Support
SYSTEMTIME

DIGCLOCKGetDateFormatDIGCLOCK1224
GetLocaleInfoGetLocaleInfo/Platform SDK/Windows E
Services/General Library/String Manipulation/String Manipulation
Reference/String Manipulation Functions/Platform
Services/International Features/National Language Support/National Language
Support Constants

DIGCLOCKWM_CREATEGetLocaleInfoLOCALE_ITIME1224
LOCALE_ITLZERO0GetLocaleInfoDIGCLOCK
DisplayTime

WM_SETTINGCHANGEDIGCLOCKGetLocaleInfo

DIGCLOCKLOCALE_STIMEGetLocaleInfoDIGCLOCK
A.M.P.M.LOCALE_S1159LOCALE_S2359GetLocaleInfo/

DIGCLOCKWM_TIMECHANGEDIGCLOCKWM_TIMER
WM_TIMECHANGE

CLOCK8-4

8-4　CLOCK

```
CLOCK.C

/*------------------------------------------------------------------------

  CLOCK.C --      Analog Clock Program

                          (c) Charles Petzold, 1998

------------------------------------------------------------------------*/

#include <windows.h>

#include <math.h>


#define ID_TIMER                1

#define TWOPI                           (2 * 3.14159)


LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns

                                        PSTR szCmdLine, int iCr
```

```
{
        static TCHAR szAppName[] = TEXT ("Clock") ;
        HWND                 hwnd;
        MSG                  msg;
        WNDCLASS             wndclass ;

 wndclass.style                   = CS_HREDRAW | CS_VR
        wndclass.lpfnWndProc             = WndProc ;
        wndclass.cbClsExtra              = 0 ;
        wndclass.cbWndExtra              = 0 ;
        wndclass.hInstance               = hInstance ;
        wndclass.hIcon                   = NULL ;
        wndclass.hCursor                 = LoadCursor (NULL,
        wndclass.hbrBackground           = (HBRUSH) GetS
        wndclass.lpszMenuName            = NULL ;
        wndclass.lpszClassName           = szAppName ;

        if (!RegisterClass (&wndclass))
```

```
     {
          MessageBox (  NULL, TEXT ("Program requires Windo
                                                  szAppName, MB_IC
          return 0 ;
     }
     hwnd = CreateWindow (szAppName, TEXT ("Analog Clock
               WS_OVERLAPPEDWINDOW,
               CW_USEDEFAULT, CW_USEDEFAULT,
               CW_USEDEFAULT, CW_USEDEFAULT,
               NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
               TranslateMessage (&msg) ;
               DispatchMessage (&msg) ;
}
```

```
        return msg.wParam ;
}

void SetIsotropic (HDC hdc, int cxClient, int cyClient)
{
        SetMapMode (hdc, MM_ISOTROPIC) ;
        SetWindowExtEx (hdc, 1000, 1000, NULL) ;
        SetViewportExtEx (hdc, cxClient / 2, -cyClient / 2, NULL) ;
        SetViewportOrgEx (hdc, cxClient / 2,  cyClient / 2, NULL) ;
}

void RotatePoint (POINT pt[], int iNum, int iAngle)
{
        int   i ;
        POINT ptTemp ;

        for (i = 0 ; i < iNum ; i++)
        {
              ptTemp.x = (int) (pt[i].x * cos (TWOPI * iAngle / 360)
```

```
                                pt[i].y * sin (TWOPI * iAngle / 360)) ;

            ptTemp.y = (int) (pt[i].y * cos (TWOPI * iAngle / 360)
                                pt[i].x * sin (TWOPI * iAngle / 360)) ;

            pt[i] = ptTemp ;
   }
}

void DrawClock (HDC hdc)
{
        int   iAngle ;
        POINT pt[3] ;
   for (iAngle = 0 ; iAngle < 360 ; iAngle += 6)
   {
            pt[0].x      =   0 ;
            pt[0].y      = 900 ;

            RotatePoint (pt, 1, iAngle) ;
```

```
            pt[2].x      = pt[2].y = iAngle % 5 ? 33 : 100 ;

            pt[0].x -    = pt[2].x / 2 ;
            pt[0].y -    = pt[2].y / 2 ;

            pt[1].x      = pt[0].x + pt[2].x ;
            pt[1].y      = pt[0].y + pt[2].y ;

            SelectObject (hdc, GetStockObject (BLACK_BRUSH)) ;

            Ellipse (hdc, pt[0].x, pt[0].y, pt[1].x, pt[1].y) ;
   }
}

void DrawHands (HDC hdc, SYSTEMTIME * pst, BOOL fChange)
{
   static POINT pt[3][5] ={0, -150, 100, 0, 0, 600, -100, 0, 0, -15
                           0, -200,  50,    0, 0, 800, -50, 0, 0,-200,
```

```
                0,0,     0,     0, 0,  0,  0,  0, 0, 800 } ;

        int                     i, iAngle[3] ;

        POINT                     ptTemp[3][5] ;


        iAngle[0]     = (pst->wHour * 30) % 360 + pst->wMinute

        iAngle[1]     =  pst->wMinute  *  6 ;

        iAngle[2]     =  pst->wSecond  *  6 ;


        memcpy (ptTemp, pt, sizeof (pt)) ;

        for (i = fChange ? 0 : 2 ; i < 3 ; i++)

        {

             RotatePoint (ptTemp[i], 5, iAngle[i]) ;

             Polyline (hdc, ptTemp[i], 5) ;

        }

}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA

{

        static int       cxClient, cyClient ;
```

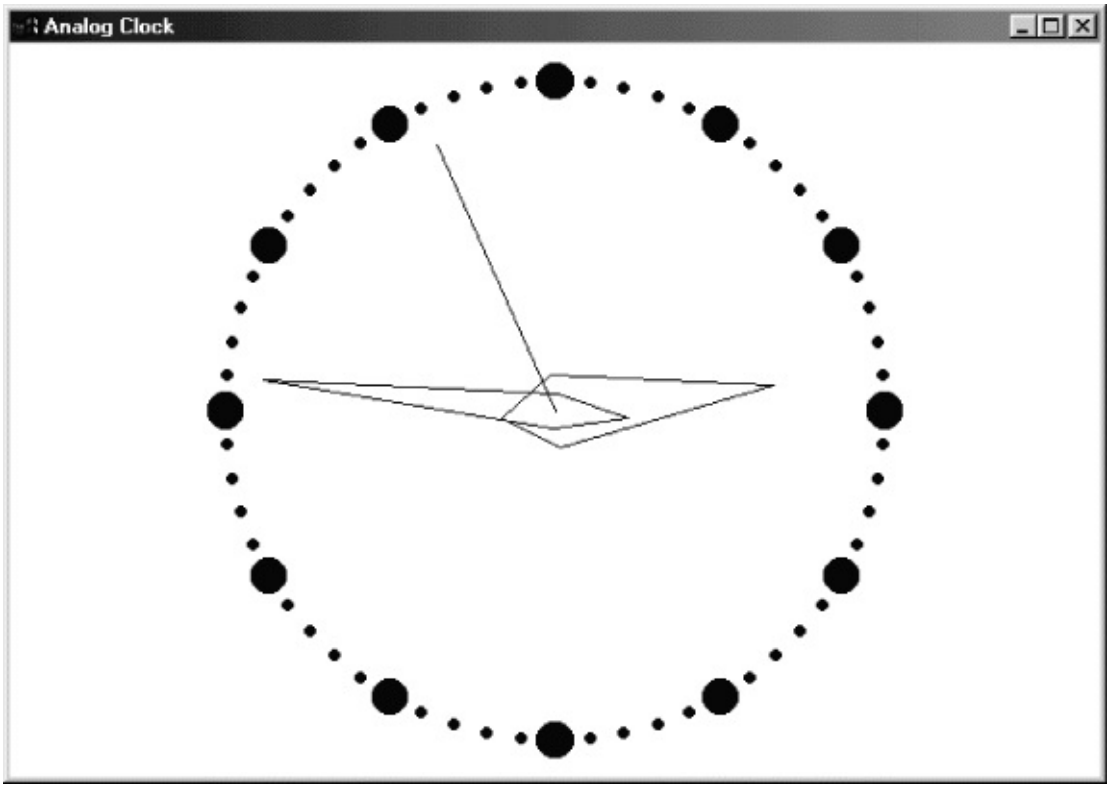
```
static SYSTEMTIME stPrevious ;
BOOL                                fChange ;
HDC                                 hdc ;
PAINTSTRUCT                         ps ;
SYSTEMTIME                          st ;

switch (message)
{
case   WM_CREATE :
       SetTimer (hwnd, ID_TIMER, 1000, NULL) ;
       GetLocalTime (&st) ;
       stPrevious = st ;
       return 0 ;

case   WM_SIZE :
      cxClient = LOWORD (lParam) ;
       cyClient = HIWORD (lParam) ;
       return 0 ;
```

```
case   WM_TIMER :
     GetLocalTime (&st) ;

     fChange =     st.wHour       !   = stPrevious.wHour ||
                                  st.wMinute !         = stPrevic

     hdc = GetDC (hwnd) ;

     SetIsotropic (hdc, cxClient, cyClient) ;

     SelectObject (hdc, GetStockObject (WHITE_PEN)) ;
     DrawHands (hdc, &stPrevious, fChange) ;

    SelectObject (hdc, GetStockObject (BLACK_PEN)) ;
     DrawHands (hdc, &st, TRUE) ;

     ReleaseDC (hwnd, hdc) ;
```

```
            stPrevious = st ;
            return 0 ;

        case   WM_PAINT :
            hdc = BeginPaint (hwnd, &ps) ;

            SetIsotropic (hdc, cxClient, cyClient) ;
            DrawClock    (hdc) ;
            DrawHands    (hdc, &stPrevious, TRUE) ;

            EndPaint (hwnd, &ps) ;
            return 0 ;

        case   WM_DESTROY :
            KillTimer (hwnd, ID_TIMER) ;
            PostQuitMessage (0) ;
            return 0 ;
}
        return DefWindowProc (hwnd, message, wParam, lParam)
```

```
}
```

CLOCK8-2



8-2 CLOCK

(isotropic)CLOCK.CSetIsotropicSetMapModeSetIsotropic1000
(0,0)1000

RotatePoint(0,100)12:0090(100,0)3:00

```
x' = x * cos (a) + y * sin (a)
```

```
y' = y * cos (a) - x * sin (a)
```

RotatePoint

DrawClock60(12:00)900(0,900)6l210033Ellipse

DrawHandsPOINTRotatePointWindowsPolyline
DrawHandsbChangeTRUE

WM_CREATEdtPrevious

WM_PAINTSetIsotropicDrawClockDrawHandsbChangeTRUE

WM_TIMERWndProc

GetPixel

WHATCLR 8-5RGB

8-5　WHATCLR

```
WHATCLR.C

/*------------------------------------------------------------------------
  WHATCLR.C -- Displays Color Under Cursor
                          (c) Charles Petzold, 1998
------------------------------------------------------------------------*/

#include <windows.h>

#define ID_TIMER    1
```

```
void FindWindowSize (int *, int *) ;
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                        PSTR szCmdLine, int iCmdSh
{
        static TCHAR szAppName[] = TEXT ("WhatClr") ;
        HWND                        hwnd ;
        int                       cxWindow, cyWindow ;
        MSG                         msg ;
        WNDCLASS                  wndclass ;

        wndclass.style                    = CS_HREDRAW | CS_V
        wndclass.lpfnWndProc                = WndProc ;
        wndclass.cbClsExtra                 = 0 ;
        wndclass.cbWndExtra                 = 0 ;
        wndclass.hInstance                  = hInstance ;
        wndclass.hIcon                    = LoadIcon (NULL, IDI_
        wndclass.hCursor                   = LoadCursor (NULL,
```

```
        wndclass.hbrBackground                = (HBRUSH) GetStoc
wndclass.lpszMenuName  = NULL ;
        wndclass.lpszClassName                = szAppName ;

        if (!RegisterClass (&wndclass))
        {
             MessageBox (  NULL, TEXT ("This program requires W
                                                  szAppName, MB_IC
             return 0 ;
        }

        FindWindowSize (&cxWindow, &cyWindow) ;
        hwnd = CreateWindow (szAppName, TEXT ("What Color")
          WS_OVERLAPPED | WS_CAPTION | WS_SYSMENU | WS_B
           CW_USEDEFAULT, CW_USEDEFAULT,
           cxWindow, cyWindow,
           NULL, NULL, hInstance, NULL) ;
```

```
        ShowWindow (hwnd, iCmdShow) ;
        UpdateWindow (hwnd) ;

        while (GetMessage (&msg, NULL, 0, 0))
        {
                TranslateMessage (&msg) ;
                DispatchMessage (&msg) ;
        }
        return msg.wParam ;
}

void FindWindowSize (int * pcxWindow, int * pcyWindow)
{
        HDC                hdcScreen ;
       TEXTMETRIC    tm ;

        hdcScreen = CreateIC (TEXT ("DISPLAY"), NULL, NULL, NU
        GetTextMetrics (hdcScreen, &tm) ;
        DeleteDC (hdcScreen) ;
```

```
        * pcxWindow = 2 *     GetSystemMetrics (SM_CXBORDER)
                              12 * tm.tmAveCharWidth ;
        * pcyWindow = 2 *     GetSystemMetrics (SM_CYBORDER)
                            GetSystemMetrics (SM_CYCAPTION) +
                              2 * tm.tmHeight ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
        static COLORREF       cr, crLast ;
        static HDC              hdcScreen ;
        HDC                     hdc ;
        PAINTSTRUCT           ps ;
        POINT                   pt ;
        RECT                rc ;
        TCHAR                   szBuffer [16] ;

        switch (message)
        {
```

```
        case   WM_CREATE:
               hdcScreen = CreateDC (TEXT ("DISPLAY"), NULL, NUL
               SetTimer (hwnd, ID_TIMER, 100, NULL) ;
               return 0 ;

        case   WM_TIMER:
               GetCursorPos (&pt) ;
               cr = GetPixel (hdcScreen, pt.x, pt.y) ;

               SetPixel (hdcScreen, pt.x, pt.y, 0) ;

               if (cr != crLast)
               {
                        crLast = cr ;
                        InvalidateRect (hwnd, NULL, FALSE) ;
               }
               return 0 ;

        case   WM_PAINT:
```

```
            hdc = BeginPaint (hwnd, &ps) ;

            GetClientRect (hwnd, &rc) ;

            wsprintf (szBuffer, TEXT ("  %02X %02X %02X  "),
                  GetRValue (cr), GetGValue (cr), GetBValue (cr)) ;

            DrawText (hdc, szBuffer, -1, &rc,
               DT_SINGLELINE | DT_CENTER | DT_VCENTER) ;

            EndPaint (hwnd, &ps) ;
            return 0 ;

case   WM_DESTROY:
            DeleteDC (hdcScreen) ;
            KillTimer (hwnd, ID_TIMER) ;
            PostQuitMessage (0) ;
            return 0 ;
```

```
        }

        return DefWindowProc (hwnd, message, wParam, lParam)

}
```

WHATCLRWinMainWHATCLRRGBCreateWindowWS_BORDER
WHATCLRCreateICGetSystemMetricsCreateWindow

WHATCLRWM_CREATECreateDCWM_TIMER
WM_PAINTRGB

CreateDC

CHECKERxxCHECKER1CHECKER2CHECKER3
ChildProc

ChildProc(WndProc)GetParent

```
hwndParent = GetParent (hwnd) ;
```

hwnd

```
SendMessage (hwndParent, message, wParam, lParam) ;
```

messageWM_USERWM_
lParam0

WindowsRecalculate
CreateWindowWindowsWM_COMMAND

CreateWindowMoveWindow

RegisterClassWindowsCreateWindow
WindowsCreateWindowCreateWindowWindows

Tab

Windows/Platform
Windows

/Platform SDK/User Interfac
Controls/Common ControlsWindowsWindows

BTNLOOKbutton look9-1BTNLOOK1010

9-1 BTNLOOK

```
BTNLOOK.C
/*------------------------------------------------------------------------
  BTNLOOK.C -- Button Look Program
                                (c) Charles Petzold, 1998
------------------------------------------------------------------------*/
#include <windows.h>
struct
{
        int             iStyle ;
        TCHAR *         szText ;
}
button[] =
{
        BS_PUSHBUTTON,      TEXT ("PUSHBUTTON"),
        BS_DEFPUSHBUTTON,   TEXT ("DEFPUSHBUTTON"),
```

```
        BS_CHECKBOX,          TEXT ("CHECKBOX"),
        BS_AUTOCHECKBOX,      TEXT ("AUTOCHECKBOX"),
        BS_RADIOBUTTON,       TEXT ("RADIOBUTTON"),
        BS_3STATE,            TEXT ("3STATE"),
        BS_AUTO3STATE,        TEXT ("AUTO3STATE"),
        BS_GROUPBOX,          TEXT ("GROUPBOX"),
        BS_AUTORADIOBUTTON,   TEXT ("AUTORADIO"),
        BS_OWNERDRAW,         TEXT ("OWNERDRAW")
} ;

#define NUM (sizeof button / sizeof button[0])
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                         PSTR szCmdLine, int iCmdSh
{
        static TCHAR szAppName[] = TEXT ("BtnLook") ;
        HWND                 hwnd ;
        MSG                  msg ;
        WNDCLASS             wndclass ;
```

```
     wndclass.style                    = CS_HREDRAW |
     wndclass.lpfnWndProc                = WndProc ;
     wndclass.cbClsExtra                 = 0 ;
     wndclass.cbWndExtra                 = 0 ;
     wndclass.hInstance                  = hInstance ;
     wndclass.hIcon                    = LoadIcon (NULL
     wndclass.hCursor          = LoadCursor (NULL, IDC_ARRO
     wndclass.hbrBackground          = (HBRUSH) GetStockO
     wndclass.lpszMenuName             = NULL ;
     wndclass.lpszClassName            = szAppName ;

     if (!RegisterClass (&wndclass))
{
          MessageBox (  NULL, TEXT ("This program requires W
                                        szAppName, MB_IC
          return 0 ;
     }
```

```
        hwnd = CreateWindow (szAppName, TEXT ("Button Look'
                             WS_OVERLAPPEDWINDOW,
                             CW_USEDEFAULT, CW_USEDEFAULT,
                            CW_USEDEFAULT, CW_USEDEFAULT,
                             NULL, NULL, hInstance, NULL) ;

        ShowWindow (hwnd, iCmdShow) ;
        UpdateWindow (hwnd) ;

        while (GetMessage (&msg, NULL, 0, 0))
       {
  TranslateMessage (&msg) ;
                    DispatchMessage (&msg) ;
        }
        return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
        static HWND          hwndButton[NUM] ;
```

```
        static RECT          rect ;
        static TCHAR         szTop[]      = TEXT ("message   wPara
                             szUnd[]      = TEXT ("_______   _____
                             szFormat[]   = TEXT ("%-16s%04X-%(
                             szBuffer[50] ;
        static int           cxChar, cyChar ;
        HDC                          hdc ;
        PAINTSTRUCT            ps ;
int                          i ;

       switch (message)
        {
        case   WM_CREATE :
              cxChar = LOWORD (GetDialogBaseUnits ()) ;
              cyChar = HIWORD (GetDialogBaseUnits ()) ;

              for (i = 0 ; i < NUM ; i++)
                             hwndButton[i] =CreateWindow ( TEXT(
```

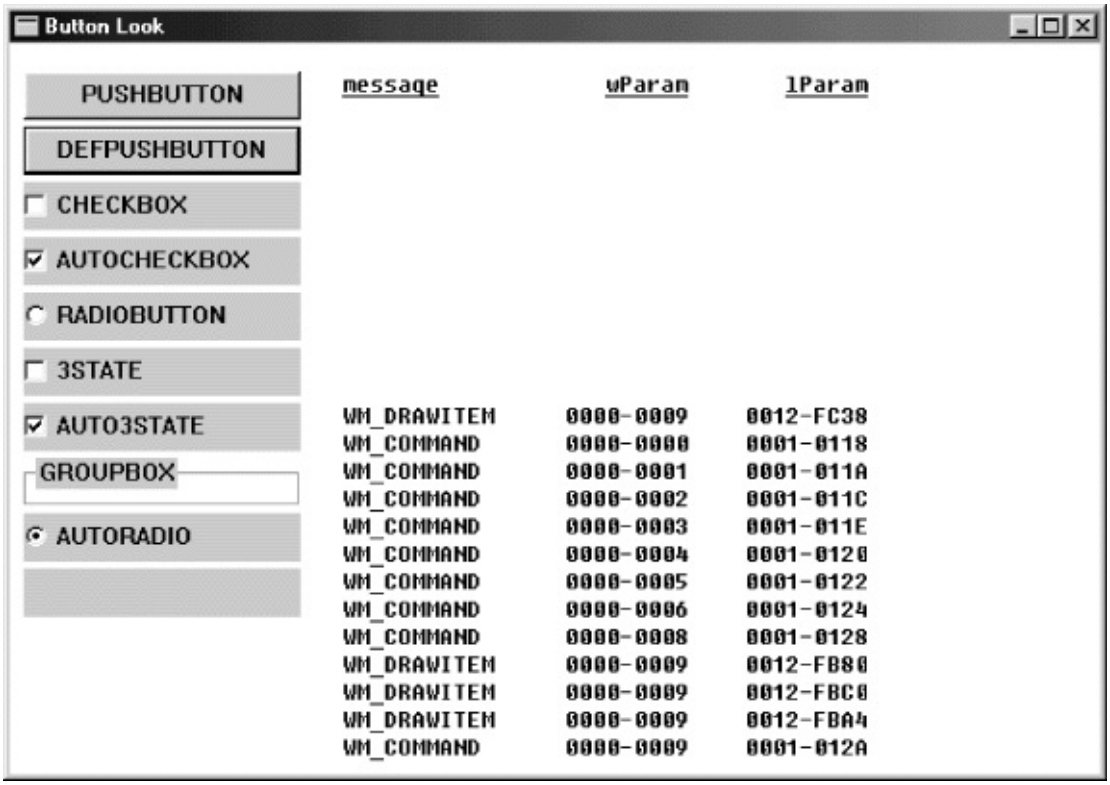
```
WS_CHILD | WS_VISIBLE | button[i].iStyle,
 cxChar, cyChar * (1 + 2 * i),
 20 * cxChar, 7 * cyChar / 4,
                    hwnd, (HMENU) i,
          ((LPCREATESTRUCT) lParam)->hInstance, NULL) ;
             return 0 ;

        case   WM_SIZE :
             rect.left            = 24 * cxChar ;
             rect.top             =  2 * cyChar ;
             rect.right           = LOWORD (lParam) ;
            rect.bottom           = HIWORD (lParam) ;
             return 0 ;
        case   WM_PAINT :
             InvalidateRect (hwnd, &rect, TRUE) ;

             hdc = BeginPaint (hwnd, &ps) ;
             SelectObject (hdc, GetStockObject (SYSTEM_FIXED_F
             SetBkMode (hdc, TRANSPARENT) ;
```

```
          TextOut (hdc, 24 * cxChar, cyChar, szTop, lstrlen (szT
          TextOut (hdc, 24 * cxChar, cyChar, szUnd, lstrlen (szU

          EndPaint (hwnd, &ps) ;
          return 0 ;

     case   WM_DRAWITEM :
     case   WM_COMMAND :
          ScrollWindow (hwnd, 0, -cyChar, &rect, &rect) ;

          hdc = GetDC (hwnd) ;
          SelectObject (hdc, GetStockObject (SYSTEM_FIXED_F

          TextOut(      hdc, 24 * cxChar, cyChar * (rect.bottom /
                              szBuffer,
                              wsprintf (szBuffer, szFormat,
                                   message == WM_DRAWITEM
                         TEXT ("WM_COMMAND"),
```

```
                                        HIWORD (wParam), LOWORD
                                        HIWORD (lParam), LOWORD


            ReleaseDC (hwnd, hdc) ;
            ValidateRect (hwnd, &rect) ;
            break ;


        case   WM_DESTROY :
            PostQuitMessage (0) ;
            return 0 ;
   }
        return DefWindowProc (hwnd, message, wParam, lParam)
}
```

WM_COMMANDWndProcBTNLOOKWndProcwParamlParam9-1

BS_OWNERDRAWlParamWM_DRAWITEMlParam DRAWITEMSTRUCTBTNLOOKowner



9-1 BTNLOOK

BTNLOOKbutton10BS10WndProcWM_CREATE forCreateWindow

Class name	TEXT ("button")
Window text	button[i].szText
Window style	WS_CHILD | WS_VISIBLE | button[i].iStyle
x positionx	cxChar
y positiony	cyChar * (1 + 2 * i)
Width	20 * xChar

Height	7 * yChar / 4
Parent window	hwnd
Child window IDID	(HMENU) i
Instance handle	((LPCREATESTRUCT) lParam) -> hInstance
Extra parameters	NULL

WS_CHILDWS_VISIBLEbutton10BS_PUSHBUTTON
BS_DEFPUSHBUTTONxy
GetDialogBaseUnits32GetDialogBaseUnitsGetTextMetrics
GetDialogBaseUnits

IDWM_COMMANDIDIDCreateWindowIDHMENU

CreateWindowWM_CREATElParamCREATESTRUCT
hInstancelParamCREATESTRUCThInstance

WindowshInstWinMainWinMain

```
hInst = hInstance ;
```

CHECKER3GetWindowLong

```
GetWindowLong (hwnd, GWL_HINSTANCE)
```

CreateWindowWindowsBS_OWNERDRAW
Windows

BTNLOOKWM_COMMANDBTNLOOK
WM_COMMANDwParamlParam

LOWORD (wParam) HIWORD (wParam) lParam	ID

16Windows32

IDCreateWindowBTNLOOKID0910WindowsCreateWindow

Windows

9-1

BN_CLICKED	0
BN_PAINT	1
BN_HILITE or BN_PUSHED	2
BN_UNHILITE or BN_UNPUSHED	3
BN_DISABLE	4

BN_DOUBLECLICKED or BN_DBLCLK	5
BN_SETFOCUS	6
BN_KILLFOCUS	7

14BS_USERBUTTONBS_OWNERDRAW67BS_NOTIFY
5BS_RADIOBUTTONBS_AUTORADIOBUTTONBS_OWNERDRAW
BS_NOTIFY

SpacebarSpacebar

BTNLOOKWMWINUSER.H8BM

9-2

BM_GETCHECK	0x00F0
BM_SETCHECK	0x00F1
BM_GETSTATE	0x00F2
BM_SETSTATE	0x00F3

BM_SETSTYLE	0x00F4
BM_CLICK	0x00F5
BM_GETIMAGE	0x00F6
BM_SETIMAGE	0x00F7

BM_GETCHECKBM_SETCHECKBM_GETSTATEBM_SETSTATE SpacebarBM_SETSTYLE

IDIDID

```
id = GetWindowLong (hwndChild, GWL_ID) ;
```

CHECKER3 SetWindowLongWindowsGWL_ID

```
id = GetDlgCtrlID (hwndChild) ;
```

Dlg

ID

```
hwndChild = GetDlgItem (hwndParent, id) ;
```

BTNLOOKCreateWindowCreateWindowMoveWindow

/BS_PUSHBUTTONBS_DEFPUSHBUTTON
BS_DEFPUSHBUTTONDEFBS_PUSHBUTTON
BS_DEFPUSHBUTTONBS_DEFPUSHBUTTON

7/4BTNLOOK

WM_COMMANDBN_CLICKEDSpacebar

BM_SETSTATE

```
SendMessage (hwndButton, BM_SETSTATE, 1, 0) ;
```

```
SendMessage (hwndButton, BM_SETSTATE, 0, 0) ;
```

hwndButtonCreateWindow

BM_GETSTATETRUEFALSE/
BM_SETCHECKBM_GETCHECK

BS_LEFTTEXTBS_RIGHT

BS_CHECKBOXBS_AUTOCHECKBOXBS_CHECKBOX
BM_SETCHECKwParam10BM_GETCHECK
WM_COMMANDX

```
SendMessage       ((HWND) lParam, BM_SETCHECK, (WPARAM)

                !SendMessage ((HWND) lParam, BM_GETCHECK, 0, 0
```

SendMessage!lParamWM_COMMANDBM_GETCHECK

BM_SETCHECKBS_CHECKBOX

```
SendMessage (hwndButton, BM_SETCHECK, 1, 0) ;
```

BS_AUTOCHECKBOXWM_COMMANDBM_GETCHECK

```
iCheck = (int) SendMessage (hwndButton, BM_GETCHECK, 0, 0)
```

iCheckTRUEiCheckFALSE0

BS_3STATEBS_AUTO3STATEwParam2WM_SETCHECK

CreateWindowWM_COMMAND2

BS_RADIOBUTTONBS_AUTORADIOBUTTON

WM_COMMANDwParam1BM_SETCHECK

```
SendMessage (hwndButton, BM_SETCHECK, 1, 0) ;
```

wParam0BM_SETCHECK

```
SendMessage (hwndButton, BM_SETCHECK, 0, 0) ;
```

BS_GROUPBOXWM_COMMAND

SetWindowText

```
SetWindowText (hwnd, pszString) ;
```

hwndpszStringnull

```
iLength = GetWindowText (hwnd, pszBuffer, iMaxLength) ;
```

iMaxLengthpszBuffer

```
iLength = GetWindowTextLength (hwnd) ;
```

WS_VISIBLEShowWindow

```
ShowWindow (hwndChild, SW_SHOWNORMAL) ;
```

WS_VISIBLEShowWindowShowWindow

```
ShowWindow (hwndChild, SW_HIDE) ;
```

```
IsWindowVisible (hwndChild) ;
```

```
EnableWindow (hwndChild, FALSE) ;
```

```
EnableWindow (hwndChild, TRUE) ;
```

```
IsWindowEnabled (hwndChild) ;
```

SpacebarSpacebar

WindowsWM_KILLFOCUSwParam
WindowsWM_SETFOCUSwParamwParamNULL

WM_KILLFOCUShwndChildCreateWindow

```
case      WM_KILLFOCUS :

            for (  i = 0 ; i < NUM ; i++)

                        if (hwndChild [i] == (HWND) wParam)

            {

                              SetFocus (hwnd) ;

                              break ;

            }
```

```
        return 0 ;
```

SetFocus

```
case WM_KILLFOCUS :
   if (hwnd == GetParent ((HWND) wParam))
                    SetFocus (hwnd) ;
        return 0 ;
```

SpacebarTabCOLORS1

9-1Windows

```
wndclass.hbrBackground = (HBRUSH) GetStockObject (WHITE_B
```

GDI

Windows

Windows29GetSysColorSetSysColorsWindowsSetSysColors
Windows

WindowsMicrosoft
Windows 98WIN.INIWIN.INI29GetSysColorSetSysColors
025529GetSysColorSetSysColorsWIN.INICOLOR_

GetSysColorSetSysColors	WIN.INI	RGB
COLOR_SCROLLBAR	Scrollbar	C0-C0-C0
COLOR_BACKGROUND	Background	00-80-80
COLOR_ACTIVECAPTION	ActiveTitle	00-00-80
COLOR_INACTIVECAPTION	InactiveTitle	80-80-80
COLOR_MENU	Menu	C0-C0-C0
COLOR_WINDOW	Window	FF-FF-FF
COLOR_WINDOWFRAME	WindowFrame	00-00-00
COLOR_MENUTEXT	MenuText	C0-C0-C0
COLOR_WINDOWTEXT	WindowText	00-00-00
COLOR_CAPTIONTEXT	TitleText	FF-FF-FF

COLOR_ACTIVEBORDER	ActiveBorder	C0-C0-C0
COLOR_INACTIVEBORDER	InactiveBorder	C0-C0-C0
COLOR_APPWORKSPACE	AppWorkspace	80-80-80
COLOR_HIGHLIGHT	Highlight	00-00-80
COLOR_HIGHLIGHTTEXT	HighlightText	FF-FF-FF
COLOR_BTNFACE	ButtonFace	C0-C0-C0
COLOR_BTNSHADOW	ButtonShadow	80-80-80
COLOR_GRAYTEXT	GrayText	80-80-80
COLOR_BTNTEXT	ButtonText	00-00-00
COLOR_INACTIVECAPTIONTEXT	InactiveTitleText	C0-C0-C0
COLOR_BTNHIGHLIGHT	ButtonHighlight	FF-FF-FF
COLOR_3DDKSHADOW	ButtonDkShadow	00-00-00
COLOR_3DLIGHT	ButtonLight	C0-C0-C0

COLOR_INFOTEXT	InfoText	00-00-00
COLOR_INFOBK	InfoWindow	FF-FF-FF
[no identifier; use value 25]	ButtonAlternateFace	B8-B4-B8
COLOR_HOTLIGHT	HotTrackingColor	00-00-FF
COLOR_GRADIENTACTIVECAPTION	GradientActiveTitle	00-00-80
COLOR_GRADIENTINACTIVECAPTION	GradientInactiveTitle	80-80-80

29

COLOR_BACKGROUNDWindowsWindows
Windows 3.013

COLOR_BTNFACECOLOR_BTNSHADOW
COLOR_BTNTEXTCOLOR_WINDOWTEXT

COLOR_BTNFACE

```
wndclass.hbrBackground = (HBRUSH) (COLOR_BTNFACE + 1) ;
```

BTNLOOKWNDCLASShbrBackgroundWindowsWindows
WNDCLASShbrBackground1NULLWindows
COLOR_BTNFACETextOutWindows

hbrBackgroundSetTextColorSetBkColor

```
SetBkColor (hdc, GetSysColor (COLOR_BTNFACE)) ;

SetTextColor (hdc, GetSysColor (COLOR_WINDOWTEXT)) ;
```

```
case       WM_SYSCOLORCHANGE:

          InvalidateRect (hwnd, NULL, TRUE) ;

          break ;
```

WM_CTLCOLORBTN

SetSysColorsWindows

WM_CTLCOLORBTNWindows16
WM_CTLCOLOR

WM_CTLCOLORBTNwParamlParam
WM_CTLCOLORBTN

- SetTextColor
- SetBkColor
-

WM_CTLCOLORBTNWM_CTLCOLORBTN

WM_CTLCOLORBTN

BS_OWNERDRAW9-2

9-2　OWNDRAW

```
OWNDRAW.C
/*------------------------------------------------------------------------
  OWNDRAW.C --    Owner-Draw Button Demo Program
                                (c) Charles Petzold, 1996
------------------------------------------------------------------------*/
#include <windows.h>

#define ID_SMALLER                              1
#define ID_LARGER                               2
#define BTN_WIDTH                       (      8 * cxChar)
#define BTN_HEIGHT                      (      4 * cyChar)

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
HINSTANCE hInst ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                PSTR szCmdLine, int iCmdShow)
```

```
{
     static TCHAR szAppName[] = TEXT ("OwnDraw") ;
     MSG                         msg ;
     HWND                        hwnd ;
     WNDCLASS                    wndclass ;

    hInst = hInstance ;
     wndclass.style                    = CS_HREDRAW | CS_V
     wndclass.lpfnWndProc                = WndProc ;
     wndclass.cbClsExtra                 = 0 ;
     wndclass.cbWndExtra                 = 0 ;
     wndclass.hInstance                  = hInstance ;
     wndclass.hIcon                      = LoadIcon (NULL, IDI_
     wndclass.hCursor                    = LoadCursor (NULL,
     wndclass.hbrBackground          = (HBRUSH) GetStockO
     wndclass.lpszMenuName           = szAppName ;
     wndclass.lpszClassName          = szAppName ;

     if (!RegisterClass (&wndclass))
```

```
{
          MessageBox (  NULL, TEXT ("This program requires W
                                                  szAppName, MB_IC
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, TEXT ("Owner-Draw
               WS_OVERLAPPEDWINDOW,
               CW_USEDEFAULT, CW_USEDEFAULT,
              CW_USEDEFAULT, CW_USEDEFAULT,
               NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
```

```
   }
  return msg.wParam ;
}

void Triangle (HDC hdc, POINT pt[])
{
        SelectObject (hdc, GetStockObject (BLACK_BRUSH)) ;
        Polygon (hdc, pt, 3) ;
        SelectObject (hdc, GetStockObject (WHITE_BRUSH)) ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
        static HWND                hwndSmaller, hwndLarger ;
        static int                  cxClient, cyClient, cxChar, cyCh
        int                                 cx, cy ;
        LPDRAWITEMSTRUCT pdis ;
        POINT                                 pt[3] ;
        RECT                                      rc ;
```

```
switch (message)
 {
 case   WM_CREATE :
        cxChar = LOWORD (GetDialogBaseUnits ()) ;
        cyChar = HIWORD (GetDialogBaseUnits ()) ;

                    // Create the owner-draw pushbuttons

        hwndSmaller = CreateWindow (TEXT ("button"), TEXT
                  WS_CHILD | WS_VISIBLE | BS_OWNERDRAW,
                   0, 0, BTN_WIDTH, BTN_HEIGHT,
                   hwnd, (HMENU) ID_SMALLER, hInst, NULL) ;

        hwndLarger  = CreateWindow (TEXT ("button"), TEXT
                  WS_CHILD | WS_VISIBLE | BS_OWNERDRAW,
                   0, 0, BTN_WIDTH, BTN_HEIGHT,
                   hwnd, (HMENU) ID_LARGER, hInst, NULL) ;
        return 0 ;
```

```
case   WM_SIZE :
     cxClient = LOWORD (lParam) ;
     cyClient = HIWORD (lParam) ;
                    // Move the buttons to the new center

MoveWindow    (      hwndSmaller, cxClient / 2 - 3 *    BTN
               cyClient / 2 -    BTN_HEIGHT / 2,
                     BTN_WIDTH, BTN_HEIGHT, TRUE) ;
     MoveWindow (  hwndLarger,   cxClient / 2 + BTN_WID
                          BTN_WIDTH, BTN_HEIGHT, TRUE
     return 0 ;

case   WM_COMMAND :
     GetWindowRect (hwnd, &rc) ;

                    // Make the window 10% smaller or lar
```

```
        switch (wParam)
        {
        case   ID_SMALLER :
                        rc.left      += cxClient / 20 ;
                        rc.right      -= cxClient / 20 ;
                        rc.top       += cyClient / 20 ;
                        rc.bottom     -= cyClient / 20 ;
                        break ;

        case   ID_LARGER :
                        rc.left      -= cxClient / 20 ;
                        rc.right      += cxClient / 20 ;
                        rc.top       -= cyClient / 20 ;
                        rc.bottom     += cyClient / 20 ;
                        break ;
    }

        MoveWindow (  hwnd, rc.left, rc.top, rc.right  - rc.left,
                                                rc.bottom - rc
```

```
        return 0 ;

    case   WM_DRAWITEM :
        pdis = (LPDRAWITEMSTRUCT) lParam ;

                        // Fill area with white and frame it blac

        FillRect            (pdis->hDC, &pdis->rcItem,
                    (HBRUSH) GetStockObject (WHITE_BRUSH)

        FrameRect (   pdis->hDC, &pdis->rcItem,
                        (      HBRUSH) GetStockObject (BLACK_
                        //                  Draw inward and outwar
        cx =          pdis->rcItem.right  - pdis->rcItem.left ;
        cy =          pdis->rcItem.bottom - pdis->rcItem.top  ;

        switch (pdis->CtlID)
        {
        case   ID_SMALLER :
```

```
                    pt[0].x = 3 * cx / 8 ;  pt[0].y = 1 * cy /
                    pt[1].x = 5 * cx / 8 ;  pt[1].y = 1 * cy /
                    pt[2].x = 4 * cx / 8 ;  pt[2].y = 3 * cy /

                    Triangle (pdis->hDC, pt) ;

                    pt[0].x = 7 * cx / 8 ;  pt[0].y = 3 * cy /
                    pt[1].x = 7 * cx / 8 ;  pt[1].y = 5 * cy /
                    pt[2].x = 5 * cx / 8 ;  pt[2].y = 4 * cy /

                    Triangle (pdis->hDC, pt) ;

                    pt[0].x = 5 * cx / 8 ;  pt[0].y = 7 * cy /
                    pt[1].x = 3 * cx / 8 ;  pt[1].y = 7 * cy /
                   pt[2].x = 4 * cx / 8 ;  pt[2].y = 5 * cy / 8

                    Triangle (pdis->hDC, pt) ;
```

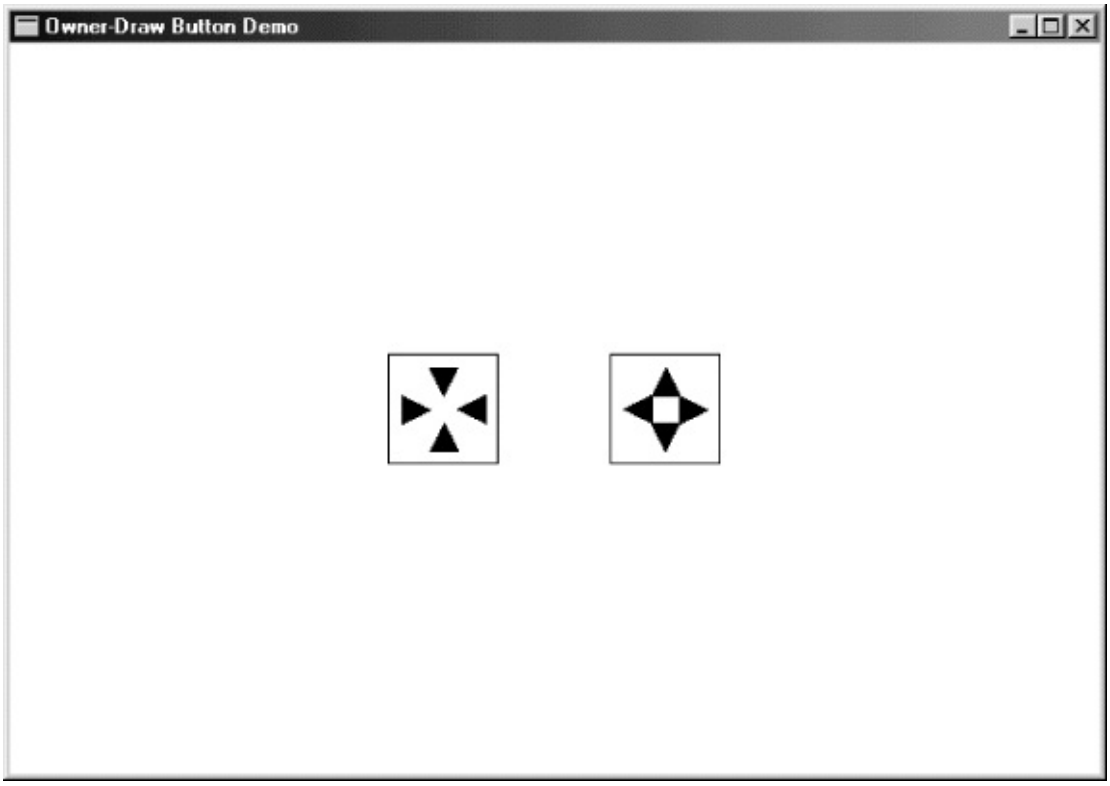
```
                    pt[0].x = 1 * cx / 8 ;  pt[0].y = 5 * cy /
                    pt[1].x = 1 * cx / 8 ;  pt[1].y = 3 * cy /
                    pt[2].x = 3 * cx / 8 ;  pt[2].y = 4 * cy /

                    Triangle (pdis->hDC, pt) ;
                    break ;

                case ID_LARGER :
                    pt[0].x = 5 * cx / 8 ;  pt[0].y = 3 * cy / 8 ;
                    pt[1].x = 3 * cx / 8 ;  pt[1].y = 3 * cy / 8 ;
                    pt[2].x = 4 * cx / 8 ;  pt[2].y = 1 * cy / 8 ;

                    Triangle (pdis->hDC, pt) ;

                    pt[0].x = 5 * cx / 8 ;  pt[0].y = 5 * cy / 8 ;
                    pt[1].x = 5 * cx / 8 ;  pt[1].y = 3 * cy / 8 ;
                    pt[2].x = 7 * cx / 8 ;  pt[2].y = 4 * cy / 8 ;

                    Triangle (pdis->hDC, pt) ;
```

```
                pt[0].x = 3 * cx / 8 ;  pt[0].y = 5 * cy / 8 ;
                pt[1].x = 5 * cx / 8 ;  pt[1].y = 5 * cy / 8 ;
                pt[2].x = 4 * cx / 8 ;  pt[2].y = 7 * cy / 8 ;

                Triangle (pdis->hDC, pt) ;
                pt[0].x = 3 * cx / 8 ;  pt[0].y = 3 * cy / 8 ;
                pt[1].x = 3 * cx / 8 ;  pt[1].y = 5 * cy / 8 ;
                pt[2].x = 1 * cx / 8 ;  pt[2].y = 4 * cy / 8 ;

                Triangle (pdis->hDC, pt) ;
                break ;
        }

                // Invert the rectangle if the button is selec

        if (pdis->itemState & ODS_SELECTED)
                        InvertRect (pdis->hDC, &pdis->rcItem
```

```
                    // Draw a focus rectangle if the button has

            if (pdis->itemState & ODS_FOCUS)
            {
                        pdis->rcItem.left  += cx / 16 ;
                        pdis->rcItem.top   += cy / 16 ;
                        pdis->rcItem.right -= cx / 16 ;
                        pdis->rcItem.bottom-= cy / 16 ;

                        DrawFocusRect (pdis->hDC, &pdis->rc
            }
            return 0 ;

 case   WM_DESTROY :
            PostQuitMessage (0) ;
            return 0 ;
       }
        return DefWindowProc (hwnd, message, wParam, lParam)
}
```

9-21010

BS_ICONBS_BITMAPBM_SETIMAGEBS_OWNERDRAW



9-2 OWNDRAW

WM_CREATEOWNDRAWBS_OWNERDRAW84VGA64
64WM_SIZEMoveWindowOWNDRAW

WM_COMMANDWM_COMMANDOWNDRAW
GetWindowRectRECTOWNDRAW
MoveWindowWM_SIZE

BS_OWNERDRAWWM_DRAWITEM

WM_DRAWITEMlParamDRAWITEMSTRUCTOWNDRAWpdis
hDC rcIte

FillRectFrameRectOWNDRAWWM_DRAWITEMPolygon
OWNDRAW4

DRAWITEMSTRUCTitemState1ODS_SELECTEDOWNDRAW
InvertRectitemStateODS_FOCUSOWNDRAW
DrawFocusRect

WindowsDRAWITEMSTRUCTGDI

CreateWindowstaticWM_COMMAND

WM_NCHITTESTHTTRANSPARENTWindowsWindows
WM_NCHITTESTDefWindowProc

RECTFRAME

SS_BLACKRECT	SS_BLACKFRAME
SS_GRAYRECT	SS_GRAYFRAME
SS_WHITERECT	SS_WHITEFRAME

BLACKGRAYWHITE9-4

BLACK	COLOR_3DDKSHADOW
GRAY	COLOR_BTNSHADOW
WHITE	COLOR_BTNHIGHLIGHT

CreateWindowxySS_ETCHEDHORZSS_ETCHEDVERT
SS_ETCHEDFRAME

SS_LEFTSS_RIGHTSS_CENTERCreateWindow
SetWindowTextDrawTextDT_WORDBREAKDT_NOCLIP
DT_EXPANDTABS

COLOR_BTNFACECOLOR_WINDOWTEXTWM_CTLCOLORSTATIC
SetTextColorSetBkColorCOLORS1

SS_ICONSS_USERITEM

SYSMETSWS_VSCROLLWS_HSCROLL
scrollbarSBS_VERTSBS_HORZ

WM_COMMANDWM_VSCROLLWM_HSCROLL
lParam0wParam

WindowsCreateWindowMoveWindow

GetSystemMetrics

```
GetSystemMetrics (SM_CYHSCROLL) ;
```

```
GetSystemMetrics (SM_CXVSCROLL) ;
```

WindowsSBS_LEFTALIGNSBS_RIGHTALIGNSBS_TOP
SBS_BOTTOMALIGN

```
SetScrollRange (hwndScroll, SB_CTL, iMin, iMax, bRedraw) ;

SetScrollPos (hwndScroll, SB_CTL, iPos, bRedraw) ;

SetScrollInfo (hwndScroll, SB_CTL, &si, bRedraw) ;
```

SB_VERTSB_HORZ

COLOR_SCROLLBARCOLOR_BTNFACE
COLOR_BTNHILIGHTCOLOR_BTNSHADOWCOLOR_BTNTEXT
COLOR_DKSHADOWCOLOR_BTNLIGHTCOLOR_BTNFACE
COLOR_BTNHIGHLIGHT

WM_CTLCOLORSCROLLBAR

COLORS1

COLORS19-3COLORS1Red

9-3 COLORS1

```
COLORS1.C

/*------------------------------------------------------------------

  COLORS1.C -- Colors Using Scroll Bars
```

```
                    (c) Charles Petzold, 1998
----------------------------------------------------------------------*/
#include <windows.h>
LRESULT CALLBACK WndProc          (HWND, UINT, WPARAM, LPA
LRESULT CALLBACK ScrollProc(HWND, UINT, WPARAM, LPARAM) ;

int     idFocus ;
WNDPROC OldScroll[3] ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                    PSTR szCmdLine, int iCmdSh
{
          static TCHAR szAppName[] = TEXT ("Colors1") ;
          HWND                 hwnd ;
          MSG                  msg ;
          WNDCLASS             wndclass ;

         wndclass.style                  = CS_HREDRAW | CS_VR
          wndclass.lpfnWndProc                = WndProc ;
```

```
        wndclass.cbClsExtra                    = 0 ;
        wndclass.cbWndExtra                    = 0 ;
        wndclass.hInstance                     = hInstance ;
        wndclass.hIcon                         = LoadIcon (NULL, IDI_
        wndclass.hCursor                       = LoadCursor (NULL,
        wndclass.hbrBackground                 = CreateSolidBrush (0)
        wndclass.lpszMenuName                  = NULL ;
        wndclass.lpszClassName                 = szAppName ;

        if (!RegisterClass (&wndclass))
        {
             MessageBox (  NULL, TEXT ("This program requires W
                                                szAppName, MB_IC
             return 0 ;
        }
  hwnd = CreateWindow (szAppName, TEXT ("Color Scroll"),
                  WS_OVERLAPPEDWINDOW,
                  CW_USEDEFAULT, CW_USEDEFAULT,
                  CW_USEDEFAULT, CW_USEDEFAULT,
```

```
                NULL, NULL, hInstance, NULL) ;

          ShowWindow (hwnd, iCmdShow) ;
          UpdateWindow (hwnd) ;

          while (GetMessage (&msg, NULL, 0, 0))
          {
               TranslateMessage (&msg) ;
               DispatchMessage  (&msg) ;
          }
          return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
          static COLORREF crPrim[3] = { RGB (255, 0, 0), RGB (0, 2
            RGB (0, 0, 255) } ;
          static HBRUSH        hBrush[3], hBrushStatic ;
          static HWND          hwndScroll[3], hwndLabel[3], hwndVa
```

```
    static int                  color[3], cyChar ;
    static RECT          rcColor ;
    static TCHAR *szColorLabel[] = {    TEXT ("Red"), TEXT ("G
TEXT ("Blue") } ;
    HINSTANCE                   hInstance ;
    int                         i, cxClient, cyClient ;
    TCHAR                       szBuffer[10] ;

    switch (message)
    {
    case   WM_CREATE :
         hInstance = (HINSTANCE) GetWindowLong (hwnd, GW

               // Create the white-rectangle window against wh
               // scroll bars will be positioned. The child windo

         hwndRect = CreateWindow (TEXT ("static"), NULL,
              WS_CHILD | WS_VISIBLE | SS_WHITERECT,
```

```
                0, 0, 0, 0,
                hwnd, (HMENU) 9, hInstance, NULL) ;

            for (i = 0 ; i < 3 ; i++)
            {
                        // The three scroll bars have IDs 0, 1, and 2
                        // scroll bar ranges from 0 through 255.

            hwndScroll[i] = CreateWindow (TEXT ("scrollbar"), NU
                        WS_CHILD | WS_VISIBLE |
                        WS_TABSTOP | SBS_VERT,
                          0, 0, 0, 0,
          hwnd, (HMENU) i, hInstance, NULL) ;

                    SetScrollRange (hwndScroll[i], SB_CTL, 0, 255, F
                    SetScrollPos   (hwndScroll[i], SB_CTL, 0, FALSE)

 // The three color-name labels have IDs 3, 4, and 5,
// and text strings "Red", "Green", and "Blue".
```

```
        hwndLabel [i] = CreateWindow (TEXT ("static"), zColorLab
                        WS_CHILD | WS_VISIBLE | SS_CENTER,
                        0, 0, 0, 0,
                        hwnd, (HMENU) (i + 3),
                        hInstance, NULL) ;

                    // The three color-value text fields have IDs 6, 7,
                    // and 8, and initial text strings of "0".

hwndValue [i] = CreateWindow (TEXT ("static"), TEXT ("0"),
                        WS_CHILD | WS_VISIBLE | SS_CENTER,
                        0, 0, 0, 0,
                        hwnd, (HMENU) (i + 6),
                        hInstance, NULL) ;

                    OldScroll[i] = (WNDPROC) SetWindowLong (hwn
                        GWL_WNDPROC, (LONG) ScrollProc) ;
```

```
            hBrush[i] = CreateSolidBrush (crPrim[i]) ;
        }

         hBrushStatic = CreateSolidBrush (
                    GetSysColor (COLOR_BTNHIGHLIGHT)) ;

 cyChar = HIWORD (GetDialogBaseUnits ()) ;
return 0 ;

case   WM_SIZE :
        cxClient = LOWORD (lParam) ;
        cyClient = HIWORD (lParam) ;
        SetRect (&rcColor, cxClient / 2, 0, cxClient, cyClient)

        MoveWindow (hwndRect, 0, 0, cxClient / 2, cyClient, T

        for (i = 0 ; i < 3 ; i++)
{
```

```
                MoveWindow (hwndScroll[i],
                        (2 * i + 1) * cxClient / 14, 2 * cyChar,
                         cxClient / 14, cyClient - 4 * cyChar, TRUE) ;

                MoveWindow (hwndLabel[i],
                         (4 * i + 1) * cxClient / 28, cyChar / 2,
                        cxClient / 7, cyChar, TRUE)

                MoveWindow (hwndValue[i],
                         (4 * i + 1) * cxClient / 28,
                         cyClient - 3 * cyChar / 2,
                         cxClient / 7, cyChar, TRUE) ;
        }
             SetFocus (hwnd) ;
             return 0 ;

        case   WM_SETFOCUS :
             SetFocus (hwndScroll[idFocus]) ;
```

```
            return 0 ;

        case   WM_VSCROLL :
            i = GetWindowLong ((HWND) lParam, GWL_ID) ;

            switch (LOWORD (wParam))
            {
            case   SB_PAGEDOWN :
                        color[i] += 15 ;
                   // fall through
            case   SB_LINEDOWN :
                       color[i] = min (255, color[i] + 1) ;
                        break ;

            case   SB_PAGEUP :
                        color[i] -= 15 ;
                   // fall through
            case   SB_LINEUP :
                        color[i] = max (0, color[i] - 1) ;
```

```
                    break ;

            case   SB_TOP :
                    color[i] = 0 ;
                    break ;

            case   SB_BOTTOM :
                    color[i] = 255 ;
                    break ;

            case   SB_THUMBPOSITION :
            case   SB_THUMBTRACK :
                    color[i] = HIWORD (wParam) ;
                    break ;

            default :
                    break ;
            }
```

```
            SetScrollPos  (hwndScroll[i], SB_CTL, color[i], TRUE) ;
            wsprintf (szBuffer, TEXT ("%i"), color[i]) ;
            SetWindowText (hwndValue[i], szBuffer) ;

            DeleteObject ((HBRUSH)
                        SetClassLong (hwnd, GCL_HBRBACKGROUN
            CreateSolidBrush (RGB (color[0], color[1], color[2]))))

            InvalidateRect (hwnd, &rcColor, TRUE) ;
            return 0 ;

        case   WM_CTLCOLORSCROLLBAR :
            i = GetWindowLong ((HWND) lParam, GWL_ID) ;
            return (LRESULT) hBrush[i] ;

        case   WM_CTLCOLORSTATIC :
            i = GetWindowLong ((HWND) lParam, GWL_ID) ;

  if (i >= 3 && i <= 8)                    // static text controls
```

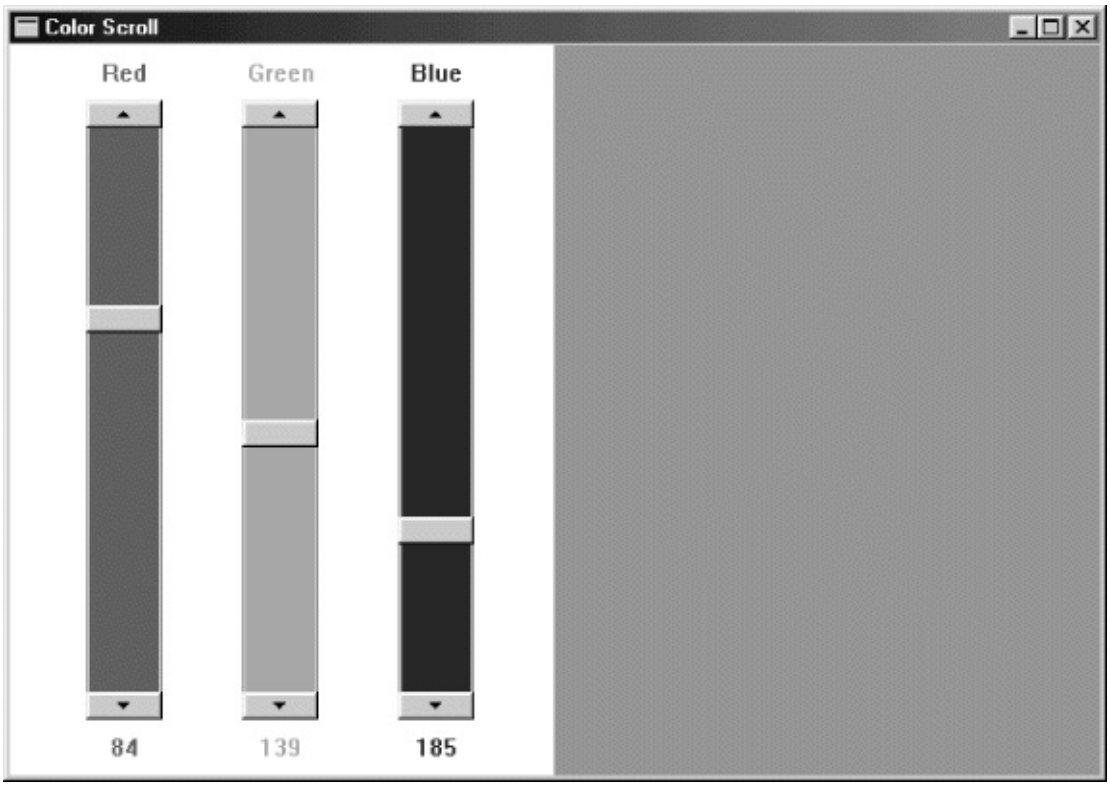
```
        {
                SetTextColor ((HDC) wParam, crPrim[i % 3]) ;
                SetBkColor ((HDC) wParam, GetSysColor (COLOI
                return (LRESULT) hBrushStatic ;
        }
        break ;
        case   WM_SYSCOLORCHANGE :
             DeleteObject (hBrushStatic) ;
              hBrushStatic = CreateSolidBrush (GetSysColor(COLO
              return 0 ;
        case   WM_DESTROY :
              DeleteObject ((HBRUSH)
                        SetClassLong (hwnd, GCL_HBRBACKGROUN
                             GetStockObject (WHITE_BRUSH))) ;

              for (i = 0 ; i < 3 ; i++)
                             DeleteObject (hBrush[i]) ;
```

```
            DeleteObject (hBrushStatic) ;
            PostQuitMessage (0) ;
            return 0 ;
   }
        return DefWindowProc (hwnd, message, wParam, lParam)
}

LRESULT CALLBACK ScrollProc (HWND hwnd, UINT message,
                    WPARAM wParam, LPARAM lParam)
{
        int id = GetWindowLong (hwnd, GWL_ID) ;
        switch (message)
        {
        case WM_KEYDOWN :
            if (wParam == VK_TAB)
                                SetFocus (GetDlgItem (GetParent (hwn
             (id + (GetKeyState (VK_SHIFT) < 0 ? 2 : 1)) % 3)) ;
            break ;
        case   WM_SETFOCUS :
```

```
            idFocus = id ;
             break ;
        }
        return CallWindowProc (OldScroll[id], hwnd, message, wP
}
```

COLORS110361COLORS1WM_CTLCOLORSCROLLBAR 3WM_CTLCOLORSTATIC

COLORS1WindowsCOLORS19-3



COLORS1WM_PAINT

SS_WHITERECTSBS_VERTSS_WHITERECTSS_CENTER
COLORS1WinMainCreateWindow10SS_WHITERECT
SS_CENTERstaticscrollbar

CreateWindowxy0COLORS1WM_SIZEMoveWindow10
COLORS1

WndProcWM_VSCROLLlParamGetWindowWordID

```
i = GetWindowLong ((HWND) lParam, GWL_ID) ;
```

ID012WndProc

WndProcSetScrollPos

```
SetScrollPos (hwndScroll[i], SB_CTL, color[i], TRUE) ;
```

WndProc

```
wsprintf (szBuffer, TEXT ("%i"), color[I]) ;

SetWindowText (hwndValue[i], szBuffer) ;
```

	wParam
Home	SB_TOP
End	SB_BOTTOM
Page Up	SB_PAGEUP
Page Down	SB_PAGEDOWN
	SB_LINEUP
	SB_LINEDOWN

SB_TOPSB_BOTTOMWS_TABSTOPCreateWindow

WndProcWM_SETFOCUSWM_SETFOCUSWndProc

```
SetFocus (hwndScroll[idFocus]) ;
```

idFocus

TabTabCOLORS1Tab

Window Subclassing

WindowsGWL_WNDPROCGetWindowLongSetWindowLong

COLORS1ScrollProcCOLORS1.CScrollProcCOLORS1Windows COLORS1ScrollProccallback

COLORS1SetWindowLong

```
OldScroll[i] = (WNDPROC) SetWindowLong (hwndScroll[i], GWL_
        (LONG) ScrollProc)) ;
```

ScrollProcWindowsCOLORS1ScrollProcTabShift-Tab CallWindowProc

COLORS1

```
wndclass.hbrBackground = CreateSolidBrush (0) ;
```

COLORS1GetWindowLongSetWindowLongGetClassWord SetClassWord

```
DeleteObject ((HBRUSH)
        SetClassLong (hwnd, GCL_HBRBACKGROUND, (LONG)
            CreateSolidBrush (RGB (color[0], color[1], color[2]))))
```

WindowsWindows

```
InvalidateRect (hwnd, &rcColor, TRUE) ;
```

```

```

TRUE

InvalidateRectWindowsWM_PAINTWM_PAINT
InvalidateRect

```
UpdateWindow (hwnd) ;
```

COLORS1WndProcWM_PAINTDefWindowProcWindowsWM_PAINT
BeginPaintEndPaintInvalidateRectBeginPaintWindows
WM_ERASEBKGNDWndProcWindows

WM_DESTROYDeleteObject

```
DeleteObject ((HBRUSH)

        SetClassLong (hwnd, GCL_HBRBACKGROUND,

            (LONG) GetStockObject (WHITE_BRUSH))) ;
```

COLORS1WM_CTLCOLORSCROLLBAR

WndProc

```
static HBRUSH hBrush [3] ;
```

WM_CREATE

```
for (I = 0 ; I < 3 ; I++)
```

```
    hBrush[0] = CreateSolidBrush (crPrim [I]) ;
```

crPrimRGBWM_CTLCOLORSCROLLBAR

```
case        WM_CTLCOLORSCROLLBAR:

          i = GetWindowLong ((HWND) lParam, GWL_ID) ;

          return (LRESULT) hBrush [i] ;
```

WM_DESTROY

```
for (i = 0 ; i < 3 ; i++)

          DeleteObject (hBrush [i])) ;
```

WM_CTLCOLORSTATICSetTextColorSetBkColor
COLOR_BTNHIGHLIGHTCOLOR_BTNHIGHLIGHT
hBrushStaticWM_CREATEWM_DESTROY

WM_CREATECOLOR_BTNHIGHLIGHT
COLOR_BTNHIGHLIGHTCOLOR_BTNHIGHLIGHT

COLORS1hBrushStaticWM_SYSCOLORCHANGE

editCreateWindowxyShift
Ctrl-XCtrl-CCtrl-V

9-4

9-4 POPPAD1

```

```

POPPAD1.C

```
/*------------------------------------------------------------------------
  POPPAD1.C -- Popup Editor using child window edit box
                        (c) Charles Petzold, 1998
------------------------------------------------------------------------*/

#include <windows.h>

#define ID_EDIT     1

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM);

TCHAR szAppName[] = TEXT ("PopPad1") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                PSTR szCmdLine, int iCmdShow)
{
          HWND                hwnd ;
          MSG                 msg ;
          WNDCLASS            wndclass ;

          wndclass.style                          = CS_HREDRAW |
```

```
        wndclass.lpfnWndProc                = WndProc ;
        wndclass.cbClsExtra                 = 0 ;
        wndclass.cbWndExtra                 = 0 ;
        wndclass.hInstance                  = hInstance ;
        wndclass.hIcon                      = LoadIcon (NULL
        wndclass.hCursor                    = LoadCursor (N
        wndclass.hbrBackground              = (HBRUSH) GetS
        wndclass.lpszMenuName               = NULL ;
        wndclass.lpszClassName              = szAppName ;

        if (!RegisterClass (&wndclass))
        {
             MessageBox (  NULL, TEXT ("This program requires W
                                         szAppName, MB_IC
             return 0 ;
        }

hwnd = CreateWindow (szAppName, szAppName,
                WS_OVERLAPPEDWINDOW,
```

```
                    CW_USEDEFAULT, CW_USEDEFAULT,
                    CW_USEDEFAULT, CW_USEDEFAULT,
                    NULL, NULL, hInstance, NULL) ;
  ShowWindow (hwnd, iCmdShow) ;
  UpdateWindow (hwnd) ;

        while (GetMessage (&msg, NULL, 0, 0))
        {
             TranslateMessage (&msg) ;
             DispatchMessage (&msg) ;
        }
        return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
        static HWND hwndEdit ;
        switch (message)
        {
```

```
        case   WM_CREATE :
               hwndEdit = CreateWindow (TEXT ("edit"), NULL,
              WS_CHILD | WS_VISIBLE | WS_HSCROLL | WS_VSCROLL
              WS_BORDER | ES_LEFT | ES_MULTILINE |
              ES_AUTOHSCROLL | ES_AUTOVSCROLL,
              0, 0, 0, 0, hwnd, (HMENU) ID_EDIT,
             ((LPCREATESTRUCT) lParam) -> hInstance, NULL) ;
               return 0 ;

        case   WM_SETFOCUS :
               SetFocus (hwndEdit) ;
               return 0 ;

        case   WM_SIZE :
        MoveWindow (hwndEdit, 0, 0, LOWORD (lParam), HIWORD
               return 0 ;

        case   WM_COMMAND :
               if (LOWORD (wParam) == ID_EDIT)
```

```
            if (HIWORD (wParam) == EN_ERRSPACE ||
                HIWORD (wParam) == EN_MAXTEXT)
           MessageBox (hwnd, TEXT ("Edit control out of space."),
                szAppName, MB_OK | MB_ICONSTOP) ;
               return 0 ;

          case   WM_DESTROY :
               PostQuitMessage (0) ;
               return 0 ;
         }
          return DefWindowProc (hwnd, message, wParam, lParam)
}
```

POPPAD1I/OC10030,000POPPAD1

CreateWindoweditWS_CHILDES_LEFT
ES_RIGHTES_CENTER

ES_MULTILINEES_AUTOHSCROLL
ES_AUTOHSCROLLEnterES_AUTOVSCROLL

WS_HSCROLLWS_VSCROLLWS_BORDER

WindowsES_NOHIDESEL

POPPAD1CreateWindow

```
WS_CHILD | WS_VISIBLE | WS_HSCROLL | WS_VSCROLL |
        WS_BORDER | ES_LEFT | ES_MULTILINE |
  ES_AUTOHSCROLL | ES_AUTOVSCROLL
```

POPPAD1WndProcWM_SIZEMoveWindow

```
MoveWindow (hwndEdit, 0, 0, LOWORD (lParam),
                        HIWORD (lParam), TRUE) ;
```

1.5

WM_COMMANDwParamlParam

LOWORD (wParam) HIWORD (wParam) lParam	ID

EN_SETFOCUS EN_KILLFOCUS

EN_CHANGE
EN_UPDATE
EN_ERRSPACE
EN_MAXTEXT
EN_HSCROLL
EN_VSCROLL

POPPAD1EN_ERRSPACEEN_MAXTEXT

TabShift-TabCOLORS1HEADEnterTab

SetWindowTextGetWindowTextLengthGetWindowTextPOPPAD

SendMessagePOPPADSendMessage

Shift

```
SendMessage (hwndEdit, WM_CUT, 0, 0) ;
SendMessage (hwndEdit, WM_COPY, 0, 0) ;
SendMessage (hwndEdit, WM_CLEAR, 0, 0) ;
```

WM_CUTWM_COPYWM_CLEAR

```
SendMessage (hwndEdit, WM_PASTE, 0, 0) ;
```

```
```

```
SendMessage (hwndEdit, EM_GETSEL, (WPARAM) &iStart,
                                  (LPARAM) &iEnd) ;
```

1

```
SendMessage (hwndEdit, EM_SETSEL, iStart, iEnd) ;
```

```
SendMessage (hwndEdit, EM_REPLACESEL, 0, (LPARAM) szString
```

```
iCount = SendMessage (hwndEdit, EM_GETLINECOUNT, 0, 0) ;
```

```
iOffset = SendMessage (hwndEdit, EM_LINEINDEX, iLine, 0) ;
```

0iLine-1

```
iLength = SendMessage (hwndEdit, EM_LINELENGTH, iLine, 0) ;
```

```
iLength = SendMessage (hwndEdit, EM_GETLINE, iLine, (LPARAM
```

WM_COMMAND

SpacebarPage

SpacebarCtrlShift

Shift

CreateWindowlistboxWS_CHILDWM_COMMAND
LBS_NOTIFYWM_COMMANDLBS_SORT

LBS_MULTIPLESELLBS_NOREDRAW
WM_SETREDRAWWM_SETREDRAW

WS_BORDERWS_VSCROLL

WindowsLBS_STANDARD

```
(LBS_NOTIFY | LBS_SORT | WS_VSCROLL | WS_BORDER)
```

WS_SIZEBOXWS_CAPTION

```
GetSystemMetrics (SM_CXVSCROLL) ;
```

SendMessage00hwndListiIndexSendMessage
lParamnull

SendMessageLB_ERRSPACE-2SendMessage
LB_ERR-1SendMessageLB_OKAY0SendMessage

LBS_SORTLB_ADDSTRING

```
SendMessage (hwndList, LB_ADDSTRING, 0, (LPARAM) szString)
```

LBS_SORTLB_INSERTSTRING

```
SendMessage (hwndList, LB_INSERTSTRING, iIndex, (LPARAM) sz
```

iIndex4szString450-1LBS_SORT
LB_INSERTSTRINGLB_DIR

LB_DELETESTRING

```
SendMessage (hwndList, LB_DELETESTRING, iIndex, 0) ;
```

LB_RESETCONTENT

```
SendMessage (hwndList, LB_RESETCONTENT, 0, 0) ;
```

```
SendMessage (hwndList, WM_SETREDRAW, FALSE, 0) ;
```

```
SendMessage (hwndList, WM_SETREDRAW, TRUE, 0) ;
```

LBS_NOREDRAW

SendMessageLB_ERR-1

```
iCount = SendMessage (hwndList, LB_GETCOUNT, 0, 0) ;
```

```
SendMessage (hwndList, LB_SETCURSEL, iIndex, 0) ;
```

iParam-1

```
iIndex = SendMessage (hwndList, LB_SELECTSTRING, iIndex,
                      (LPARAM) szSearchString) ;
```

SendMessageiIndexiParamiIndexszSearchStringiIndex-1
SendMessageszSearchStringSendMessageLB_ERR

WM_COMMANDLB_GETCURSEL

```
iIndex = SendMessage (hwndList, LB_GETCURSEL, 0, 0) ;
```

iIndexLB_ERR

```
iLength = SendMessage (hwndList, LB_GETTEXTLEN, iIndex, 0) ;
```

```
iLength = SendMessage (    hwndList, LB_GETTEXT, iIndex,
                                                  (LPARAM) szBuffer) ;
```

iLengthNULLszBufferLB_GETTEXTLEN

LB_SETCURSELLB_GETCURSELLB_SELECTSTRINGLB_SETSEL

```
SendMessage (hwndList, LB_SETSEL, wParam, iIndex) ;
```

wParam0wParam0wParam-1

```
iSelect = SendMessage (hwndList, LB_GETSEL, iIndex, 0) ;
```

iIndexiSelect00

```
SetFocus (hwndList) ;
```

Spacebar

WM_COMMANDwParamlParam

]	
LOWORD (wParam)	ID
HIWORD (wParam)	
lParam	

LBN_ERRSPACE	-2
LBN_SELCHANGE	1
LBN_DBLCLK	2
LBN_SELCANCEL	3
LBN_SETFOCUS	4
LBN_KILLFOCUS	5

LBS_NOTIFYLBN_SELCHANGELBN_DBLCLK

LBN_ERRSPACELBN_SELCHANGESpacebar
LBN_DBLCLKLBN_SELCHANGELBN_DBLCLK

LBN_SELCHANGELBN_DBLCLKLBN_SELCHANGE
LBN_DBLCLKLBN_DBLCLK

9-5ENVIRONPATHWINDIR

9-5 ENVIRON

```
ENVIRON.C

/*------------------------------------------------------------------------

  ENVIRON.C -- Environment List Box

                          (c) Charles Petzold, 1998

------------------------------------------------------------------------*/

#include <windows.h>

#define ID_LIST     1

#define ID_TEXT     2

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                         PSTR szCmdLine, int iCmdSh
{
```

```
static TCHAR szAppName[] = TEXT ("Environ") ;
HWND                        hwnd ;
MSG                         msg ;
WNDCLASS                    wndclass ;

wndclass.style                          = CS_HREDRAW |
wndclass.lpfnWndProc                      = WndProc ;
wndclass.cbClsExtra                     = 0 ;
wndclass.cbWndExtra                       = 0 ;
wndclass.hInstance                      = hInstance ;
wndclass.hIcon                          = LoadIcon (NULL,
wndclass.hCursor                        = LoadCursor (N
wndclass.hbrBackground                  = (HBRUSH) (COL
wndclass.lpszMenuName                   = NULL ;
wndclass.lpszClassName                  = szAppName ;

if (!RegisterClass (&wndclass))
{
```

```
        MessageBox (  NULL, TEXT ("This program requires W
                                                szAppName, MB_IC
        return 0 ;
    }

    hwnd = CreateWindow (szAppName, TEXT ("Environment
            WS_OVERLAPPEDWINDOW,
            CW_USEDEFAULT, CW_USEDEFAULT,
            CW_USEDEFAULT, CW_USEDEFAULT,
            NULL, NULL, hInstance, NULL) ;

    ShowWindow (hwnd, iCmdShow) ;
   UpdateWindow (hwnd) ;

    while (GetMessage (&msg, NULL, 0, 0))
    {
            TranslateMessage (&msg) ;
            DispatchMessage (&msg) ;
    }
```

```
        return msg.wParam ;
}

void FillListBox (HWND hwndList)
{
        int     iLength ;
        TCHAR * pVarBlock, * pVarBeg, * pVarEnd, * pVarName ;

        pVarBlock = GetEnvironmentStrings () ;       // Get pointer

        while (*pVarBlock)
        {
        if (*pVarBlock != '=')              // Skip variable names begi
        {
          pVarBeg = pVarBlock ;     // Beginning of variable name
         while (*pVarBlock++ != '=') ; // Scan until '='
         pVarEnd = pVarBlock - 1 ;           // Points to '=' sign
         iLength = pVarEnd - pVarBeg ; // Length of variable name

         // Allocate memory for the variable name and terminatin
```

```
          // zero. Copy the variable name and append a zero.

          pVarName = calloc (iLength + 1, sizeof (TCHAR)) ;
          CopyMemory (pVarName, pVarBeg, iLength * sizeof (TCH
          pVarName[iLength] = '\0' ;

          // Put the variable name in the list box and free memory.
          SendMessage (hwndList, LB_ADDSTRING, 0, (LPARAM) pV
                              free (pVarName) ;
          }
          while (*pVarBlock++ != '\0') ;       // Scan until terminatir
          }
          FreeEnvironmentStrings (pVarBlock) ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
          static HWND   hwndList, hwndText ;
          int                         iIndex, iLength, cxChar, cyChar ;
          TCHAR                    *      pVarName, * pVarValue ;
```

```
switch (message)
{
case   WM_CREATE :
      cxChar = LOWORD (GetDialogBaseUnits ()) ;
      cyChar = HIWORD (GetDialogBaseUnits ()) ;
                        // Create listbox and static text window

      hwndList = CreateWindow (TEXT ("listbox"), NULL,
            WS_CHILD | WS_VISIBLE | LBS_STANDARD,
           cxChar, cyChar * 3,
            cxChar * 16 + GetSystemMetrics (SM_CXVSCRO
            cyChar * 5,
            hwnd, (HMENU) ID_LIST,
           (HINSTANCE) GetWindowLong (hwnd, GWL_HINS
                                          NULL) ;

      hwndText = CreateWindow (TEXT ("static"), NULL,
            WS_CHILD | WS_VISIBLE | SS_LEFT,
           cxChar, cyChar,
```

```
                    GetSystemMetrics (SM_CXSCREEN), cyChar,
                    hwnd, (HMENU) ID_TEXT,
                    (HINSTANCE) GetWindowLong (hwnd, GWL_HINS
                                                        NULL) ;

               FillListBox (hwndList) ;
               return 0 ;

case   WM_SETFOCUS :
               SetFocus (hwndList) ;
               return 0 ;
          case   WM_COMMAND :
               if (LOWORD (wParam) == ID_LIST && HIWORD (wPar
               {
                                        // Get current selection.

            iIndex  = SendMessage (hwndList, LB_GETCURSEL, 0, 0)
             iLength = SendMessage (hwndList, LB_GETTEXTLEN, iIn
           pVarName = calloc (iLength, sizeof (TCHAR)) ;
```

```
          SendMessage (hwndList, LB_GETTEXT, iIndex, (LPARAM)

                                        // Get environment string.

           iLength = GetEnvironmentVariable (pVarName, NULL, 0

           pVarValue = calloc (iLength, sizeof (TCHAR)) ;

           GetEnvironmentVariable (pVarName, pVarValue, iLengt

                                        // Show it in window.


           SetWindowText (hwndText, pVarValue) ;

          free (pVarName) ;

           free (pVarValue) ;

              }

              return 0 ;

        case   WM_DESTROY :

              PostQuitMessage (0) ;

              return 0 ;

        }

        return DefWindowProc (hwnd, message, wParam, lParam)
```

```
}
```

ENVIRONLBS_STANDARDSS_LEFTENVIRON
GetEnvironmentStringsENVIRONFillListBoxLB_ADDSTRING

ENVIRONWndProcWM_COMMANDWndProc
WM_COMMANDwParamID_LISTIDwParam
LBN_SELCHANGELB_GETCURSELLB_GETTEXTENVIRON
CGetEnvironmentVariableSetWindowText

LB_DIR

```
SendMessage (hwndList, LB_DIR, iAttr, (LPARAM) szFileSpec) ;
```

iAttr9-6

9-6

iAttr	
DDL_READWRITE	0x0000
DDL_READONLY	0x0001
DDL_HIDDEN	0x0002

DDL_SYSTEM	0x0004
DDL_DIRECTORY	0x0010
DDL_ARCHIVE	0x0020

iAttr	
DDL_DRIVES	0x4000
DDL_EXCLUSIVE	0x8000

DDL

LB_DIRiAttrDDL_READWRITEDDL_DIRECTORY
DDL_DRIVES | DDL_DIRECTORY

iAttrWindowsDDL_EXCLUSIVE

lParam*.*

LBS_SORT

[..]

[SUBDIR]

[-A-]

Windowshead

UNIXheadWindows9-6HEADEnterHEAD8
KB

9-6 HEAD

```
HEAD.C
/*------------------------------------------------------------------------
  HEAD.C -- Displays beginning (head) of file
                    (c) Charles Petzold, 1998
------------------------------------------------------------------------*/
#include <windows.h>
#define ID_LIST     1
#define ID_TEXT     2

#define MAXREAD     8192
#define DIRATTR     (DDL_READWRITE | DDL_READONLY | DDL_H
```

```
                DDL_DIRECTORY | DDL_ARCHIVE  | DDL_DRIVES)
#define DTFLAGS     (DT_WORDBREAK | DT_EXPANDTABS | DT_N
LRESULT CALLBACK WndProc  (HWND, UINT, WPARAM, LPARAM)
LRESULT CALLBACK ListProc (HWND, UINT, WPARAM, LPARAM) ;

WNDPROC OldList ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                          PSTR szCmdLine, int iCmdSh
{
        static TCHAR        szAppName[] = TEXT ("head") ;
        HWND                              hwnd ;
        MSG                               msg ;
        WNDCLASS                          wndclass ;
   wndclass.style                         = CS_HREDRAW | CS_VREDR
        wndclass.lpfnWndProc                = WndProc ;
        wndclass.cbClsExtra                 = 0 ;
        wndclass.cbWndExtra                 = 0 ;
        wndclass.hInstance                  = hInstance ;
```

```
        wndclass.hIcon                  = LoadIcon (NULL, IDI_APPL
       wndclass.hCursor                 = LoadCursor (NULL, IDC_A
        wndclass.hbrBackground          = (HBRUSH) (COLOR_B
        wndclass.lpszMenuName           = NULL ;
        wndclass.lpszClassName          = szAppName ;

if (!RegisterClass (&wndclass))
        {
             MessageBox (  NULL, TEXT ("This program requires W
                                                szAppName, MB_IC
              return 0 ;
        }

        hwnd = CreateWindow (szAppName, TEXT ("head"),
                  WS_OVERLAPPEDWINDOW | WS_CLIPCHILDREN,
                  CW_USEDEFAULT, CW_USEDEFAULT,
                  CW_USEDEFAULT, CW_USEDEFAULT,
                 NULL, NULL, hInstance, NULL) ;
```

```
            ShowWindow (hwnd, iCmdShow) ;
            UpdateWindow (hwnd) ;

            while (GetMessage (&msg, NULL, 0, 0))
   {
            TranslateMessage (&msg) ;
            DispatchMessage (&msg) ;
   }
        return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
        static BOOL             bValidFile ;
  static BYTE                   buffer[MAXREAD] ;
        static HWND                     hwndList, hwndText ;
        static RECT                   rect ;
        static TCHAR                    szFile[MAX_PATH + 1] ;
        HANDLE                        hFile ;
```

```
     HDC                         hdc ;
     int                        i, cxChar, cyChar ;
     PAINTSTRUCT                     ps ;
     TCHAR                       szBuffer[MAX_PATH + 1] ;
   switch (message)
{
     case   WM_CREATE :
          cxChar = LOWORD (GetDialogBaseUnits ()) ;
          cyChar = HIWORD (GetDialogBaseUnits ()) ;

          rect.left = 20 * cxChar ;
          rect.top  =  3 * cyChar ;

          hwndList = CreateWindow (TEXT ("listbox"), NULL,
                WS_CHILDWINDOW | WS_VISIBLE | LBS_STANDA
                cxChar, cyChar * 3,
               cxChar * 13 + GetSystemMetrics (SM_CXVSCROI
               cyChar * 10,
```

```
                hwnd, (HMENU) ID_LIST,
                (HINSTANCE) GetWindowLong (hwnd, GWL_HINS
                                                NULL) ;

          GetCurrentDirectory (MAX_PATH + 1, szBuffer) ;

          hwndText =    CreateWindow (TEXT ("static"), szBuffe
                WS_CHILDWINDOW | WS_VISIBLE | SS_LEFT,
                cxChar, cyChar, cxChar * MAX_PATH, cyChar,
                hwnd, (HMENU) ID_TEXT,
                (HINSTANCE) GetWindowLong (hwnd, GWL_HINS
                                        NULL) ;

     OldList = (WNDPROC) SetWindowLong    (hwndList, GWL_
          (LPARAM) ListProc) ;

     SendMessage (hwndList, LB_DIR, DIRATTR, (LPARAM) TEX
return 0 ;
```

```
caseWM_SIZE :
        rect.right    = LOWORD (lParam) ;
  rect.bottom                 = HIWORD (lParam) ;
  return 0 ;
case       WM_SETFOCUS :
   SetFocus (hwndList) ;
   return 0 ;


case       WM_COMMAND :
   if (LOWORD (wParam) == ID_LIST && HIWORD (wParam) == l
   {
        if (LB_ERR == (i = SendMessage (hwndList, LB_GETCURS
                  break ;

        SendMessage (hwndList, LB_GETTEXT, i, (LPARAM) szBuf

        if (INVALID_HANDLE_VALUE != (hFile = CreateFile (szBuff
                  GENERIC_READ, FILE_SHARE_READ, NULL,
                       OPEN_EXISTING, 0, NULL)))
```

```
{
            CloseHandle (hFile) ;
            bValidFile = TRUE ;
            lstrcpy (szFile, szBuffer) ;
            GetCurrentDirectory (MAX_PATH + 1, szBuffer)

if (szBuffer [lstrlen (szBuffer) - 1] != '\\')
            lstrcat (szBuffer, TEXT ("\\")) ;
SetWindowText (hwndText, lstrcat (szBuffer, szFile)) ;
}
else
{
      bValidFile = FALSE ;
      szBuffer [lstrlen (szBuffer) - 1] = '\0' ;

            // If setting the directory doesn't work, maybe i
            // a drive change, so try that.

if (!SetCurrentDirectory (szBuffer + 1))
      {
```

```
                szBuffer [3] = ':' ;
                                szBuffer [4] = '\0' ;
                                SetCurrentDirectory (szBuffer + 2
                        }

                        // Get the new directory name and fill the lis

                        GetCurrentDirectory (MAX_PATH + 1, szBuf
                        SetWindowText (hwndText, szBuffer) ;
                        SendMessage (hwndList, LB_RESETCONTEN
                        SendMessage (hwndList, LB_DIR, DIRATTR,
                        (LPARAM) TEXT ("*.*")) ;
                        }
                InvalidateRect (hwnd, NULL, TRUE) ;
        }
        return 0 ;
case   WM_PAINT :
        if (!bValidFile)
                        break ;
```

```
 if (INVALID_HANDLE_VALUE == (hFile = CreateFile (sz
GENERIC_READ, FILE_SHARE_READ, NULL, OPEN_EXIS
 {
              bValidFile = FALSE ;
              break ;
 }

 ReadFile (hFile, buffer, MAXREAD, &i, NULL) ;
 CloseHandle (hFile) ;

             // i now equals the number of bytes in buffe
              // Commence getting a device context for c

 hdc = BeginPaint (hwnd, &ps) ;
 SelectObject (hdc, GetStockObject (SYSTEM_FIXED_F
SetTextColor (hdc, GetSysColor (COLOR_BTNTEXT)) ;
 SetBkColor   (hdc, GetSysColor (COLOR_BTNFACE)) ;

             // Assume the file is ASCII

 DrawTextA (hdc, buffer, i, &rect, DTFLAGS) ;
```

```
            EndPaint (hwnd, &ps) ;
            return 0 ;

        case   WM_DESTROY :
            PostQuitMessage (0) ;
            return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam)
}

LRESULT CALLBACK ListProc (HWND hwnd, UINT message,
            WPARAM wParam, LPARAM lParam)
{
        if (message == WM_KEYDOWN && wParam == VK_RETU
            SendMessage (GetParent (hwnd), WM_COMMAND,
            MAKELONG (1, LBN_DBLCLK), (LPARAM) hwnd) ;
        return CallWindowProc (OldList, hwnd, message, wParam,
}
```

ENVIRONHEADHEAD

HEADListProcwParamVK_RETURNWM_KEYDOWN
LBN_DBLCLKWM_COMMANDWndProcWM_COMMAND
WindowsCreateFileCreateFileHEADSetCurrentDirectory
SetCurrentDirectorySetCurrentDirectory
LB_RESETCONTENTLB_DIR

WndProcWM_PAINTWindowsCreateFileWindowsReadFile
CloseHandle

UnicodeReadFileASCIIUnicodeUnicodeASCII
ReadFileDrawTextAUNICODEASCIIDrawTextWUnicode
Unicode

ASCIIUnicodeDrawTextADrawTextWHEADDrawTextA

WindowsWindowsWindows
WindowsWindows

Windows.EXEWindowsLoadIcon
LoadCursorWindows.EXE

-
-
-
-
-
-
-
-

.EXE.EXE.EXE

Visual C++ Developer StudioImage

.ICODeveloper Studio.RCRESOURCE.H

ICONDEMODeveloper
Project NameICONDEMOOK Developer StudioICONDEMO.DSW
ICONDEMO.DSPICONDEMO.MAK **ToolsOpen Open Bu**
Export makefile when saving project fileC **FileNewFiles**
C++Source FileFile NameICONDEMO.C**OK**Developer Studio
ICONDEMO.C10-1 **Insert File As Text**

10-1 ICONDEMO

```
ICONDEMO.C

/*------------------------------------------------------------------------
   ICONDEMO.C --        Icon Demonstration Program

                              (c) Charles Petzold, 1998

------------------------------------------------------------------------*/

#include <windows.h>

#include "resource.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns

                                    PSTR szCmdLine, int iCmdSh

{

        TCHAR                   szAppName[]         = TEXT ("IconD
```

```
     HWND                  hwnd ;
     MSG                   msg ;
     WNDCLASS              wndclass ;

     wndclass.style                    = CS_HREDRAW | CS_V
     wndclass.lpfnWndProc                = WndProc ;
     wndclass.cbClsExtra                 = 0 ;
    wndclass.cbWndExtra                  = 0 ;
     wndclass.hInstance                  = hInstance ;
     wndclass.hIcon                    = LoadIcon (hInstance
     wndclass.hCursor                   = LoadCursor (NULL,
     wndclass.hbrBackground            = GetStockObject (WH
     wndclass.lpszMenuName              = NULL ;
     wndclass.lpszClassName            = szAppName ;
     if (!RegisterClass (&wndclass))
     {
          MessageBox (  NULL, TEXT ("This program requires W
                                          szAppName, MB_IC
           return 0 ;
```

```
   }

 hwnd = CreateWindow (szAppName, TEXT ("Icon Demo"),
                WS_OVERLAPPEDWINDOW,
                CW_USEDEFAULT, CW_USEDEFAULT,
                CW_USEDEFAULT, CW_USEDEFAULT,
               NULL, NULL, hInstance, NULL) ;

         ShowWindow (hwnd, iCmdShow) ;
         UpdateWindow (hwnd) ;

  while (GetMessage (&msg, NULL, 0, 0))
         {
              TranslateMessage (&msg) ;
              DispatchMessage (&msg) ;
  }
         return msg.wParam ;
}
```

```
LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
        static HICON hIcon ;
        static int   cxIcon, cyIcon, cxClient, cyClient ;
        HDC         hdc ;
        HINSTANCE    hInstance ;
        PAINTSTRUCT  ps ;
        int         x, y ;

        switch (message)
        {
        case   WM_CREATE :
              hInstance     = ((LPCREATESTRUCT) lParam)->hInstar
              hIcon         = LoadIcon (hInstance, MAKEINTRESOURC
              cxIcon        = GetSystemMetrics (SM_CXICON) ;
             cyIcon        = GetSystemMetrics (SM_CYICON) ;
              return 0 ;

   case   WM_SIZE :
```

```
            cxClient            = LOWORD (lParam) ;
            cyClient            = HIWORD (lParam) ;
            return 0 ;

case   WM_PAINT :
            hdc = BeginPaint (hwnd, &ps) ;

            for (y = 0 ; y < cyClient ; y += cyIcon)
                        for (x = 0 ; x < cxClient ; x += cxIcon)
                                    DrawIcon (hdc, x, y, hIcon) ;

                        EndPaint (hwnd, &ps) ;
                        return 0 ;

      case   WM_DESTROY :
            PostQuitMessage (0) ;
            return 0 ;
      }
      return DefWindowProc (hwnd, message, wParam, lParam)
```

```
}
```

RESOURCE.HRESOURCE.HDeveloper

FileNewFilesResource
ICONDEMOOKDeveloper StudioICONDEMO.RC
RESOURCE.HCDeveloper
StudioDeveloper

RESOURCE.HICONDEMOIDI_ICON

```
wndclass.hIcon  = LoadIcon (hInstance, MAKEINTRESOURCE (IDI_
```

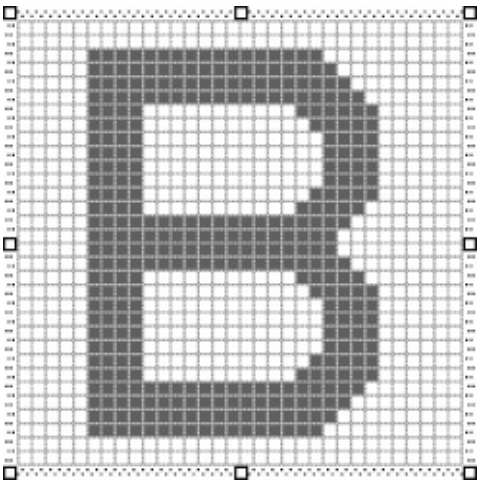
```
wndclass.hIcon = LoadIcon (NULL, IDI_APPLICATION) ;
```

Developer StudioFile ViewICONDEMO.CICONI
CONDEMO.CICONDEMO.RC
New

32×32

Icon PropertiesIDDeveloper
ICONDEMOIDIIDICONDEMO.ICO

BBIG10-1



10-1 Developer Studio32×32
ICONDEMO

Developer StudioICONDEMO.RCIDI_ICON ICONDEMO.ICORESOURCE.HIDI_ICON

Developer StudioRC.EXE.RES.OBJ.LIBLINK.EXE

ICONDEMO

ICONDEMO

```
hIcon = LoadIcon (hInstance, MAKEINTRESOURCE (IDI_ICON)) ;
```

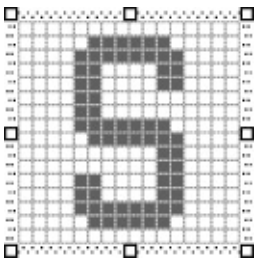
```
cxIcon = GetSystemMetrics (SM_CXICON) ;

cyIcon = GetSystemMetrics (SM_CYICON) ;
```

```
DrawIcon (hdc, x, y, hIcon) ;
```

xy

SM_CXICONSM_CYICONGetSystemMetrics32×32Developer
ICONDEMOSM_CXSMSIZESM_CYSMSIZE
GetSystemMetricsSMsystem metricsSMsmall
16×16

Windows32×3216×1616×16Developer
DeviceNew Icon Ima
S



10-2 Developer Studio16×16
ICONDEMO

ICONDEMO.ICOIDI_ICONWindowsDrawIcon
Windows

ICONDEMO.RCRESOURCE.HDeveloper Studio
ICONDEMO.RCRESOURCE.H10-2

```
ICONDEMO.RC

//Microsoft Developer Studio generated resource script.

#include "resource.h"

#include "afxres.h"

/////////////////////////////////////////////////////////////////////////////

// Icon

IDI_ICON                        ICON    DISCARDABLE     "icondemo.ico
```

```
RESOURCE.H

// Microsoft Developer Studio generated include file.

// Used by IconDemo.rc

#define IDI_ICON       101
```

10-2　ICONDEMO.RCRESOURCE.H

10-2ICONDEMO.RCRESOURCE.H80WindowsAFXRES.HMFC Developer StudioAFXRES.H

ICONDEMO.RC

```
IDI_ICON ICON DISCARDABLE "icondemo.ico"
```

ICONIDI_ICON101Developer

WindowsDISCARDABLEDeveloper

ICONDEMO.RESICONDEMO.EXERT_ICONIDI_ICON101
LoadIcon

```
hIcon = LoadIcon (hInstance, MAKEINTRESOURCE (IDI_ICON)) ;
```

ICONDEMOLoadIconHICON

LoadIconhInstance.EXELoadIcon
MAKEINTRESOURCE

```
#define MAKEINTRESOURCE(i)  (LPTSTR) ((DWORD) ((WORD) (i)))
```

LoadIcon016

```
LoadIcon (NULL, IDI_APPLICATION) ;
```

hInstanceNULLWindowsIDI_APPLICATIONWINUSER.H
MAKEINTRESOURCE

```
#define IDI_APPLICATION MAKEINTRESOURCE(32512)
```

LoadIcon
IDONDEMO.RC IconDemo ResourceIconIDI_ICON
PropertiesID

MYPROGIcon

```
MYPROG ICON DISCARDABLE myprog.ico
```

RESOURCE.H#defineMYPROGMYPROG

CLoadIcon

```
static TCHAR szAppName [] = TEXT ("MyProg") ;
```

```
hIcon = LoadIcon (hInstance, szAppName) ;
```

MAKEINTRESOURCE

Icon Properti

```
125 ICON DISCARDABLE myprog.ico
```

```
hIcon = LoadIcon (hInstance, MAKEINTRESOURCE (125)) ;
```

```
hIcon = LoadIcon (hInstance, TEXT ("#125")) ;
```

Windows#ASCII

WindowsWindowsWNDCLASSRegisterClassWindows

RegisterClassRegisterClassExWNDCLASSEXWNDCLASSEXcbSize

hIconSmcbSizeWNDCLASSEXhIconSmWNDCLASSEX

WindowsRegisterClassExRegisterClasshIconSm
RegisterClassExWNDCLASSRegisterClass

SetClassLongIDI_ALTICON

```
SetClassLong (hwnd, GCL_HICON,

    LoadIcon (hInstance, MAKEINTRESOURCE (IDI_ALTICON))) ;
```

DrawIconGetClassLong

```
DrawIcon (hdc, x, y, GetClassLong (hwnd, GCL_HICON)) ;
```

WindowsLoadIconLoadImageLoadIcon/Platform
Interface Services/Resources/IconsLoadImage/Platform SDK/User
Interface Services/Resources/ResourcesLoadImageLoadIcon
ICONDEMOLoadIconLoadIconDestroyIconCreateIcon
CreateIconIndirectCreateIconFromResource

Windows32×32Developer
Cursor

```
wndclass.hCursor = LoadCursor (hInstance, MAKEINTRESOURCE
```

```
wndclass.hCursor = LoadCursor (hInstance, szCursor) ;
```

IDC_CURSORszCursor

hCursorhCursor

```
SetClassLong (hwndChild, GCL_HCURSOR,
    LoadCursor (hInstance, TEXT ("childcursor")) ;
```

SetCursor

```
SetCursor (hCursor) ;
```

WM_MOUSEMOVESetCursorWindowsSetCursor

InsertResource String Table

```
STRINGTABLE DISCARDABLE
BEGIN
    IDS_STRING1, "character string 1"
    IDS_STRING2, "character string 2"

END
```

```

```

WindowsDeveloper

ID4097\t\nlinefeedDrawTextMessageBox

LoadString

```
LoadString (hInstance, id, szBuffer, iMaxLength) ;
```

idIDszBufferiMaxLengthszBuffer

IDWindowsIDS_

Unicode.RES.EXELoadStringWUnicodeLoadStringAWindows 98Unicode

RESOURCE.H

```
#define IDS_FILENOTFOUND          1

#define IDS_FILETOOBIG            2

#define IDS_FILEREADONLY          3
```

```
STRINGTABLE

BEGIN

    IDS_FILENOTFOUND,           "File %s not found."

    IDS_FILETOOBIG,             "File %s too large to edit."

    IDS_FILEREADONLY,            "File %s is read-only."
```

```
END
```

CszAppName

```
OkMessage (HWND hwnd, int iErrorNumber, TCHAR *szFileName
{
   TCHAR szFormat [40] ;
   TCHAR szBuffer [60] ;
   LoadString (hInst, iErrorNumber, szFormat, 40) ;
   wsprintf (szBuffer, szFormat, szFilename) ;

   return MessageBox (   hwnd, szBuffer, szAppName,
                                          MB_OK | MB_ICONEXCLAMA
}
```

file not found

```
OkMessage (hwnd, IDS_FILENOTFOUND, szFileName) ;
```

Windows.EXEWindowsWindows

BINDATA.BINMYPROGMYPROG.RCDeveloper

InsertResource CustomBINTYPEDeveloper
IDR_BINTYPE1IDR_BINTYPE1
BINDATA.BIN

```
IDR_BINTYPE1 BINTYPE BINDATA.BIN
```

BINTYPETICONDEMOICON

BINDATA.BINMYPROG.EXE

WM_CREATE

```
hResource = LoadResource (hInstance,

                FindResource (hInstance, TEXT ("BINTYPE"),

                                                MAKEINTRESOURCE (ID
```

hResourceHGLOBALLoadResourceLoadResourceFindResource
LoadIconLoadCursorLoadIconLoadCursorLoadResource
FindResource

LockResource

```
pData = LockResource (hResource) ;
```

LockResource

```
FreeResource (hResource) ;
```

FreeResource.

10-3POEPOEMEdgar
\

```
10-3  POEPOEM

POEPOEM.C

/*------------------------------------------------------------------------
  POEPOEM.C -- Demonstrates Custom Resource
                              (c) Charles Petzold, 1998
------------------------------------------------------------------------*/

#include <windows.h>
#include "resource.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
HINSTANCE hInst ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                PSTR szCmdLine, int iCmdShow)
{
   TCHAR              szAppName [16], szCaption [64], szErrMsg [
   HWND               hwnd ;
   MSG                msg ;
   WNDCLASS           wndclass ;
```

```
LoadString (  hInstance, IDS_APPNAME, szAppName,
                        sizeof (szAppName) / sizeof (TCHAR)) ;

LoadString (  hInstance, IDS_CAPTION, szCaption,
                        sizeof (szCaption) / sizeof (TCHAR)) ;

wndclass.style                      = CS_HREDRAW | CS_VREDI
wndclass.lpfnWndProc                  = WndProc ;
wndclass.cbClsExtra                   = 0 ;
wndclass.cbWndExtra                   = 0 ;
     wndclass.hInstance               = hInstance ;
     wndclass.hIcon                   = LoadIcon (hInstance, szAp
     wndclass.hCursor                 = LoadCursor (NULL, IDC_A
     wndclass.hbrBackground           = (HBRUSH) GetStockO
     wndclass.lpszMenuName             = NULL ;
     wndclass.lpszClassName           = szAppName ;

     if (!RegisterClass (&wndclass))
```

```
{
LoadStringA (hInstance, IDS_APPNAME, (char *) szAppNar
                                        sizeof (szAppName)) ;
LoadStringA (hInstance, IDS_ERRMSG, (char *) szErrMsg,
                                       sizeof (szErrMsg)) ;
MessageBoxA (NULL, (char *) szErrMsg,
                                        (char *) szAppName, ME
return 0 ;
}

hwnd = CreateWindow (szAppName, szCaption,
        WS_OVERLAPPEDWINDOW | WS_CLIPCHILDREN,
        CW_USEDEFAULT, CW_USEDEFAULT,
        CW_USEDEFAULT, CW_USEDEFAULT,
        NULL, NULL, hInstance, NULL) ;

ShowWindow (hwnd, iCmdShow) ;
UpdateWindow (hwnd) ;
```

```
        while (GetMessage (&msg, NULL, 0, 0))
   {
             TranslateMessage (&msg) ;
             DispatchMessage (&msg) ;
        }
        return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
        static char               * pText ;
        static HGLOBAL       hResource ;
        static HWND               hScroll ;
        static int                  iPosition, cxChar, cyChar, cyClie
        HDC                         hdc ;
        PAINTSTRUCT                    ps ;
        RECT                        rect ;
        TEXTMETRIC                     tm ;
```

```
switch (message)
{
case WM_CREATE :
      hdc                 = GetDC (hwnd) ;
      GetTextMetrics (hdc, &tm) ;
      cxChar        = tm.tmAveCharWidth ;
      cyChar        = tm.tmHeight + tm.tmExternalLeading
      ReleaseDC (hwnd, hdc) ;

      xScroll             = GetSystemMetrics (SM_CXVSCROL
      hScroll             = CreateWindow (TEXT ("scrollbar"),
                  WS_CHILD | WS_VISIBLE | SBS_VERT,
                   0, 0, 0, 0,
   hwnd, (HMENU) 1, hInst, NULL) ;

      hResource =   LoadResource (hInst,
   FindResource (hInst,      TEXT ("AnnabelLee"),
                        TEXT ("TEXT"))) ;
```

```
        pText = (char *) LockResource (hResource) ;
        iNumLines = 0 ;

        while (*pText != '\\' && *pText != '\0')
        {
              if (*pText == '\n')
                        iNumLines ++ ;
               pText = AnsiNext (pText) ;
        }
        *pText = '\0' ;

        SetScrollRange (hScroll, SB_CTL, 0, iNumLines, FALSE
        SetScrollPos   (hScroll, SB_CTL, 0, FALSE) ;
        return 0 ;

   case   WM_SIZE :
        MoveWindow (hScroll, LOWORD (lParam) - xScroll, 0,
      xScroll, cyClient = HIWORD (lParam), TRUE) ;
```

```
        SetFocus (hwnd) ;
        return 0 ;

    case   WM_SETFOCUS :
        SetFocus (hScroll) ;
        return 0 ;

    case   WM_VSCROLL :
        switch (wParam)
        {
        case   SB_TOP :
                        iPosition = 0 ;
                        break ;
        case   SB_BOTTOM :
                        iPosition = iNumLines ;
                        break ;
        case   SB_LINEUP :
                        iPosition -= 1 ;
```

```
                    break ;
          case   SB_LINEDOWN :
                    iPosition += 1 ;
                    break ;
          case   SB_PAGEUP :
                    iPosition -= cyClient / cyChar ;
                    break ;
          case   SB_PAGEDOWN :
                    iPosition += cyClient / cyChar ;
                    break ;
          case   SB_THUMBPOSITION :
                    iPosition = LOWORD (lParam) ;
                    break ;
          }
          iPosition = max (0, min (iPosition, iNumLines)) ;

          if (iPosition != GetScrollPos (hScroll, SB_CTL))
          {
                    SetScrollPos (hScroll, SB_CTL, iPosition
```

```
                        InvalidateRect (hwnd, NULL, TRUE) ;
        }
        return 0 ;

    case   WM_PAINT :
        hdc = BeginPaint (hwnd, &ps) ;

        pText = (char *) LockResource (hResource) ;

        GetClientRect (hwnd, &rect) ;
        rect.left += cxChar ;
        rect.top  += cyChar * (1 - iPosition) ;
        DrawTextA (hdc, pText, -1, &rect, DT_EXTERNALLEAD

        EndPaint (hwnd, &ps) ;
        return 0 ;

    case   WM_DESTROY :
        FreeResource (hResource) ;
```

```
                PostQuitMessage (0) ;
                return 0 ;
  }
        return DefWindowProc (hwnd, message, wParam, lParam)
}
```

```
POEPOEM.RC
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
//////////////////////////////////////////////////////////////////////////////
// TEXT
ANNABELLEE                              TEXT    DISCARDABLE     "poep

//////////////////////////////////////////////////////////////////////////////
// Icon
POEPOEM                                 ICON    DISCARDABLE     "poepoe

//////////////////////////////////////////////////////////////////////////////
// String Table
```

```
STRINGTABLE DISCARDABLE
BEGIN
   IDS_APPNAME         "PoePoem"
   IDS_CAPTION         """Annabel Lee"" by Edgar Allan Poe"
   IDS_ERRMSG             "This program requires Windows NT!"
END
```

RESOURCE.H

```
// Microsoft Developer Studio generated include file.
// Used by PoePoem.rc
#define IDS_APPNAME   1
#define IDS_CAPTION   2
#define IDS_ERRMSG    3
```

POEPOEM.TXT

```
It was many and many a year ago,
  In a kingdom by the sea,
That a maiden there lived whom you may know
By the name of Annabel Lee;
```

And this maiden she lived with no other thought

  Than to love and be loved by me.

I was a child and she was a child

  In this kingdom by the sea,

But we loved with a love that was more than love --

  I and my Annabel Lee --

With a love that the winged seraphs of Heaven

  Coveted her and me.

And this was the reason that, long ago,

  In this kingdom by the sea,

A wind blew out of a cloud, chilling

  My beautiful Annabel Lee;

So that her highborn kinsmen came

  And bore her away from me,

To shut her up in a sepulchre

In this kingdom by the sea.

The angels, not half so happy in Heaven,

  Went envying her and me --

Yes! that was the reason (as all men know,

In this kingdom by the sea)

That the wind came out of the cloud by night,

Chilling and killing my Annabel Lee.

But our love it was stronger by far than the love Of those who w

And neither the angels in Heaven above

Nor the demons down under the sea

Can ever dissever my soul from the soul

Of the beautiful Annabel Lee:

For the moon never beams, without bringing me dreams

Of the beautiful Annabel Lee;

And the stars never rise, but I feel the bright eyes

Of the beautiful Annabel Lee:

And so, all the night-tide, I lie down by the side

Of my darling -- my darling -- my life and my bride,

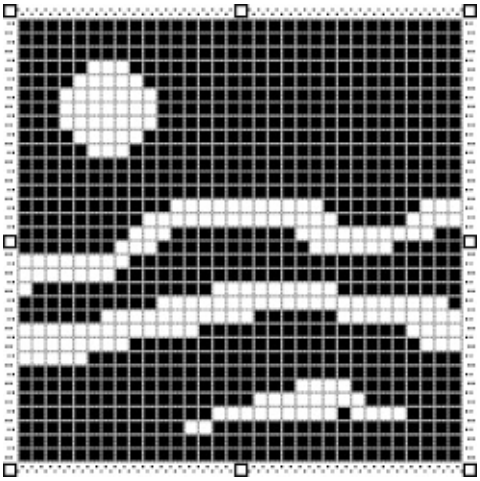
In her sepulchre there by the sea --

In her tomb by the sounding sea.

[May, 1849]

\

POEPOEM.ICO



POEPOEM.RCTEXTAnnabelLee

```
ANNABELLEE  TEXT  POEPOEM.TXT
```

WndProcWM_CREATEFindResourceLoadResourceLockResource
\0WM_PAINTDrawText

POEPOEMWM_KEYDOWN

POEPOEMIDRESOURCE.HIDS_APPNAME
IDS_CAPTIONPOEPOEMLoadString

```
LoadString (hInstance, IDS_APPNAME, szAppName,    sizeof (szA
                                   sizeof (TCHAR)) ;
LoadString (hInstance, IDS_CAPTION, szCaption,    sizeof (szCapt
                                   sizeof (TCHAR)) ;
```

RegisterClassWindows 98UnicodePOEPOEMLoadSt
LoadStringWLoadStringAUnicodeANSILoadStringWLoadStringW
Windows 98Windows 98RegisterClassWMessageBoxWWi
98LoadStringWLoadStringAIDS_APPNAMEIDS_ERRMSG
MessageBoxA

```
if (!RegisterClass (&wndclass))
{
        LoadStringA (hInstance, IDS_APPNAME, (char *) szAppNar
                sizeof (szAppName)) ;
        LoadStringA (hInstance, IDS_ERRMSG, (char *) szErrMsg,
                sizeof (szErrMsg)) ;
        MessageBoxA (NULL, (char *) szErrMsg,
                (char *) szAppName, MB_ICONERROR) ;
        return 0 ;
}
```

TCHARchar

POEPOEMAnnabel

Monty Python40

Windows

WindowsWindowsDeveloper
WM_COMMAND

Windows

Windows
WM_COMMAND

WM_COMMANDWindowsIDWindows

Developer Studio **Insert ResourceMenu** **Properties** **Pop-up**ID **Pop-up**IDWM_COMMAND POPUPMENUITEM

&WindowsAltWindows&WindowsAlt

Menu Items PropertiesGrayedWM_COMMAND **Inactive** WM_COMMAND **Checked** **Separato**

\t\t\a

IDWindowsIDIDMID

Windows

```
wndclass.lpszMenuName = szAppName ;
```

WindowsLoadMenuLoadIconLoadCursorLoadMenu

```
hMenu = LoadMenu (hInstance, TEXT ("MyMenu")) ;
```

LoadMenu

```
hMenu = LoadMenu (hInstance, MAKEINTRESOURCE (ID_MENU))
```

CreateWindow

```
hwnd = CreateWindow (      TEXT ("MyClass"), TEXT ("Window Ca
  WS_OVERLAPPEDWINDOW,
  CW_USEDEFAULT, CW_USEDEFAULT,
  CW_USEDEFAULT, CW_USEDEFAULT,
  NULL, hMenu, hInstance, NULL) ;
```

CreateWindowCreateWindowNULL

NULLCreateWindowNULL

```
SetMenu (hwnd, hMenu) ;
```

NOPOPUPS

DestroyMenu

Windows

wParam

lParam 0

wParamWindowsWM_INITMENU

WM_MENUSELECTWM_MENUSELECT
WM_MENUSELECT

LOWORD (wParam)ID

HIWORD (wParam)

lParam:

WM_MENUSELECTwParamwParamMF_GRAYED
MF_DISABLEDMF_CHECKEDMF_BITMAPMF_POPUP
MF_HELPMF_SYSMENUMF_MOUSESELECTWM_MENUSELECT
DefWindowProc

WindowsWM_INITMENUPOPUP

wParam

LOWORD (lParam)

HIWORD (lParam) 10

WM_COMMANDWM_COMMANDIDlParamlParam010-
1

LOWORD (wParam):	ID	ID
HIWORD (wParam):	0	
lParam:	0	

WM_SYSCOMMANDWM_COMMANDWM_SYSCOMMAND

wParam: ID

lParam: 0

WM_SYSCOMMANDLOWORDlParamHIWORDlParamxy

WM_SYSCOMMANDID0xFFF0ANDSC_SIZE
SC_MOVESC_MINIMIZESC_MAXIMIZESC_NEXTWINDOW
SC_PREVWINDOWSC_CLOSESC_VSCROLLSC_HSCROLL
SC_ARRANGESC_RESTORESC_TASKLISTwParam
SC_MOUSEMENUSC_KEYMENU

wParamIDID0xF000WM_SYSCOMMANDDefWindowProc

WM_MENUCHARWindowsAltWM_MENUCHAR

LOWORD (wParam): ASCIIUnicode

HIWORD (wParam):

lParam:

- 0
- MF_POPUP
- MF_SYSMENU

WindowsDefWindowProcWindows0Windows
WM_MENUCHAR

10-4MENUDEMOFileEditBackgroundTimerHelp
MENUDEMOWM_COMMANDwParam

10-4　　MENUDEMO

```
MENUDEMO.C

/*------------------------------------------------------------------

  MENUDEMO.C --   Menu Demonstration

                                          (c) Charles Petzold, 1998

------------------------------------------------------------------*/

#include <windows.h>

#include "resource.h"


#define ID_TIMER 1
```

```
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

TCHAR szAppName[] = TEXT ("MenuDemo") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                        PSTR szCmdLine, int iCmdSh
{
   HWND     hwnd ;
         MSG                 msg ;
         WNDCLASS            wndclass ;

         wndclass.style                  = CS_HREDRAW | CS_V
         wndclass.lpfnWndProc              = WndProc ;
         wndclass.cbClsExtra               = 0 ;
         wndclass.cbWndExtra               = 0 ;
         wndclass.hInstance                = hInstance ;
         wndclass.hIcon                  = LoadIcon (NULL, IDI_
         wndclass.hCursor                = LoadCursor (NULL,
         wndclass.hbrBackground          = (HBRUSH) GetStockO
```

```
        wndclass.lpszMenuName        = szAppName ;
        wndclass.lpszClassName             = szAppName ;
if (!RegisterClass (&wndclass))
        {
        MessageBox (  NULL, TEXT ("This program requires Windo
                                        szAppName, MB_ICONERRO
        return 0 ;
}

        hwnd = CreateWindow ( szAppName, TEXT ("Menu Demo
                 WS_OVERLAPPEDWINDOW,
                CW_USEDEFAULT, CW_USEDEFAULT,
                 CW_USEDEFAULT, CW_USEDEFAULT,
                 NULL, NULL, hInstance, NULL) ;

        ShowWindow (hwnd, iCmdShow) ;
        UpdateWindow (hwnd) ;
```

```
        while (GetMessage (&msg, NULL, 0, 0))
        {
             TranslateMessage (&msg) ;
             DispatchMessage (&msg) ;
        }
        return msg.wParam ;
}

LRESULT CALLBACK WndProc ( HWND hwnd, UINT message, WPA
{
        static int idColor [5] = {    WHITE_BRUSH,  LTGRAY_BRUSH
                             DKGRAY_BRUSH, BLACK_BRUSH } ;
        static int iSelection = IDM_BKGND_WHITE ;
        HMENU                        hMenu ;

        switch (message)
        {
        case   WM_COMMAND:
             hMenu = GetMenu (hwnd) ;
```

```
switch (LOWORD (wParam))
      {
case   IDM_FILE_NEW:
case   IDM_FILE_OPEN:
case   IDM_FILE_SAVE:
case   IDM_FILE_SAVE_AS:
                  MessageBeep (0) ;
                  return 0 ;

case   IDM_APP_EXIT:
                 SendMessage (hwnd, WM_CLOSE, 0, 0)
                  return 0 ;


case IDM_EDIT_UNDO:
 case IDM_EDIT_CUT:
 case IDM_EDIT_COPY:
 case IDM_EDIT_PASTE:
 case IDM_EDIT_CLEAR:
```

```
                    MessageBeep (0) ;
                    return 0 ;

          case   IDM_BKGND_WHITE:          // Note: Logic bel
          case   IDM_BKGND_LTGRAY:         // assumes that ID
          case   IDM_BKGND_GRAY:           // through IDM_BL
          case   IDM_BKGND_DKGRAY:          // consecutive nu
        case   IDM_BKGND_BLACK:            // the order shown

               CheckMenuItem (hMenu, iSelection, MF_UNCHE
               iSelection = LOWORD (wParam) ;
              CheckMenuItem (hMenu, iSelection, MF_CHECKE

               SetClassLong (hwnd, GCL_HBRBACKGROUND, (L
                                   GetStockObject
         (idColor [LOWORD (wParam) - IDM_BKGND_WHITE]))

              InvalidateRect (hwnd, NULL, TRUE) ;
               return 0 ;
```

```
                case   IDM_TIMER_START:
                                if (SetTimer (hwnd, ID_TIMER, 1000, N
                                {
                 EnableMenuItem (hMenu, IDM_TIMER_START, MF_GR
                 EnableMenuItem (hMenu, IDM_TIMER_STOP,  MF_EN
                                }
                           return 0 ;

                case   IDM_TIMER_STOP:
                                KillTimer (hwnd, ID_TIMER) ;
                                EnableMenuItem (hMenu, IDM_TIMER_
                               EnableMenuItem (hMenu, IDM_TIMER_S
                                return 0 ;
                case   IDM_APP_HELP:
                                MessageBox (hwnd, TEXT ("Help not y
implemented!"),
                      szAppName, MB_ICONEXCLAMATION | MB_OK) ;
```

```
                    return 0 ;

          case   IDM_APP_ABOUT:
                    MessageBox (hwnd,TEXT ("Menu Demo
Program\n")
                TEXT ("(c) Charles Petzold, 1998"),
               szAppName, MB_ICONINFORMATION | MB_OK) ;
                    return 0 ;
          }
          break ;

          case   WM_TIMER:
                    MessageBeep (0) ;
                    return 0 ;

          case   WM_DESTROY:
                    PostQuitMessage (0) ;
                    return 0 ;
   }
```

```
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

MENUDEMO.RC

```
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
//////////////////////////////////////////////////////////////////////////////
// Menu
MENUDEMO MENU DISCARDABLE
BEGIN
  POPUP "&File"
  BEGIN
              MENUITEM "&New",                        IDM_FILE_NEW
             MENUITEM "&Open",                       IDM_FILE_OPEN
              MENUITEM "&Save",                       IDM_FILE_SAVE
              MENUITEM "Save &As...",            IDM_FILE_SAVE_AS
              MENUITEM SEPARATOR
              MENUITEM "E&xit",                       IDM_APP_EXIT
```

```
END
POPUP "&Edit"
BEGIN
        MENUITEM "&Undo",                   IDM_EDIT_UND
        MENUITEM SEPARATOR
        MENUITEM "C&ut",                        IDM_EDIT_
        MENUITEM "&Copy",                   IDM_EDIT_COP
        MENUITEM "&Paste",                  IDM_EDIT_PAST
        MENUITEM "De&lete",             IDM_EDIT_CLEAR
END
POPUP "&Background"
BEGIN
        MENUITEM "&White",      IDM_BKGND_WHITE, CHECK
        MENUITEM "&Light Gray", IDM_BKGND_LTGRAY
        MENUITEM "&Gray",     IDM_BKGND_GRAY
        MENUITEM "&Dark Gray",  IDM_BKGND_DKGRAY
        MENUITEM "&Black",      IDM_BKGND_BLACK
END
```

```
  POPUP "&Timer"
  BEGIN
            MENUITEM "&Start",      IDM_TIMER_START
            MENUITEM "S&top",     IDM_TIMER_STOP, GRAYED
  END
  POPUP "&Help"
  BEGIN
           MENUITEM "&Help...",               IDM_APP_HELP
          MENUITEM "&About MenuDemo...",   IDM_APP_ABOUT
  END
END
```

```
RESOURCE.H
// Microsoft Developer Studio generated include file.
// Used by MenuDemo.rc
#define IDM_FILE_NEW              40001
#define IDM_FILE_OPEN             40002
#define IDM_FILE_SAVE             40003
#define IDM_FILE_SAVE_AS          40004
```

```
#define IDM_APP_EXIT            40005
#define IDM_EDIT_UNDO            40006
#define IDM_EDIT_CUT            40007
#define IDM_EDIT_COPY            40008
#define IDM_EDIT_PASTE            40009
#define IDM_EDIT_CLEAR            40010
#define IDM_BKGND_WHITE            40011
#define IDM_BKGND_LTGRAY            40012
#define IDM_BKGND_GRAY            40013
#define IDM_BKGND_DKGRAY            40014
#define IDM_BKGND_BLACK            40015
#define IDM_TIMER_START            40016
#define IDM_TIMER_STOP            40017
#define IDM_APP_HELP            40018
#define IDM_APP_ABOUT            40019
```

MENUDEMO.RCMenuDemo&MENUITEM
Menu Item PropertiesSeparator Checked Grayed BackgroundRESOURCE.H

FileEditWM_COMMANDMENUDEMO **Background** MENUDEMO.RC **White**IDIDM_BKGND_WHITE

CHECKEDMENUDEMO.CiSelectionIDM_BKGND_WHITE

BackgroundMENUDEMO.CWM_COMMANDwParam **Background**

```
hMenu = GetMenu (hwnd) ;
```

CheckMenuItem

```
CheckMenuItem (hMenu, iSelection, MF_UNCHECKED) ;
```

iSelectionwParam

```
iSelection = wParam ;

CheckMenuItem (hMenu, iSelection, MF_CHECKED) ;
```

Windows

TimerStartStopStopStartMENUDEMO
StartStop

```
EnableMenuItem (hMenu, IDM_TIMER_START, MF_GRAYED) ;

EnableMenuItem (hMenu, IDM_TIMER_STOP,  MF_ENABLED) ;
```

WM_COMMANDwParamIDM_TIMER_STOPMENUDEMO
Stop

```
EnableMenuItem (hMenu, IDM_TIMER_START, MF_ENABLED) ;

EnableMenuItem (hMenu, IDM_TIMER_STOP,  MF_GRAYED) ;
```

MENUDEMOwParamIDM_TIMER_STARTWM_COMMAND
MENUDEMOwParamIDM_TIMER_STOPWM_COMMAND

MENUDEMOWM_COMMANDwParamIDM_APP_ABOUT
IDM_APP_HELPMENUDEMO

MENUDEMOWM_COMMANDwParamIDM_APP_EXITWM_CLOSE
DefWindowProcWM_SYSCOMMANDwParamSC_CLOSE

MENUDEMO **FileEdit**WindowsWindows **FileEdit**WindowsA

FileEditWindowsWindows

MENUITEM

CreateMenuAppendMenuCreateWindowSetMenu

CreateMenu

```
hMenu = CreateMenu () ;
```

AppendMenu10-5MENUDEMOASCII

```
10-5  MENUDEMOC

hMenu = CreateMenu () ;

hMenuPopup = CreateMenu () ;

AppendMenu        (hMenuPopup, MF_STRING, IDM_FILE_NEW, "&
```

```
AppendMenu      (hMenuPopup, MF_STRING, IDM_FILE_OPEN, "
AppendMenu      (hMenuPopup, MF_STRING, IDM_FILE_SAVE, "&
AppendMenu      (hMenuPopup,  MF_STRING, IDM_FILE_SAVE_A
AppendMenu      (hMenuPopup, MF_SEPARATOR, 0, NULL) ;
AppendMenu      (hMenuPopup, MF_STRING, IDM_APP_EXIT, "E&
AppendMenu      (hMenu, MF_POPUP, hMenuPopup, "&File") ;

hMenuPopup = CreateMenu () ;
AppendMenu      (hMenuPopup, MF_STRING, IDM_EDIT_UNDO,"
AppendMenu      (hMenuPopup, MF_SEPARATOR, 0, NULL) ;
AppendMenu      (hMenuPopup, MF_STRING,IDM_EDIT_CUT, "Cu
AppendMenu      (hMenuPopup, MF_STRING,IDM_EDIT_COPY,"&
AppendMenu      (hMenuPopup,  MF_STRING,IDM_EDIT_PASTE,"
AppendMenu      (hMenuPopup, MF_STRING,IDM_EDIT_CLEAR,"
AppendMenu      (hMenu,     MF_POPUP,    hMenuPopup, "&Ed

hMenuPopup = CreateMenu () ;
AppendMenu (hMenuPopup, MF_STRING| MF_CHECKED,      IDM
AppendMenu (hMenuPopup, MF_STRING,        IDM_BKGND_LTGR
AppendMenu (hMenuPopup, MF_STRING,              IDM_BKGNI
```

```
AppendMenu (hMenuPopup, MF_STRING,         IDM_BKGND_DKG
AppendMenu (hMenuPopup,    MF_STRING,         IDM_BKGND_BL

AppendMenu (hMenu, MF_POPUP, hMenuPopup, "&Background")
hMenuPopup = CreateMenu () ;
AppendMenu (hMenuPopup, MF_STRING,         IDM_TIMER_STAR
AppendMenu (hMenuPopup, MF_STRING | MF_GRAYED, IDM_TIME

AppendMenu (hMenu,       MF_POPUP, hMenuPopup, "&Timer") ;

hMenuPopup =      CreateMenu () ;

AppendMenu (hMenuPopup, MF_STRING,    IDM_HELP_HELP,
AppendMenu (hMenuPopup, MF_STRING,    IDM_APP_ABOUT,

AppendMenu (hMenu, MF_POPUP, hMenuPopup, "&Help") ;
```

IDWindowsLoadMenuIndirect
MENUITEMTEMPLATEWindows

10-6POPMENU

10-6　POPMENU

```
POPMENU.C

/*---------------------------------------------------------------------
  POPMENU.C -- Popup Menu Demonstration
                                        (c) Charles Petzold, 1998
---------------------------------------------------------------------*/

#include <windows.h>
#include "resource.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

HINSTANCE hInst ;
TCHAR             szAppName[] = TEXT ("PopMenu") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                          PSTR szCmdLine, int iCmdSh
{
          HWND                  hwnd ;
          MSG                   msg ;
          WNDCLASS              wndclass ;
```

```
     wndclass.style                  = CS_HREDRAW | CS_V
     wndclass.lpfnWndProc              = WndProc ;
     wndclass.cbClsExtra               = 0 ;
     wndclass.cbWndExtra               = 0 ;
     wndclass.hInstance                = hInstance ;
     wndclass.hIcon                  = LoadIcon (NULL, szA
     wndclass.hCursor                = LoadCursor (NULL,
     wndclass.hbrBackground          = (HBRUSH) GetStockO
    wndclass.lpszMenuName            = NULL ;
     wndclass.lpszClassName          = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (  NULL, TEXT ("This program requires W
                                          szAppName, MB_IC
          return 0 ;
}

     hInst = hInstance ;
```

```
        hwnd = CreateWindow ( szAppName, TEXT ("Popup Menu
                WS_OVERLAPPEDWINDOW,
                CW_USEDEFAULT, CW_USEDEFAULT,
                CW_USEDEFAULT, CW_USEDEFAULT,
                NULL, NULL, hInstance, NULL) ;
        ShowWindow (hwnd, iCmdShow) ;
        UpdateWindow (hwnd) ;

        while (GetMessage (&msg, NULL, 0, 0))
        {
                TranslateMessage (&msg) ;
                DispatchMessage (&msg) ;
        }
        return msg.wParam ;
}

LRESULT CALLBACK WndProc ( HWND hwnd, UINT message, WPA
{
        static HMENU  hMenu ;
```

```
     static int          idColor [5] = {WHITE_BRUSH, LTGRAY_BR
                            DKGRAY_BRUSH, BLACK_BRUSH } ;
     static int          iSelection    = IDM_BKGND_WHITE ;
     POINT                          point ;

     switch (message)
{
     case   WM_CREATE:
          hMenu = LoadMenu (hInst, szAppName) ;
          hMenu = GetSubMenu (hMenu, 0) ;
          return 0 ;

    case   WM_RBUTTONUP:
          point.x = LOWORD (lParam) ;
          point.y = HIWORD (lParam) ;
          ClientToScreen (hwnd, &point) ;

          TrackPopupMenu (hMenu, TPM_RIGHTBUTTON, point.
          return 0 ;
```

```
case   WM_COMMAND:
       switch (LOWORD (wParam))
       {
       case   IDM_FILE_NEW:
       case   IDM_FILE_OPEN:
       case   IDM_FILE_SAVE:
       case   IDM_FILE_SAVE_AS:
       case   IDM_EDIT_UNDO:
       case   IDM_EDIT_CUT:
       case   IDM_EDIT_COPY:
       case   IDM_EDIT_PASTE:
       case   IDM_EDIT_CLEAR:
                        MessageBeep (0) ;
                        return 0 ;

       case   IDM_BKGND_WHITE:            // Note: Logic bel
      case   IDM_BKGND_LTGRAY:           //   assumes that I
```

```
            case   IDM_BKGND_GRAY:              //   through IDM_B
            case   IDM_BKGND_DKGRAY:            //   consecutive r
            case   IDM_BKGND_BLACK:             //   the order show

                        CheckMenuItem (hMenu, iSelection, M
                        iSelection = LOWORD (wParam) ;
                        CheckMenuItem (hMenu, iSelection, M

                       SetClassLong (hwnd, GCL_HBRBACKGR
                                          GetStockObject
                     (idColor [LOWORD (wParam) - IDM_BKGND_WH

                        InvalidateRect (hwnd, NULL, TRUE) ;
                       return 0 ;

            case   IDM_APP_ABOUT:
                        MessageBox (hwnd, TEXT ("Popup Mer
                   TEXT ("(c) Charles Petzold, 1998"),
                         szAppName, MB_ICONINFORMATION | MB_
```

```
                    return 0 ;

          case   IDM_APP_EXIT:
                    SendMessage (hwnd, WM_CLOSE, 0, 0
                    return 0 ;

          case   IDM_APP_HELP:
                    MessageBox (hwnd, TEXT ("Help not ye
               szAppName, MB_ICONEXCLAMATION | MB_OK) ;
                    return 0 ;
          }
          break ;

     case   WM_DESTROY:
          PostQuitMessage (0) ;
          return 0 ;
}
     return DefWindowProc (hwnd, message, wParam, lParam)
```

```
}
```

POPMENU.RC

```
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
/////////////////////////////////////////////////////////////////////////////
// Menu
POPMENU MENU DISCARDABLE
BEGIN
   POPUP "MyMenu"
   BEGIN
                POPUP "&File"
                BEGIN
                MENUITEM "&New",                        IDM_FILE_NEW
                MENUITEM "&Open",                IDM_FILE_OPEN
                MENUITEM "&Save",                IDM_FILE_SAVE
                MENUITEM "Save &As",     IDM_FILE_SAVE_AS
                MENUITEM SEPARATOR
```

```
            MENUITEM "E&xit",                   IDM_APP_EXIT
           END
            POPUP "&Edit"
            BEGIN
                MENUITEM "&Undo",           IDM_EDIT_UNDO
                MENUITEM SEPARATOR
                MENUITEM "Cu&t",            IDM_EDIT_CUT
                MENUITEM "&Copy",           IDM_EDIT_COPY
                MENUITEM "&Paste",          IDM_EDIT_PASTE
                MENUITEM "De&lete",         IDM_EDIT_CLEAR
END
   POPUP "&Background"
BEGIN
  MENUITEM "&White",                IDM_BKGND_WHITE, CHECKE
MENUITEM "&Light Gray",       IDM_BKGND_LTGRAY
MENUITEM "&Gray",                 IDM_BKGND_GRAY
MENUITEM "&Dark Gray",            IDM_BKGND_DKGRAY
MENUITEM "&Black",              IDM_BKGND_BLACK
```

```
END
    POPUP "&Help"
BEGIN
MENUITEM "&Help...",                IDM_APP_HELP
    MENUITEM "&About PopMenu...", IDM_APP_ABOUT
END
  END
END
```

RESOURCE.H

```
// Microsoft Developer Studio generated include file.
// Used by PopMenu.rc
#define IDM_FILE_NEW                 40001
#define IDM_FILE_OPEN                40002
#define IDM_FILE_SAVE                40003
#define IDM_FILE_SAVE_AS             40004
#define IDM_APP_EXIT                 40005
#define IDM_EDIT_UNDO                40006
#define IDM_EDIT_CUT                 40007
```

```
#define IDM_EDIT_COPY               40008
#define IDM_EDIT_PASTE              40009
#define IDM_EDIT_CLEAR              40010
#define IDM_BKGND_WHITE             40011
#define IDM_BKGND_LTGRAY            40012
#define IDM_BKGND_GRAY              40013
#define IDM_BKGND_DKGRAY            40014
#define IDM_BKGND_BLACK             40015
#define IDM_APP_HELP                40016
#define IDM_APP_ABOUT               40017
```

POPMENU.RCMENUDEMO.RCMyMenuFileEdit
BackgroundHelp

WndProcWM_CREATEPOPMENUMyMenu

```
hMenu = LoadMenu (hInst, szAppName) ;
hMenu = GetSubMenu (hMenu, 0) ;
```

WM_RBUTTONUPPOPMENUTrackPopupMenu

```
point.x = LOWORD (lParam) ;
point.y = HIWORD (lParam) ;
```

```
ClientToScreen (hwnd, &point) ;
TrackPopupMenu (hMenu, TPM_RIGHTBUTTON, point.x, point.y,
                                    0, hwnd, NULL) ;
```

WindowsFileEditBackgroundHelp

TrackPopupMenuMicrosoft

WS_SYSMENUWindowsAboutID
0xF000WindowsIDWM_SYSCOMMANDWM_SYSCOMMAND
DefWindowProc

10-7POORMENU

```
10-7  POORMENU

POORMENU.C

/*--------------------------------------------------------------------
  POORMENU.C --        The Poor Person's Menu
                                        (c) Charles Petzold, 1998
----------------------------------------------------------------------*/

#include <windows.h>
#define IDM_SYS_ABOUT                    1
#define IDM_SYS_HELP                     2
```

```
#define IDM_SYS_REMOVE                  3

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
static TCHAR szAppName[] = TEXT ("PoorMenu") ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                    PSTR szCmdLine, int iCmdSh
{
     HMENU                   hMenu ;
     HWND                    hwnd ;
     MSG                     msg ;
     WNDCLASS                wndclass ;

     wndclass.style                  = CS_HREDRAW | CS_V
     wndclass.lpfnWndProc              = WndProc ;
     wndclass.cbClsExtra               = 0 ;
     wndclass.cbWndExtra               = 0 ;
     wndclass.hInstance                = hInstance ;
     wndclass.hIcon                  = LoadIcon (NULL, IDI_
     wndclass.hCursor                  = LoadCursor (NULL,
```

```
        wndclass.hbrBackground          = (HBRUSH) GetStockO
        wndclass.lpszMenuName           = NULL ;
        wndclass.lpszClassName          = szAppName ;

        if (!RegisterClass (&wndclass))
        {
             MessageBox (  NULL, TEXT ("This program requires W
                                                 szAppName, MB_IC
             return 0 ;
}

        hwnd = CreateWindow ( szAppName, TEXT ("The Poor-Pe
                  WS_OVERLAPPEDWINDOW,
                  CW_USEDEFAULT, CW_USEDEFAULT,
                  CW_USEDEFAULT, CW_USEDEFAULT,
                  NULL, NULL, hInstance, NULL) ;

        hMenu = GetSystemMenu (hwnd, FALSE) ;
        AppendMenu    (hMenu, MF_SEPARATOR, 0,
```

```
        AppendMenu    (hMenu, MF_STRING, IDM_SYS_ABOUT,    T
        AppendMenu    (hMenu, MF_STRING, IDM_SYS_HELP,     TE
        AppendMenu    (hMenu, MF_STRING, IDM_SYS_REMOVE,

        ShowWindow (hwnd, iCmdShow) ;
        UpdateWindow (hwnd) ;

        while (GetMessage (&msg, NULL, 0, 0))
        {
             TranslateMessage (&msg) ;
             DispatchMessage (&msg) ;
        }
        return msg.wParam ;
}

LRESULT CALLBACK WndProc ( HWND hwnd, UINT message, WPA
{
  switch (message)
        {
```

```
    case   WM_SYSCOMMAND:
         switch (LOWORD (wParam))
         {
         case   IDM_SYS_ABOUT:
                        MessageBox (  hwnd, TEXT ("A Poor-Pe
           TEXT ("(c) Charles Petzold, 1998"),
           szAppName, MB_OK | MB_ICONINFORMATION) ;
                        return 0 ;

         case   IDM_SYS_HELP:
                        MessageBox (  hwnd, TEXT ("Help not
           szAppName, MB_OK | MB_ICONEXCLAMATION) ;
                        return 0 ;

         case   IDM_SYS_REMOVE:
                        GetSystemMenu (hwnd, TRUE) ;
                        return 0 ;
         }
```

```
            break ;


             case   WM_DESTROY:
                          PostQuitMessage (0) ;
                          return 0 ;
          }
          return DefWindowProc (hwnd, message, wParam, lParam)
}
```

IDPOORMENU.C

```
#define IDM_ABOUT          1
#define IDM_HELP           2
#define IDM_REMOVE         3
```

POORMENU

```
hMenu = GetSystemMenu (hwnd, FALSE) ;
```

GetSystemMenuFALSE

AppendMenu

```
AppendMenu (hMenu,       MF_SEPARATOR, 0,
```

```
AppendMenu (hMenu,      MF_STRING, IDM_SYS_ABOUT,    TEXT
AppendMenu (hMenu,      MF_STRING, IDM_SYS_HELP,       TE
AppendMenu (hMenu,      MF_STRING, IDM_SYS_REMOVE,   TEX
```

AppendMenuRemove AdditionsPOORMENUTRUI
GetSystemMenu

```
GetSystemMenu (hwnd, TRUE) ;
```

RestoreMoveSizeMinimizeMaximizeClosewParam
SC_RESTORESC_MOVESC_SIZESC_MINIMUMSC_MAXIMUM
SC_CLOSEWM_SYSCOMMANDWindowsDefWindowProc
WindowsSC_NEXTWINDOWSC_PREVWINDOWSC_VSCROLL
SC_HSCROLLSC_ARRANGE

AppendMenuWindows
WindowsChangeMenuChangeMenu

- **AppendMenu**
- **DeleteMenu**
- **InsertMenu**
- **ModifyMenu**
- **RemoveMenu**

DeleteMenuRemoveMenuDeleteMenuRemoveMenu

Windows

```
DrawMenuBar (hwnd) ;
```

DrawMenuBar

```
hMenuPopup = GetSubMenu (hMenu, iPosition) ;
```

iPositionhMenu0AppendMenu

```
iCount = GetMenuItemCount (hMenu) ;
```

ID

```
id = GetMenuItemID (hMenuPopup, iPosition) ;
```

iPosition0

MENUDEMO

```
CheckMenuItem (hMenu, id, iCheck) ;
```

MENUDEMOhMenuidIDiCheckMF_CHECKED
MF_UNCHECKEDhMenuidIDMF_BYPOSITION

```
CheckMenuItem (hMenu, iPosition, MF_CHECKED | MF_BYPOSITIC
```

```

```

MF_ENABLEDMF_DISABLEDMF_GRAYEDEnableMenuItem
CheckMenuItemEnableMenuItemMF_BYPOSITIONID
EnableMenuItem HiliteMenuItemCheckMenuItemEnableMenuItem
MF_HILITEMF_UNHILITEWindowsHiliteMenuItem

```
iCharCount = GetMenuString (hMenu, id, pString, iMaxCount, iFl
```

iFlagMF_BYCOMMANDidIDMF_BYPOSITIONid
iMaxCountpString

```
iFlags = GetMenuState (hMenu, id, iFlag) ;
```

iFlagMF_BYCOMMANDMF_BYPOSITIONiFlags
MF_DISABLEDMF_GRAYEDMF_CHECKEDMF_MENUBREAK
MF_MENUBARBREAKMF_SEPARATOR

```
DestroyMenu (hMenu) ;
```

SetMenuLotus
EditMENUDEMO

```
NOPOPUPS.C

/*------------------------------------------------------------------------
  NOPOPUPS.C --        Demonstrates No-Popup Nested Menu
                                          (c) Charles Petzold, 1998
------------------------------------------------------------------------*/
#include <windows.h>
#include "resource.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                          PSTR szCmdLine, int iCmdSh
{
          static TCHAR szAppName[] = TEXT ("NoPopUps") ;
          HWND                         hwnd ;
          MSG                          msg ;
          WNDCLASS                     wndclass ;

          wndclass.style                    = CS_HREDRAW | CS_VR
          wndclass.lpfnWndProc                   = WndProc ;
```

```
        wndclass.cbClsExtra                    = 0 ;
        wndclass.cbWndExtra                    = 0 ;
        wndclass.hInstance                     = hInstance ;
        wndclass.hIcon                         = LoadIcon (NULL, IDI_
        wndclass.hCursor                       = LoadCursor (NULL,
        wndclass.hbrBackground                 = (HBRUSH) GetStockO
        wndclass.lpszMenuName                  = NULL ;
        wndclass.lpszClassName = szAppName ;
        if (!RegisterClass (&wndclass))
        {
        MessageBox (  NULL, TEXT ("This program requires Windo
                                        szAppName, MB_ICONERRO
        return 0 ;
}


        hwnd = CreateWindow (szAppName,
            TEXT ("No-Popup Nested Menu Demonstration"),
            WS_OVERLAPPEDWINDOW,
```

```
            CW_USEDEFAULT, CW_USEDEFAULT,
            CW_USEDEFAULT, CW_USEDEFAULT,
            NULL, NULL, hInstance, NULL) ;

        ShowWindow (hwnd, iCmdShow) ;
        UpdateWindow (hwnd) ;

        while (GetMessage (&msg, NULL, 0, 0))
        {
            TranslateMessage (&msg) ;
            DispatchMessage (&msg) ;
        }
        return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
        static HMENU hMenuMain, hMenuEdit, hMenuFile ;
        HINSTANCE           hInstance ;
```

```
     switch (message)
{
     case   WM_CREATE:
          hInstance = (HINSTANCE) GetWindowLong (hwnd, GW

          hMenuMain = LoadMenu (hInstance, TEXT ("MenuMa
          hMenuFile = LoadMenu (hInstance, TEXT ("MenuFile"
          hMenuEdit = LoadMenu (hInstance, TEXT ("MenuEdit

          SetMenu (hwnd, hMenuMain) ;
          return 0 ;

     case   WM_COMMAND:
          switch (LOWORD (wParam))
          {
          case   IDM_MAIN:
                         SetMenu (hwnd, hMenuMain) ;
                         return 0 ;
```

```
            case   IDM_FILE:
                        SetMenu (hwnd, hMenuFile) ;
                        return 0 ;

            case   IDM_EDIT:
                        SetMenu (hwnd, hMenuEdit) ;
                        return 0 ;

            case   IDM_FILE_NEW:
            case   IDM_FILE_OPEN:
            case   IDM_FILE_SAVE:
            case   IDM_FILE_SAVE_AS:
            case   IDM_EDIT_UNDO:
            case   IDM_EDIT_CUT:
            case   IDM_EDIT_COPY:
            case   IDM_EDIT_PASTE:
            case   IDM_EDIT_CLEAR:
                        MessageBeep (0) ;
```

```
                                return 0 ;
                    }
                    break ;

          case   WM_DESTROY:
                    SetMenu (hwnd, hMenuMain) ;
                   DestroyMenu (hMenuFile) ;
                    DestroyMenu (hMenuEdit) ;

                    PostQuitMessage (0) ;
                    return 0 ;
     }
          return DefWindowProc (hwnd, message, wParam, lParam)
}
```

```
NOPOPUPS.RC
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
//////////////////////////////////////////////////////////////////////////////
```

```
// Menu
MENUMAIN MENU DISCARDABLE
BEGIN
   MENUITEM "MAIN:",         0, INACTIVE
   MENUITEM "&File...",  IDM_FILE
   MENUITEM "&Edit...", IDM_EDIT
END
MENUFILE MENU DISCARDABLE
BEGIN
   MENUITEM "FILE:",            0, INACTIVE
   MENUITEM "&New",                                        IDM
   MENUITEM "&Open...",                                IDM_FILE_O
   MENUITEM "&Save",                                     IDM_FILE
   MENUITEM "Save &As",                                IDM_FILE_
   MENUITEM "(&Main)",                                       IDM
END
MENUEDIT MENU DISCARDABLE
BEGIN
```

```
  MENUITEM "EDIT:",                       0, INACTIVE
  MENUITEM "&Undo",                                       IDM_E
  MENUITEM "Cu&t",                IDM_EDIT_CUT
  MENUITEM "&Copy",                IDM_EDIT_COPY
  MENUITEM "&Paste",               IDM_EDIT_PASTE
  MENUITEM "De&lete",              IDM_EDIT_CLEAR
  MENUITEM "(&Main)",              IDM_MAIN
END
```

RESOURCE.H

```
// Microsoft Developer Studio generated include file.
// Used by NoPopups.rc
#define IDM_FILE                40001
#define IDM_EDIT                40002
#define IDM_FILE_NEW            40003
#define IDM_FILE_OPEN           40004
#define IDM_FILE_SAVE           40005
#define IDM_FILE_SAVE_AS        40006
#define IDM_MAIN                40007
```

```
#define IDM_EDIT_UNDO            40008

#define IDM_EDIT_CUT             40009

#define IDM_EDIT_COPY            40010

#define IDM_EDIT_PASTE           40011

#define IDM_EDIT_CLEAR           40012
```

Microsoft Developer StudioInsertResource
WM_CREATEWindows

```
hMenuMain = LoadMenu (hInstance, TEXT ("MenuMain")) ;

hMenuFile = LoadMenu (hInstance, TEXT ("MenuFile")) ;

hMenuEdit = LoadMenu (hInstance, TEXT ("MenuEdit")) ;
```

```
SetMenu (hwnd, hMenuMain) ;
```

MAIN:File...Edit...MAIN:WM_COMMAND
FileEditFILE:EDIT:(Main)

```
case      WM_COMMAND :

        switch (wParam)

        {

        case   IDM_MAIN :
```

```
        SetMenu (hwnd, hMenuMain) ;
            return 0 ;
        case   IDM_FILE :
            SetMenu (hwnd, hMenuFile) ;
            return 0 ;

        case   IDM_EDIT :
            SetMenu (hwnd, hMenuEdit) ;
            return 0 ;



  }
      break ;
```

WM_DESTROYNOPOPUPSDestroyMenuFileEdit

WM_COMMANDWM_SYSCOMMAND
WindowsDeleteClearEditDel
WM_COMMAND

?WM_KEYDOWNWM_CHAR

WindowsWindowsWM_COMMANDWindows

TranslateAccelerator

WM_COMMAND

ShiftCtrlAltWindowsTabEnterEscSpacebar

EditWindows

10-2

Undo	Alt+Backspace	Ctrl+Z
Cut	Shift+Del	Ctrl+X
Copy	Ctrl+Ins	Ctrl+C
Paste	Shift+Ins	Ctrl+V
DeleteClear	Del	Del

F1F4F5F6MDI

Developer Studio

Accel PropertiesIDID

ASCIIShiftCtrlAlt^CtrlASCII

\tCtrlShiftAlt+Shift+F6
Ctrl+F6

LoadAccelerators Loac
LoadMenu

HANDLE

```
HANDLE hAccel ;
```

```
hAccel = LoadAccelerators (hInstance, TEXT ("MyAccelerators"))
```

LoadAcceleratorsMAKEINTRESOURCE#

Windows

```
while      (GetMessage (&msg, NULL, 0, 0))

{

  TranslateMessage (&msg) ;

    DispatchMessage (&msg) ;

}
```

```
while      (GetMessage (&msg, NULL, 0, 0))

{

  if (!TranslateAccelerator (hwnd, hAccel, &msg))

   {

            TranslateMessage (&msg) ;

            DispatchMessage (&msg) ;

        }

}
```

TranslateAcceleratormsghAccelhwndID
WM_SYSCOMMANDWM_COMMAND

TranslateAccelerator0TranslateAccelerator
TranslateMessageDispatchMessageGetMessage

TranslateMessagehwndmsghwnd

msgGetMessageGetMessageNULLGetMessagemsghwnd
TranslateAcceleratorWM_COMMANDWM_SYSCOMMANDhwnd
msg.hwndWindowsTranslateAccelerator

TranslateAcceleratorWM_SYSCOMMANDTranslateAccelerator
WM_COMMANDWM_COMMAND

LOWORD (wParam)	ID	ID	ID
HIWORD (wParam)	1	0	
lParam	0	0	

WM_INITMENUWM_INITMENUPOPUPWM_MENUSELECT
WM_INITMENUPOPUPTranslateAccelerator
WM_COMMANDWM_SYSCOMMAND

TranslateAcceleratorWM_SYSCOMMANDWM_COMMAND
TranslateAcceleratorWM_COMMAND

POPPAD

POPPAD1FileEditPOPPAD2Edit File
PrintPOPPAD210-9

10-9 POPPAD2

```
POPPAD2.C

/*------------------------------------------------------------------------

  POPPAD2.C --         Popup Editor Version 2 (includes menu)

                                        (c) Charles Petzold, 1998
```

```
------------------------------------------------------------------------*/
#include <windows.h>
#include "resource.h"

#define ID_EDIT                 1
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM);
TCHAR szAppName[] = TEXT ("PopPad2") ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                        PSTR szCmdLine, int iCmdSh
{
          HACCEL             hAccel ;
          HWND                 hwnd ;
          MSG                  msg ;
          WNDCLASS             wndclass ;
          wndclass.style                          = CS_HREDRAW |
          wndclass.lpfnWndProc                      = WndProc ;
          wndclass.cbClsExtra                     = 0 ;
          wndclass.cbWndExtra                       = 0 ;
          wndclass.hInstance                      = hInstance ;
```

```
wndclass.hIcon                          = LoadIcon (hInst
wndclass.hCursor                         = LoadCursor (N
wndclass.hbrBackground                  = (HBRUSH) GetS
wndclass.lpszMenuName                    = szAppName ;
wndclass.lpszClassName                  = szAppName ;

if (!RegisterClass (&wndclass))
{
     MessageBox (  NULL, TEXT ("This program requires W
      szAppName, MB_ICONERROR) ;
     return 0 ;
}

hwnd = CreateWindow (szAppName, szAppName,
        WS_OVERLAPPEDWINDOW,
       GetSystemMetrics (SM_CXSCREEN) / 4,
      GetSystemMetrics (SM_CYSCREEN) / 4,
       GetSystemMetrics (SM_CXSCREEN) / 2,
```

```
                GetSystemMetrics (SM_CYSCREEN) / 2,
                NULL, NULL, hInstance, NULL) ;

        ShowWindow (hwnd, iCmdShow) ;
        UpdateWindow (hwnd) ;

        hAccel = LoadAccelerators (hInstance, szAppName) ;
        while (GetMessage (&msg, NULL, 0, 0))
        {
             if (!TranslateAccelerator (hwnd, hAccel, &msg))
                  {
                       TranslateMessage (&msg) ;
                       DispatchMessage (&msg) ;
                  }
        }
        return msg.wParam ;
}

AskConfirmation (HWND hwnd)
```

```
{
        return MessageBox (   hwnd, TEXT ("Really want to close
                    szAppName, MB_YESNO | MB_ICONQUESTION) ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
        static HWND   hwndEdit ;
        int                          iSelect, iEnable ;


        switch (message)
        {
        case   WM_CREATE:
             hwndEdit = CreateWindow (TEXT ("edit"), NULL,
             WS_CHILD | WS_VISIBLE | WS_HSCROLL | WS_VSCRO
             WS_BORDER | ES_LEFT | ES_MULTILINE |
             ES_AUTOHSCROLL | ES_AUTOVSCROLL,
             0, 0, 0, 0, hwnd, (HMENU) ID_EDIT,
             ((LPCREATESTRUCT) lParam)->hInstance, NULL) ;
```

```
            return 0 ;

        case   WM_SETFOCUS:
             SetFocus (hwndEdit) ;
             return 0 ;

        case   WM_SIZE:
             MoveWindow (hwndEdit, 0, 0, LOWORD (lParam), HIW
             return 0 ;

        case   WM_INITMENUPOPUP:
             if (lParam == 1)
             {
                                 EnableMenuItem ((HMENU) wParam, ID
                       SendMessage (hwndEdit, EM_CANUNDO, 0, 0) ?
                        MF_ENABLED : MF_GRAYED) ;

                                 EnableMenuItem ((HMENU) wParam, ID
                       IsClipboardFormatAvailable (CF_TEXT) ?
```

```
                MF_ENABLED : MF_GRAYED) ;

                        iSelect = SendMessage (hwndEdit, EM

                        if (HIWORD (iSelect) == LOWORD (iSe
                                iEnable = MF_GRAYED ;
                        else
                                iEnable = MF_ENABLED ;

    EnableMenuItem ((HMENU) wParam, IDM_EDIT_CUT,  iE
    EnableMenuItem ((HMENU) wParam, IDM_EDIT_COPY,  i
    EnableMenuItem ((HMENU) wParam, IDM_EDIT_CLEAR,
                        return 0 ;
        }
        break ;
  case   WM_COMMAND:
        if (lParam)
        {
```

```
                    if (LOWORD (lParam) == ID_EDIT &&
            (HIWORD (wParam) == EN_ERRSPACE ||
            HIWORD (wParam) == EN_MAXTEXT))
          MessageBox (hwnd, TEXT ("Edit control out of space
            szAppName, MB_OK | MB_ICONSTOP) ;
                    return 0 ;
}
        else   switch (LOWORD (wParam))
        {
        case   IDM_FILE_NEW:
        case   IDM_FILE_OPEN:
       case   IDM_FILE_SAVE:
        case   IDM_FILE_SAVE_AS:
        case   IDM_FILE_PRINT:
                        MessageBeep (0) ;
                        return 0 ;

        case   IDM_APP_EXIT:
                        SendMessage (hwnd, WM_CLOSE, 0, 0
```

```
                    return 0 ;

          case   IDM_EDIT_UNDO:
                    SendMessage (hwndEdit, WM_UNDO, 0
                    return 0 ;

          case   IDM_EDIT_CUT:
                    SendMessage (hwndEdit, WM_CUT, 0,
                    return 0 ;

          case   IDM_EDIT_COPY:
                    SendMessage (hwndEdit, WM_COPY, 0
                    return 0 ;

          case   IDM_EDIT_PASTE:
                    SendMessage (hwndEdit, WM_PASTE, 0
                    return 0 ;

          case   IDM_EDIT_CLEAR:
```

```
                    SendMessage (hwndEdit, WM_CLEAR,
                    return 0 ;

        case   IDM_EDIT_SELECT_ALL:
                    SendMessage (hwndEdit, EM_SETSEL,
                    return 0 ;

            case   IDM_HELP_HELP:
                    MessageBox (hwnd, TEXT ("Help not y
                szAppName, MB_OK | MB_ICONEXCLAMATION) ;
                    return 0 ;

            case   IDM_APP_ABOUT:
                    MessageBox (hwnd, TEXT ("POPPAD2 (
                szAppName, MB_OK | MB_ICONINFORMATION) ;
                    return 0 ;
            }
            break ;
```

```
  case   WM_CLOSE:
               if (IDYES == AskConfirmation (hwnd))
                              DestroyWindow (hwnd) ;
               return 0 ;

          case   WM_QUERYENDSESSION:
               if (IDYES == AskConfirmation (hwnd))
                              return 1 ;
               else
                              return 0 ;

          case   WM_DESTROY:
               PostQuitMessage (0) ;
               return 0 ;
    }
          return DefWindowProc (hwnd, message, wParam, lParam)
}
```

POPPAD2.RC

```
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
//////////////////////////////////////////////////////////////////////////////
// Icon
POPPAD2                 ICON    DISCARDABLE     "poppad2.ico"
//////////////////////////////////////////////////////////////////////////////
// Menu
POPPAD2 MENU DISCARDABLE
BEGIN
   POPUP "&File"
    BEGIN
                MENUITEM "&New",                                IDM_FIL
                MENUITEM "&Open...",                    IDM_FILE_OPE
                MENUITEM "&Save",                           IDM_FILE_
                MENUITEM "Save &As...",                 IDM_FILE_SAV
                MENUITEM SEPARATOR
                MENUITEM "&Print",                          IDM_FILE_P
```

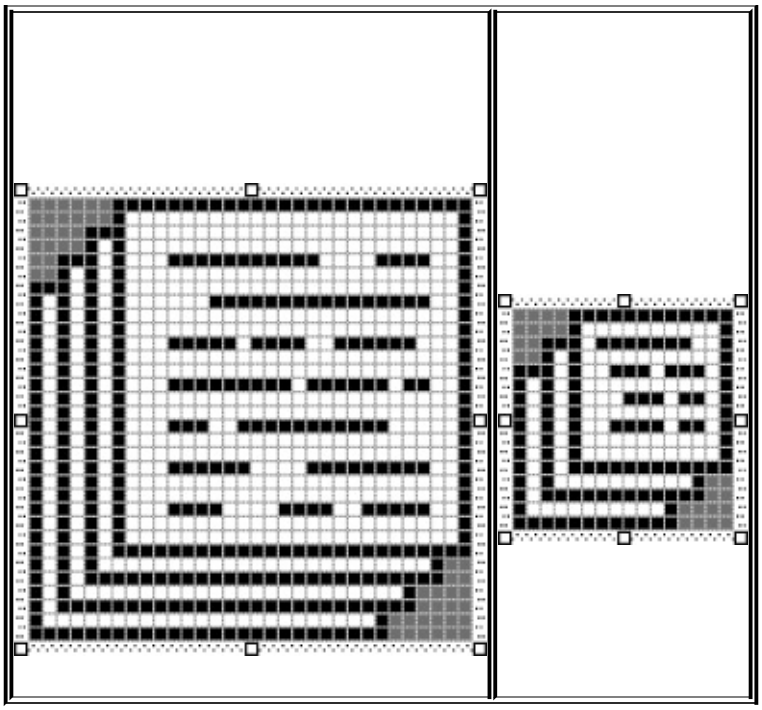
```
        MENUITEM SEPARATOR
        MENUITEM "E&xit",                       IDM_APP_EXIT
    END
    POPUP "&Edit"
    BEGIN
        MENUITEM "&Undo\tCtrl+Z",               IDM_EDIT_U
        MENUITEM SEPARATOR
        MENUITEM "Cu&t\tCtrl+X",                  IDM_EDIT_
        MENUITEM "&Copy\tCtrl+C",               IDM_EDIT_C
        MENUITEM "&Paste\tCtrl+V",              IDM_EDIT_PA
        MENUITEM "De&lete\tDel",                    IDM_ED
        MENUITEM SEPARATOR
        MENUITEM "&Select All",                 IDM_EDIT_S
    END
    POPUP "&Help"
    BEGIN
        MENUITEM "&Help...",                        IDM_HE
        MENUITEM "&About PopPad2...",       IDM_APP_ABOU
    END
```

```
END

//////////////////////////////////////////////////////////////////////
// Accelerator
POPPAD2 ACCELERATORS DISCARDABLE
BEGIN
  VK_BACK,                IDM_EDIT_UNDO,        VIRTKEY, ALT, N
  VK_DELETE,              IDM_EDIT_CLEAR,           VIRTKEY, N
  VK_DELETE,              IDM_EDIT_CUT,         VIRTKEY, SHIFT,
  VK_F1,                  IDM_HELP_HELP,        VIRTKEY, NOINVE
  VK_INSERT,              IDM_EDIT_COPY,        VIRTKEY, CONTR
  VK_INSERT,              IDM_EDIT_PASTE,       VIRTKEY, SHIFT,
  "^C",       IDM_EDIT_COPY,        ASCII,  NOINVERT
  "^V",       IDM_EDIT_PASTE,       ASCII,  NOINVERT
 "^X",       IDM_EDIT_CUT,         ASCII,  NOINVERT
  "^Z",       IDM_EDIT_UNDO,        ASCII,  NOINVERT
END
```

RESOURCE.H

```
// Microsoft Developer Studio generated include file.
// Used by POPPAD2.RC
#define IDM_FILE_NEW            40001
#define IDM_FILE_OPEN           40002
#define IDM_FILE_SAVE           40003
#define IDM_FILE_SAVE_AS        40004
#define IDM_FILE_PRINT          40005
#define IDM_APP_EXIT            40006
#define IDM_EDIT_UNDO           40007
#define IDM_EDIT_CUT            40008
#define IDM_EDIT_COPY           40009
#define IDM_EDIT_PASTE          40010
#define IDM_EDIT_CLEAR          40011
#define IDM_EDIT_SELECT_ALL     40012
#define IDM_HELP_HELP           40013
#define IDM_APP_ABOUT           40014
```

POPPAD2.ICO



POPPAD2.RC\tEdit

EditWM_INITMENUPOPUPEditEditFile0
1EditlParam1

UndoPOPPAD2EM_CANUNDOUndoSendMessage

```
EnableMenuItem (wParam, IDM_UNDO,

        SendMessage (hwndEdit, EM_CANUNDO, 0, 0) ?

                                MF_ENABLED : MF_GRAYED) ;
```

PasteCF_TEXTIsClipboardFormatAvailable

```
EnableMenuItem (wParam, IDM_PASTE,
```

```
        IsClipboardFormatAvailable (CF_TEXT) ? MF_ENABLED : M
```

CutCopyDeleteEM_GETSEL

```
iSelect = SendMessage (hwndEdit, EM_GETSEL, 0, 0) ;
```

iSelectiSelect

```
if (HIWORD (iSelect) == LOWORD (iSelect))
        iEnable = MF_GRAYED ;
else
        iEnable = MF_ENABLED ;
```

iEnableCutCopyDelete

```
EnableMenuItem (wParam, IDM_CUT,  iEnable) ;
EnableMenuItem (wParam, IDM_COPY, iEnable) ;
EnableMenuItem (wParam, IDM_DEL,  iEnable) ;
```

POPPAD2EditUndoCutCopyPaste
ClearSelect All

```
case    IDM_UNDO :
        SendMessage (hwndEdit, WM_UNDO, 0, 0) ;
```

```
        return 0 ;
case      IDM_CUT :
        SendMessage (hwndEdit, WM_CUT, 0, 0) ;
        return 0 ;

case      IDM_COPY :
        SendMessage (hwndEdit, WM_COPY, 0, 0) ;
        return 0 ;

case      IDM_PASTE :
        SendMessage (hwndEdit, WM_PASTE, 0, 0) ;
        return 0 ;

case      IDM_DEL :
        SendMessage (hwndEdit, WM_DEL, 0, 0) ;
        return 0 ;
case      IDM_SELALL :
        SendMessage (hwndEdit, EM_SETSEL, 0, -1) ;
        return 0 ;
```

IDM_UNDOIDM_CUTWM_UNDOWM_CUT

FileAbout

```
case       IDM_ABOUT :
           MessageBox (hwnd, TEXT ("POPPAD2 (c) Charles Petzold,
                                    szAppName, MB_OK | MB_ICONIN
           return 0 ;
```

HelpF1

ExitWM_CLOSE

```
case       IDM_EXIT :
           SendMessage (hwnd, WM_CLOSE, 0, 0) ;
           return 0 ;
```

DefWindowProcwParamSC_CLOSEWM_SYSCOMMAND

WM_CLOSEDefWindowProcDefWindowProcWM_CLOSE
DestroyWindowWM_CLOSEDefWindowProcPOPPAD2

```
case WM_CLOSE :
           if (IDYES == AskConfirmation (hwnd))
                DestroyWindow (hwnd) ;
  return 0 ;
```

AskConfirmationPOPPAD2

```
AskConfirmation (HWND hwnd)

{

        return MessageBox (hwnd, TEXT ("Really want to close Po

                        szAppName, MB_YESNO | MB_ICONQUESTION) ;

}
```

YesAskConfirmationIDYESDestroyWindow

WM_QUERYENDSESSIONWindowsWindows
WM_QUERYENDSESSION0Windows
WM_QUERYENDSESSION

```
case       WM_QUERYENDSESSION :

        if (IDYES == AskConfirmation (hwnd))

             return 1 ;

        else

              return 0 ;
```

WM_CLOSEWM_QUERYENDSESSIONPOPPAD2Exit
WM_CLOSE

WM_QUERYENDSESSIONWM_ENDSESSIONWindows
WM_QUERYENDSESSIONWM_QUERYENDSESSION0Windows
WM_ENDSESSIONwParam0WM_ENDSESSIONWindows

POPPAD2FileNewOpenSaveSave

…

Visual

Microsoft Windows 98Windows

WindowsWM_PAINT

W

OKCancel
Windows

About

WindowsAboutOKAbout
ABOUT111-1

11-1 ABOUT1

```
ABOUT1.C

/*------------------------------------------------------------------

  ABOUT1.C -- About Box Demo Program No. 1

                                        (c) Charles Petzold, 1998
```

```
------------------------------------------------------------------------*/

#include <windows.h>
#include "resource.h"

LRESULT     CALLBACK WndProc                (HWND, UINT, WPAR
BOOL             CALLBACK AboutDlgProc       (HWND, UINT, WPA

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                        PSTR szCmdLine, int iCmdSh
{
        static TCHAR szAppName[] = TEXT ("About1") ;
         MSG                         msg ;
         HWND                         hwnd ;
   WNDCLASS                        wndclass ;

         wndclass.style                      = CS_HREDRAW | CS_V
         wndclass.lpfnWndProc                  = WndProc ;
         wndclass.cbClsExtra                   = 0 ;
         wndclass.cbWndExtra                    = 0 ;
         wndclass.hInstance                    = hInstance ;
```

```
     wndclass.hIcon                       = LoadIcon (hInstance
     wndclass.hCursor                     = LoadCursor (NULL,
     wndclass.hbrBackground               = (HBRUSH) GetStockO
     wndclass.lpszMenuName                = szAppName ;
     wndclass.lpszClassName               = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Wi
        szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, TEXT ("About Box D
                          WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                        CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;
```

```
        ShowWindow (hwnd, iCmdShow) ;
        UpdateWindow (hwnd) ;

        while (GetMessage (&msg, NULL, 0, 0))
        {
                TranslateMessage (&msg) ;
                DispatchMessage (&msg) ;
        }
        return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
        static HINSTANCE hInstance ;
        switch (message)
        {
        case   WM_CREATE :
            hInstance = ((LPCREATESTRUCT) lParam)->hInstance ;
```

```
            return 0 ;

        case   WM_COMMAND :
            switch (LOWORD (wParam))
            {
            case IDM_APP_ABOUT :
                            DialogBox (hInstance, TEXT ("AboutBo
                           break ;
            }
            return 0 ;

        case   WM_DESTROY :
            PostQuitMessage (0) ;
            return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam)
}

BOOL CALLBACK AboutDlgProc (HWND hDlg, UINT message,WPA
```

```
{
        switch (message)
        {
        case   WM_INITDIALOG :
              return TRUE ;


        case   WM_COMMAND :
              switch (LOWORD (wParam))
              {
              case   IDOK :
              case   IDCANCEL :
                                  EndDialog (hDlg, 0) ;
                                  return TRUE ;
       }
              break ;
   }
  return FALSE ;
}
```

ABOUT1.RC

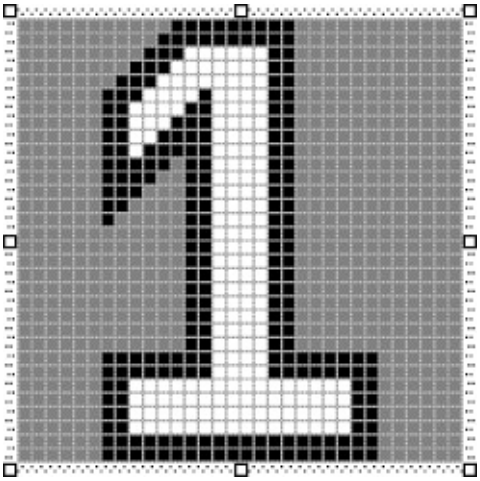
```
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
/////////////////////////////////////////////////////////////////////////////
// Dialog
ABOUTBOX DIALOG DISCARDABLE  32, 32, 180, 100
STYLE DS_MODALFRAME | WS_POPUP
FONT 8, "MS Sans Serif"
BEGIN
    DEFPUSHBUTTON           "OK",IDOK,66,80,50,14
    ICON                                    "ABOUT1",IDC_STATIC,7
    CTEXT                                    "About1",IDC_STATIC,4
    CTEXT                       "About Box Demo Program",IDC_STA
    CTEXT                       "(c) Charles Petzold,
1998",IDC_STATIC,7,52,166,8
END

/////////////////////////////////////////////////////////////////////////////
```

```
// Menu

ABOUT1      MENU DISCARDABLE

BEGIN

   POPUP "&Help"

  BEGIN

               MENUITEM "&About About1...",                          IDM

  END

END


////////////////////////////////////////////////////////////////////////////

// Icon

ABOUT1            ICON    DISCARDABLE     "About1.ico"
```

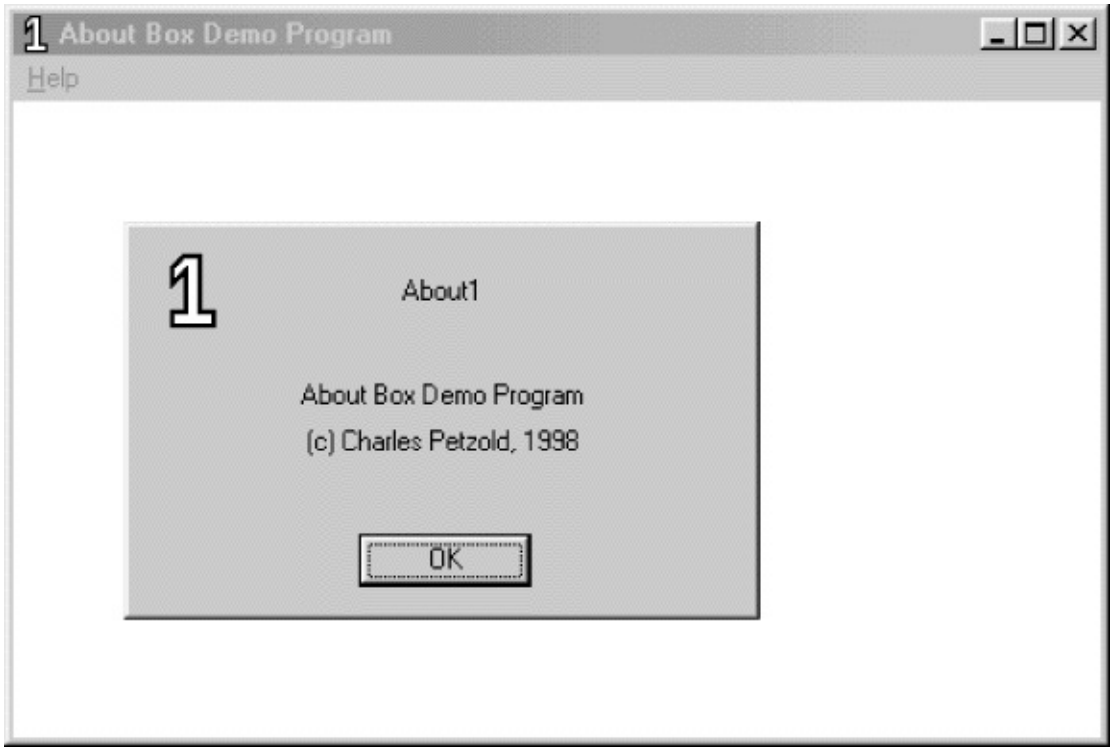
RESOURCE.H

```
// Microsoft Developer Studio generated include file.

// Used by About1.rc

#define IDM_APP_ABOUT       40001

#define IDC_STATIC             -1
```

ABOUT1.ICO



IDAbout1IDIDM_APP_ABOUTWM_COMMAND11-1



11-1
ABOUT1

Visual C++ Developer Studio **Insert** **OKCancel** **Controls**

Developer StudioIDIDD_DIALOG1
AboutBox **X PosY Pos**32

PropertiesStyles **Title Bar** **Properties**

Cancel **Delete** **OK**Developer

Controls Pictures
RESOURCE.H-1ID **TypeIconImage**About1

Controls Static Text **PropertiesProp**

Align TextCenter

ShiftDeveloper

ABOUT1.RCDeveloper

```
ABOUTBOX DIALOG DISCARDABLE  32, 32, 180, 100
STYLE DS_MODALFRAME | WS_POPUP
FONT 8, "MS Sans Serif"
BEGIN
  DEFPUSHBUTTON   "OK",IDOK,66,80,50,14
   ICON                                    "ABOUT1",IDC_STATIC,7,7,2
   CTEXT                                    "About1",IDC_STATIC,40,1
   CTEXT      "About Box Demo Program",IDC_STATIC,7,40,166,8
   CTEXT      "(c) Charles Petzold, 1998",IDC_STATIC,7,52,166,8
END
```

ABOUTBOXDIALOGDISCARDABLExy

8MS

xy

STYLECreateWindowstyleWS_POPUPDS_MODALFRAME

BEGINENDDEFPUSHBUTTONICON
CTEXT

```
control-type "text" id, xPos, yPos, xWidth, yHeight, iStyle
```

iStyleWindows

DEFPUSHBUTTONICONCTEXTCTEXT

```
WS_CHILD | SS_CENTER | WS_VISIBLE | WS_GROUP
```

WS_GROUP COLORS1WS_CHILDSS_CENTERWS_VISIBLE

ABOUT1CreateWindow

idWM_COMMMANDWindowsID
CreateWindowIDIDC_STATICRESOURCE.H-1IDIDOK
WINUSER.H1

1/41/8ICON

DEFPUSHBUTTONDEFPUSHBUTTONWS_GROUPABOUT2
WS_GROUPWS_TABSTOP

WindowsABOUT1

```
BOOL       CALLBACK AboutDlgProc (HWND hDlg, UINT message,

{

        switch (message)

  {

        case   WM_INITDIALOG :

              return TRUE ;
```

```
        case   WM_COMMAND :
             switch (LOWORD (wParam))
             {
             case   IDOK :
             case   IDCANCEL :
                              EndDialog (hDlg, 0) ;
                             return TRUE ;
             }
             break ;
        }
   return FALSE ;
}
```

CALLBACKcallbackhDlghwnd

- LRESULTBOOLWindowsint
- DefWindowProcTRUE0FALSE0
- WM_PAINTWM_DESTROYWM_CREATWM_INITDIALOG

WM_INITDIALOGTRUEWindowsWS_TABSTOPABOUT2
WS_TABSTOPWM_INITDIALOGSetFocusFALSE

WM_COMMANDIDIDOKwParam
EndDialogWindowsFALSEWindows

WndProcWM_CREATEABOUT1

```
hInstance = ((LPCREATESTRUCT) lParam)->hInstance ;
```

ABOUT1WM_COMMANDwParamIDM_APP_ABOUTDialogBox

```
DialogBox (hInstance, TEXT ("AboutBox"), hwnd, AboutDlgProc) ;
```

WM_CREATEMAKEINTRESOURCE

About About111-2OKEnterEnterWindows
WM_COMMANDwParamIDIDIDOKEscapeWindows
WM_COMMANDIDIDCANCEL

DialogBoxWndProcDialogBoxEndDialogABOUT1ABOUT2
WndProcWindows

WndProcWndProcABOUT1AboutDlgProcSendMessage

```
SendMessage (GetParent (hDlg),  . . . ) ;
```

Visual C++ Developer Studio HEXCALC
/Platform SDK/Windows Programming Guidelines/Platform SDK

Tools/Compiling/Using the Resource Compiler

Developer StudioPropertiesSTYLEABOUT1

```
STYLE WS_POPUP | DS_MODALFRAME
```

WS_CAPTIONWS_CAPTIONDIALOGxyy

CAPTIONCAPTIONSTYLE

```
CAPTION "Dialog Box Caption"
```

WM_INITDIALOG

```
SetWindowText (hDlg, TEXT ("Dialog Box Caption")) ;
```

WS_CAPTIONWS_SYSMENU

PropertiesBorder ResizingWS_THICKFRAME

```
MENU menu-name
```

ID

FONTDeveloper

Windows

```
CLASS "class-name"
```

HEXCALC

DialogBoxWindowsCreateWindowWindowsDialogBox
WindowsDialogBoxDialogBox

WindowsDialogBoxIndirect

ABOUT1.RCCTEXTICONDEFPUSHBUTTON

11-1

PUSHBUTTON	BS_PUSHBUTTON | WS_TABSTOP
DEFPUSHBUTTON	BS_DEFPUSHBUTTON | WS_TABSTOP
CHECKBOX	BS_CHECKBOX | WS_TABSTOP
RADIOBUTTON	BS_RADIOBUTTON | WS_TABSTOP
GROUPBOX	BS_GROUPBOX | WS_TABSTOP
LTEXT	SS_LEFT | WS_GROUP
CTEXT	SS_CENTER | WS_GROUP
RTEXT	SS_RIGHT | WS_GROUP

ICON	SS_ICON
EDITTEXT	ES_LEFT | WS_BORDER | WS_TABSTOP
SCROLLBAR	SBS_HORZ
LISTBOX	LBS_NOTIFY | WS_BORDER | WS_VSCROLL
COMBOBOX	CBS_SIMPLE | WS_TABSTOP

```
WS_CHILD | WS_VISIBLE
```

EDITTEXTSCROLLBARLISTBOXCOMBOBOX

```
control-type "text", id, xPos, yPos, xWidth, yHeight, iStyle
```

EDITTEXTSCROLLBARLISTBOXCOMBOBOX

```
control-type id, xPos, yPos, xWidth, yHeight, iStyle
```

iStyle

1/41/8

style

```
CHECKBOX "text", id, xPos, yPos, xWidth, yHeight, BS_LEFTTEXT
```

EDITTEXT

```
EDITTEXT id, xPos, yPos, xWidth, yHeight, NOT WS_BORDER
```

```
CONTROL "text", id, "class", iStyle, xPos, yPos, xWidth, yHeight
```

```
PUSHBUTTON "OK", IDOK, 10, 20, 32, 14
```

```
CONTROL  "OK", IDOK, "button", WS_CHILD | WS_VISIBLE |
                BS_PUSHBUTTON | WS_TABSTOP, 10, 20, 32, 14
```

.RES.EXEDeveloper Stu
ABOUT3

CONTROLWS_CHILDWS_VISIBLEWindowsCONTROL
WindowsDialogBoxCONTROLCreateWindow

```
hCtrl      =CreateWindow (TEXT ("button"), TEXT ("OK"),
                              WS_CHILD | WS_VISIBLE | WS_TABSTO
                                    10 * cxChar / 4, 20 * cyChar /
```

```
                                        32 * cxChar / 4, 14 * cyChar /
                                         hDlg, IDOK, hInstance, NULL
```

cxCharcyCharhDlgCreateWindowhInstanceDialogBox

ABOUT111-2ABOUT2

11-2　ABOUT2

```
ABOUT2.C
/*---------------------------------------------------------------------
  ABOUT2.C --     About Box Demo Program No. 2
                             (c) Charles Petzold, 1998
----------------------------------------------------------------------*/
#include <windows.h>
#include "resource.h"

LRESULT     CALLBACK WndProc                (HWND, UINT, WPAR
BOOL             CALLBACK AboutDlgProc        (HWND, UINT, WPA


int iCurrentColor        = IDC_BLACK,
```

```
  iCurrentFigure       = IDC_RECT ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                  PSTR szCmdLine, int iCmdShow)
{
        static TCHAR        szAppName[] = TEXT ("About2") ;
        MSG                             msg ;
        HWND                            hwnd ;
        WNDCLASS                        wndclass ;

        wndclass.style                          = CS_HREDRAW | C
        wndclass.lpfnWndProc                      = WndProc ;
        wndclass.cbClsExtra                     = 0 ;
        wndclass.cbWndExtra                       = 0 ;
        wndclass.hInstance                      = hInstance ;
        wndclass.hIcon                          = LoadIcon (hInst
        wndclass.hCursor                        = LoadCursor (N
        wndclass.hbrBackground                  = (HBRUSH) GetS
        wndclass.lpszMenuName                   = szAppName ;
```

```
     wndclass.lpszClassName                    = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (  NULL, TEXT ("This program requires W
                              szAppName, MB_ICONERROR) ;
      return 0 ;
}

     hwnd = CreateWindow ( szAppName, TEXT ("About Box D
                         WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
```

```
        {
                TranslateMessage (&msg) ;
                DispatchMessage (&msg) ;
   }
  return msg.wParam ;
}

void PaintWindow (HWND hwnd, int iColor, int iFigure)
{
        static COLORREF crColor[8] = { RGB ( 0, 0, 0), RGB ( 0, 0
                RGB ( 0, 255, 0), RGB ( 0, 255, 255),
                        RGB (255,   0, 0), RGB (255,   0, 255),
                RGB (255, 255, 0), RGB (255, 255, 255)} ;

        HBRUSH                          hBrush ;
        HDC                             hdc ;
        RECT                            rect ;

        hdc = GetDC (hwnd) ;
```

```
        GetClientRect (hwnd, &rect) ;
        hBrush = CreateSolidBrush (crColor[iColor - IDC_BLACK])
        hBrush = (HBRUSH) SelectObject (hdc, hBrush) ;

        if (iFigure == IDC_RECT)
              Rectangle (hdc, rect.left, rect.top, rect.right, rect.bott
        else
         Ellipse   (hdc, rect.left, rect.top, rect.right, rect.bottom) ;
        DeleteObject (SelectObject (hdc, hBrush)) ;
        ReleaseDC (hwnd, hdc) ;
}

void PaintTheBlock (HWND hCtrl, int iColor, int iFigure)
{
        InvalidateRect (hCtrl, NULL, TRUE) ;
        UpdateWindow (hCtrl) ;
        PaintWindow (hCtrl, iColor, iFigure) ;
}
LRESULT CALLBACK WndProc ( HWND hwnd, UINT message, WPA
```

```
{
        static HINSTANCE            hInstance ;
        PAINTSTRUCT                      ps ;

        switch (message)
   {
        case   WM_CREATE:
             hInstance = ((LPCREATESTRUCT) lParam)->hInstance
             return 0 ;

        case   WM_COMMAND:
             switch (LOWORD (wParam))
             {
             case IDM_APP_ABOUT:
                            if (DialogBox (hInstance, TEXT ("About
                                  InvalidateRect (hwnd, NULL, TRU
                            return 0 ;
             }
             break ;
```

```
        case   WM_PAINT:
               BeginPaint (hwnd, &ps) ;
               EndPaint (hwnd, &ps) ;

               PaintWindow (hwnd, iCurrentColor, iCurrentFigure) ;
               return 0 ;

        case   WM_DESTROY:
               PostQuitMessage (0) ;
               return 0 ;
        }
  return DefWindowProc (hwnd, message, wParam, lParam) ;
}

BOOL CALLBACK AboutDlgProc (HWND hDlg, UINT message, WPA
{
        static HWND   hCtrlBlock ;
        static int          iColor, iFigure ;
```

```
switch (message)
{
case   WM_INITDIALOG:
      iColor              = iCurrentColor ;
      iFigure              = iCurrentFigure ;

      CheckRadioButton (hDlg, IDC_BLACK, IDC_WHITE,   iC
      CheckRadioButton (hDlg, IDC_RECT,  IDC_ELLIPSE, iFi

      hCtrlBlock = GetDlgItem (hDlg, IDC_PAINT) ;

     SetFocus (GetDlgItem (hDlg, iColor)) ;
      return FALSE ;

case   WM_COMMAND:
      switch (LOWORD (wParam))
      {
      case   IDOK:
```

```
                    iCurrentColor        = iColor ;
                    iCurrentFigure       = iFigure ;
                    EndDialog (hDlg, TRUE) ;
                    return TRUE ;

          case   IDCANCEL:
                    EndDialog (hDlg, FALSE) ;
                    return TRUE ;

          case   IDC_BLACK:
          case   IDC_RED:
          case   IDC_GREEN:
          case   IDC_YELLOW:
          case   IDC_BLUE:
          case   IDC_MAGENTA:
          case   IDC_CYAN:
          case   IDC_WHITE:
                    iColor = LOWORD (wParam) ;
```

```
                    CheckRadioButton (hDlg, IDC_BLACK, I
                    PaintTheBlock (hCtrlBlock, iColor, iFigu
                    return TRUE ;

          case   IDC_RECT:
          case   IDC_ELLIPSE:
                    iFigure = LOWORD (wParam) ;
                    CheckRadioButton (hDlg, IDC_RECT, ID
                    PaintTheBlock (hCtrlBlock, iColor, iFigu
                    return TRUE ;
          }
          break ;

     case   WM_PAINT:
          PaintTheBlock (hCtrlBlock, iColor, iFigure) ;
        break ;
     }
     return FALSE ;
}
```

ABOUT2.RC

```
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
/////////////////////////////////////////////////////////////////////////////
// Dialog
ABOUTBOX DIALOG DISCARDABLE  32, 32, 200, 234
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION
FONT 8, "MS Sans Serif"
BEGIN
    ICON                                    "ABOUT2",IDC_STATIC,7
    CTEXT         "About2",IDC_STATIC,57,12,86,8
    CTEXT         "About Box Demo Program",IDC_STATIC,7,40,186,
    LTEXT         "",IDC_PAINT,114,67,74,72
    GROUPBOX                          "&Color",IDC_STATIC,7,60,8
    RADIOBUTTON                     "&Black",IDC_BLACK,16,76,64,
    RADIOBUTTON          "B&lue",IDC_BLUE,16,92,64,8
```

```
    RADIOBUTTON         "&Green",IDC_GREEN,16,108,64,8
    RADIOBUTTON         "Cya&n",IDC_CYAN,16,124,64,8
    RADIOBUTTON         "&Red",IDC_RED,16,140,64,8
    RADIOBUTTON         "&Magenta",IDC_MAGENTA,16,156,64,8
    RADIOBUTTON         "&Yellow",IDC_YELLOW,16,172,64,8
    RADIOBUTTON         "&White",IDC_WHITE,16,188,64,8
   GROUPBOX                     "&Figure",IDC_STATIC,109,156,
    RADIOBUTTON                 "Rec&tangle",IDC_RECT,116,17
    RADIOBUTTON                 "&Ellipse",IDC_ELLIPSE,116,18
    DEFPUSHBUTTON                "OK",IDOK,35,212,50,14,WS_
    PUSHBUTTON                      "Cancel",IDCANCEL,113,2
END

////////////////////////////////////////////////////////////////////////
// Icon
ABOUT2        ICON        DISCARDABLE        "About2.ico"

////////////////////////////////////////////////////////////////////////
// Menu
ABOUT2      MENU DISCARDABLE
```

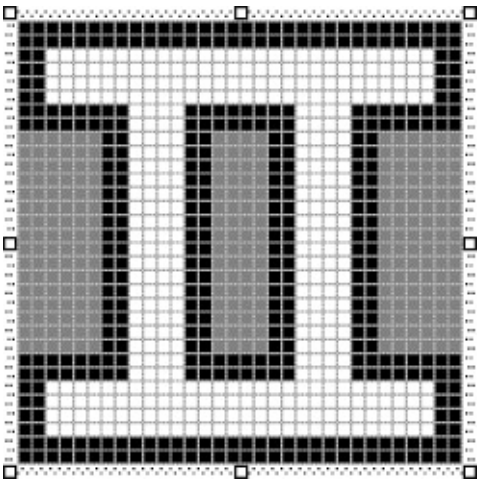
```
BEGIN
  POPUP "&Help"
  BEGIN
            MENUITEM "&About",              IDM_APP_ABOUT
  END
END
```

RESOURCE.H

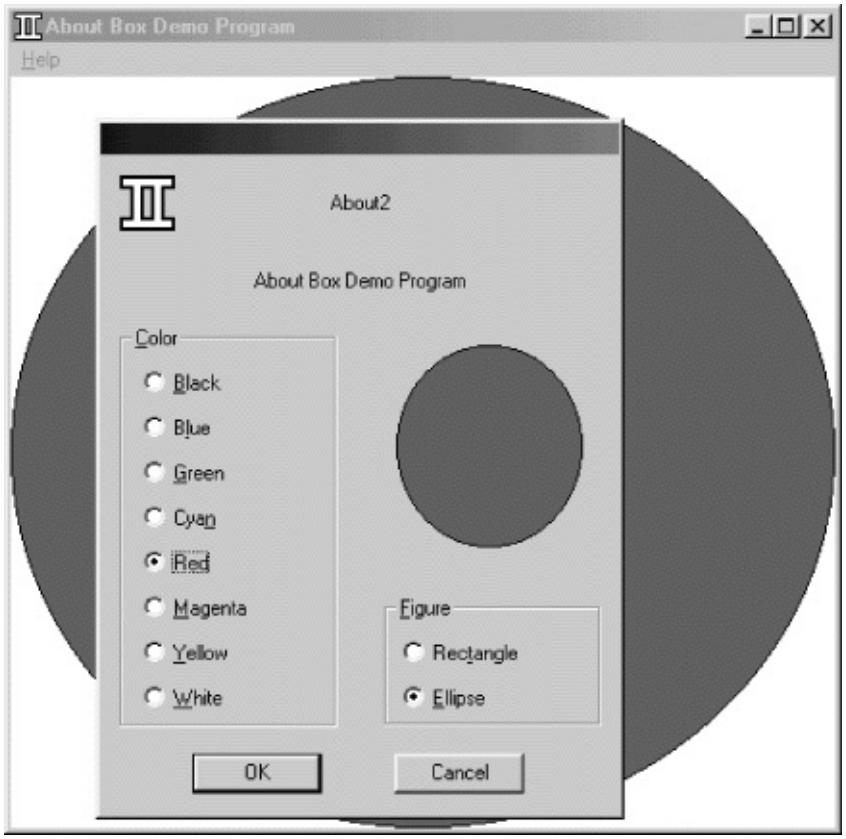
```
// Microsoft Developer Studio generated include file.
// Used by About2.rc
#define IDC_BLACK          1000
#define IDC_BLUE           1001
#define IDC_GREEN          1002
#define IDC_CYAN           1003
#define IDC_RED            1004
#define IDC_MAGENTA        1005
#define IDC_YELLOW         1006
#define IDC_WHITE          1007
```

```
#define IDC_RECT            1008

#define IDC_ELLIPSE         1009

#define IDC_PAINT           1010

#define IDM_APP_ABOUT       40001

#define IDC_STATIC          -1
```

ABOUT2.ICO



ABOUT2AboutOKCancel11-2
ABOUT2IDOKIDCANCELIDCIDRESOURCE.H



11-2 ABOUT2

ABOUT2Developer Studio
ButtonABOUT2.RC

PropertiesGroupOKCancelFigureBlack
RectangleTab Stop

LayoutTab Order

WM_COMMAND
Windows

ABOUT211-2ABOUT2.RCGROUPBOXColorFigure

WM_COMMANDwParamIDwParamlParam
BN_CLICKED0WindowsWM_COMMANDABOUT2.C
WM_COMMAND

BM_CHECK

```
SendMessage (hwndCtrl, BM_SETCHECK, 1, 0) ;
```

```
SendMessage (hwndCtrl, BM_SETCHECK, 0, 0) ;
```

hwndCtrl

WindowsID

```
hwndCtrl = GetDlgItem (hDlg, id) ;
```

ID

```
id = GetWindowLong (hwndCtrl, GWL_ID) ;
```

11-2ABOUT2.HIDIDC_BLACKIDC_WHITEWM_COMMAND

```
static int iColor ;


case       WM_COMMAND:
```

```
        switch (LOWORD (wParam))
        {


        case   IDC_BLACK:
        case   IDC_RED:
        case   IDC_GREEN:
        case   IDC_YELLOW:
        case   IDC_BLUE:
        case   IDC_MAGENTA:
        case   IDC_CYAN:
        case   IDC_WHITE:
             iColor = LOWORD (wParam) ;
             for (i = IDC_BLACK, i <= IDC_WHITE, i++)
                                 SendMessage (GetDlgItem (hDlg, i),
                       BM_SETCHECK, i == LOWORD (wParam), 0) ;
             return TRUE ;
```

iColorIDSendMessageBM_SETCHECK
WM_COMMANDwParam1

SendDlgItemMessage

```
SendDlgItemMessage (hDlg, id, iMsg, wParam, lParam) ;
```

```
SendMessage (GetDlgItem (hDlg, id), id, wParam, lParam) ;
```

```
for (i = IDC_BLACK, i <= IDC_WHITE, i++)
    SendDlgItemMessage (hDlg, i, BM_SETCHECK, i == LWORD (w
```

CheckRadioButton

```
CheckRadioButton (hDlg, idFirst, idLast, idCheck) ;
```

IDidFirstidLastIDidCheckID

```
CheckRadioButton (hDlg, IDC_BLACK, IDC_WHITE, LOWORD (wPa
```

ABOUT2

CHECKBOX

```
CheckDlgButton (hDlg, idCheckbox, iCheck) ;
```

iCheck10

```
iCheck = IsDlgButtonChecked (hDlg, idCheckbox) ;
```

```

```

WM_COMMAND

```
CheckDlgButton (hDlg, idCheckbox,

    !IsDlgButtonChecked (hDlg, idCheckbox)) ;
```

BS_AUTOCHECKBOXWM_COMMANDIsDlgButtonChecked
BS_AUTORADIOBUTTONIsDlgButtonCheckedTRUE
WM_COMMAND

OKCancel

ABOUT2OKCancelABOUT2.RCOKIDIDOK
WINUSER.H1CancelIDIDCANCEL2OK

```
DEFPUSHBUTTON             "OK",IDOK,35,212,50,14

  PUSHBUTTON                 "Cancel",IDCANCEL,113,212,50,14
```

OKCancelOKSpacebarEnter
WM_COMMANDwParamIDwParamIDWindows
WM_COMMANDwParamIDOKEscCtrl-BreakWindowswParam
IDCANCELWM_COMMANDWindowsWM_COMMAND

AboutDlgProcEndDialogWM_COMMAND

```
switch (LWORD (wParam))

{

case IDOK:

    iCurrentColor  = iColor ;
```

```
    iCurrentFigure = iFigure ;
    EndDialog (hDlg, TRUE) ;
    return TRUE ;
case IDCANCEL :
    EndDialog (hDlg, FALSE) ;
    return TRUE ;
```

ABOUT2iCurrentColoriCurrentFigureAboutDlgProciColoriFigure

EndDialogWndProcDialogBox

```
case        IDM_ABOUT:
          if (DialogBox (hInstance, TEXT ("AboutBox"), hwnd, About
               InvalidateRect (hwnd, NULL, TRUE) ;
          return 0 ;
```

DialogBoxTRUE0OKWndProcAboutDlgProc
WM_COMMANDwParamIDOKAboutDlgProciCurrentColoriCurrentFigure
DialogBoxFALSEiCurrentColoriCurrentFigure

TRUEFALSEEndDialogOKCancelEndDialogint
DialogBoxintTRUEFALSE

ABOUT2ABOUT2iCurrentColoriCurrentFigure

typedefABOUT2About

```
typedef struct
{
        int iColor, iFigure ;
}
ABOUTBOX_DATA ;
```

WndProc

```
static ABOUTBOX_DATA ad = { IDC_BLACK, IDC_RECT } ;
```

WndProcad.iColorad.iFigureiCurrentColoriCurrentFigure
DialogBoxParamDialogBox32WndProcABOUTBOX_DATA

```
case       IDM_ABOUT:
        if (DialogBoxParam (hInstance, TEXT ("AboutBox"),
                  hwnd, AboutDlgProc, &ad))
             InvalidateRect (hwnd, NULL, TRUE) ;
        return 0 ;
```

DialogBoxParamWM_INITDIALOGlParam

ABOUTBOX_DATA

```
static ABOUTBOX_DATA ad, * pad ;
```

```

```

AboutDlgProciColoriFigureWM_INITDIALOGlParam

```
pad = (ABOUTBOX_DATA *) lParam ;

ad = * pad ;
```

padlParampadWndProcABOUTBOX_DATAWndProc
DlgProc

OKAboutDlgProcad.iColorad.iFigureiFigureiColorWndProc

```
case      IDOK:

        * pad = ad ;

        EndDialog (hDlg, TRUE) ;

        return TRUE ;
```

Tab

COLORS1TabWindowsWS_TABSTOPWS_GROUP
TabWS_TABSTOP

11-1WS_TABSTOPWS_TABSTOP
WM_INITDIALOGFALSEWindowsWS_TABSTOP

WindowsTabWS_GROUPWindowsWS_GROUP
WS_GROUPWindows

LTEXTCTEXTRTEXTICONWS_GROUPWS_GROUP

ABOUT2.RCWS_TABSTOPTab

ColorBlac

Figure RectangleDEFPUSHBUTTONWS_GROUP **RectangleEllipse** WS_GROUP

ABOUT2WindowsWM_COMMANDWindows WS_TABSTOPTabWindows

C

WindowsTab

```
hwndCtrl = GetNextDlgTabItem (hDlg, hwndCtrl, bPrevious) ;
```

```
hwndCtrl = GetNextDlgGroupItem (hDlg, hwndCtrl, bPrevious) ;
```

bPreviousTRUETabFALSETab

ABOUT2ABOUT2.RC

```
LTEXT  ""  IDC_PAINT, 114, 67, 72, 72
```

189

WM_PAINTPaintTheBlockABOUT2.C

```
PaintTheBlock (hCtrlBlock, iColor, iFigure) ;
```

AboutDlgProchCtrlBlockWM_INITDIALOG

```
hCtrlBlock = GetDlgItem (hDlg, IDD_PAINT) ;
```

```

```

PaintTheBlock

```
void PaintTheBlock (HWND hCtrl, int iColor, int iFigure)
{
        InvalidateRect (hCtrl, NULL, TRUE) ;
        UpdateWindow (hCtrl) ;
        PaintWindow (hCtrl, iColor, iFigure) ;
}
```

WM_PAINTABOUT2PaintWindow

PaintWindowhCtrlGetClientRectGetClientRect
MapDialogRect

WM_PAINT

MoveWindow

```
EnableWindow (hwndCtrl, bEnable) ;
```

bEnableTRUE0bEnableFALSE0

Windows

11-3 ABOUT3

```
ABOUT3.C

/*---------------------------------------------------------------------------

  ABOUT3.C -- About Box Demo Program No. 3

                                        (c) Charles Petzold, 1998

---------------------------------------------------------------------------*/

#include    <windows.h>

#include    "resource.h"

LRESULT     CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM

BOOL            CALLBACK AboutDlgProc (HWND, UINT, WPARAM,

LRESULT     CALLBACK EllipPushWndProc (HWND, UINT, WPARAM


int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns

                PSTR szCmdLine, int iCmdShow)

{

        static TCHAR szAppName[] = TEXT ("About3") ;

        MSG                             msg ;
```

```
     HWND                    hwnd ;
     WNDCLASS                wndclass ;

     wndclass.style                      = CS_HREDRAW |
     wndclass.lpfnWndProc                  = WndProc ;
     wndclass.cbClsExtra                 = 0 ;
     wndclass.cbWndExtra                   = 0 ;
     wndclass.hInstance                  = hInstance ;
     wndclass.hIcon                      = LoadIcon (hInst
     wndclass.hCursor                    = LoadCursor (N
     wndclass.hbrBackground              = (HBRUSH) GetS
     wndclass.lpszMenuName               = szAppName ;
     wndclass.lpszClassName              = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (  NULL, TEXT ("This program requires W
                                          szAppName, MB_IC
          return 0 ;
```

```
}

        wndclass.style                  = CS_HREDRAW | CS_V
        wndclass.lpfnWndProc              = EllipPushWndPro
        wndclass.cbClsExtra               = 0 ;
        wndclass.cbWndExtra               = 0 ;
        wndclass.hInstance              = hInstance ;
        wndclass.hIcon                  = NULL ;
        wndclass.hCursor                = LoadCursor (NULL,
        wndclass.hbrBackground          = (HBRUSH) (COLOR_B
        wndclass.lpszMenuName           = NULL ;
        wndclass.lpszClassName          = TEXT ("EllipPush") ;

        RegisterClass (&wndclass) ;
        hwnd = CreateWindow ( szAppName, TEXT ("About Box D
                    WS_OVERLAPPEDWINDOW,
                    CW_USEDEFAULT, CW_USEDEFAULT,
                    CW_USEDEFAULT, CW_USEDEFAULT,
                   NULL, NULL, hInstance, NULL) ;
```

```
        ShowWindow (hwnd, iCmdShow) ;
        UpdateWindow (hwnd) ;

        while (GetMessage (&msg, NULL, 0, 0))
        {
             TranslateMessage (&msg) ;
             DispatchMessage (&msg) ;
  }
   return msg.wParam ;
}

LRESULT CALLBACK WndProc ( HWND hwnd, UINT message, WPA
{
        static HINSTANCE hInstance ;
        switch (message)
        {
        case   WM_CREATE :
             hInstance = ((LPCREATESTRUCT) lParam)->hInstance
```

```
            return 0 ;

 case WM_COMMAND :
            switch (LOWORD (wParam))
      {
            case   IDM_APP_ABOUT :
                        DialogBox (hInstance, TEXT ("AboutBo
                        return 0 ;
           }
            break ;

        case   WM_DESTROY :
            PostQuitMessage (0) ;
            return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam)
}

BOOL CALLBACK AboutDlgProc (HWND hDlg, UINT message, WPA
```

```
{
        switch (message)
        {
        case   WM_INITDIALOG :
              return TRUE ;


        case   WM_COMMAND :
              switch (LOWORD (wParam))
              {
              case   IDOK :
                                  EndDialog (hDlg, 0) ;
                                  return TRUE ;
             }
              break ;
   }
  return FALSE ;
}

LRESULT CALLBACK EllipPushWndProc (HWND hwnd, UINT messa
```

```
{
        TCHAR                   szText[40] ;
        HBRUSH                  hBrush ;
        HDC                     hdc ;
        PAINTSTRUCT             ps ;
        RECT                            rect ;

        switch (message)
        {
        case   WM_PAINT :
              GetClientRect (hwnd, &rect) ;
              GetWindowText (hwnd, szText, sizeof (szText)) ;

              hdc = BeginPaint (hwnd, &ps) ;

              hBrush = CreateSolidBrush (GetSysColor (COLOR_WI
              hBrush = (HBRUSH) SelectObject (hdc, hBrush) ;
              SetBkColor (hdc, GetSysColor (COLOR_WINDOW)) ;
              SetTextColor (hdc, GetSysColor (COLOR_WINDOWTEX
```

```
            Ellipse (hdc, rect.left, rect.top, rect.right, rect.bottom
            DrawText (hdc, szText, -1, &rect,
          DT_SINGLELINE | DT_CENTER | DT_VCENTER) ;

            DeleteObject (SelectObject (hdc, hBrush)) ;

            EndPaint (hwnd, &ps) ;
            return 0 ;

      case   WM_KEYUP :
            if (wParam != VK_SPACE)
                        break ;// fall through
      case   WM_LBUTTONUP :
            SendMessage (GetParent (hwnd), WM_COMMAND,
                  GetWindowLong (hwnd, GWL_ID), (LPARAM
            return 0 ;
}
      return DefWindowProc (hwnd, message, wParam, lParam)
```

```
}
```

ABOUT3.RC

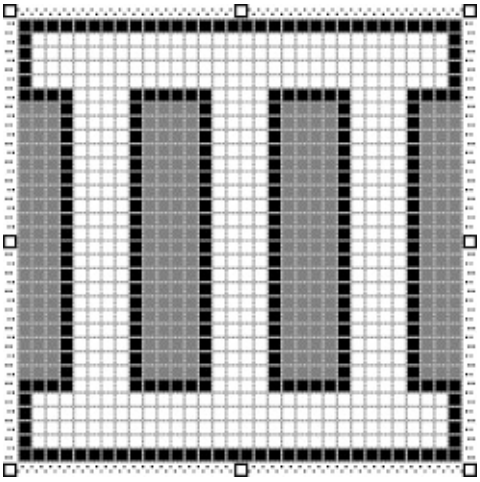
```
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
//////////////////////////////////////////////////////////////////////////////
// Dialog
ABOUTBOX DIALOG DISCARDABLE  32, 32, 180, 100
STYLE DS_MODALFRAME | WS_POPUP
FONT 8, "MS Sans Serif"
BEGIN
    CONTROL                 "OK",IDOK,"EllipPush",WS_GROUP | WS
    ICON       "ABOUT3",IDC_STATIC,7,7,20,20
    CTEXT      "About3",IDC_STATIC,40,12,100,8
    CTEXT      "About Box Demo Program",IDC_STATIC,7,40,166,8
    CTEXT                   "(c) Charles Petzold, 1998",IDC_STA
END
```

```
//////////////////////////////////////////////////////////////////////////////
// Menu
ABOUT3 MENU DISCARDABLE
BEGIN
   POPUP "&Help"
   BEGIN
                MENUITEM "&About About3...",
   END
END

//////////////////////////////////////////////////////////////////////////////
// Icon
ABOUT3      ICON    DISCARDABLE    "icon1.ico"
```

RESOURCE.H

```
// Microsoft Developer Studio generated include file.
// Used by About3.rc
#define IDM_APP_ABOUT          40001
#define IDC_STATIC             -1
```

ABOUT3.ICO



EllipPushDeveloper StudioCancelOI
Custom Control Properties Class EllipPush DEFPUSHBUTTON
CONTROL

```
CONTROL "OK" IDOK, "EllipPush", TABGRP, 64, 60, 32, 14
```

CreateWindow

ABOUT3.CWinMainEllipPush

```
wndclass.style                  = CS_HREDRAW | CS_VREDRAW ;

wndclass.lpfnWndProc            = EllipPushWndProc ;

wndclass.cbClsExtra       = 0 ;
```

```
wndclass.cbWndExtra       = 0 ;
wndclass.hInstance          = hInstance ;
wndclass.hIcon              = NULL ;
wndclass.hCursor            = LoadCursor (NULL, IDC_ARROW)
wndclass.hbrBackground          = (HBRUSH) (COLOR_WINDOW
wndclass.lpszMenuName           = NULL ;
wndclass.lpszClassName          = TEXT ("EllipPush") ;
RegisterClass (&wndclass) ;
```

EllipPushWndProcABOUT3.C

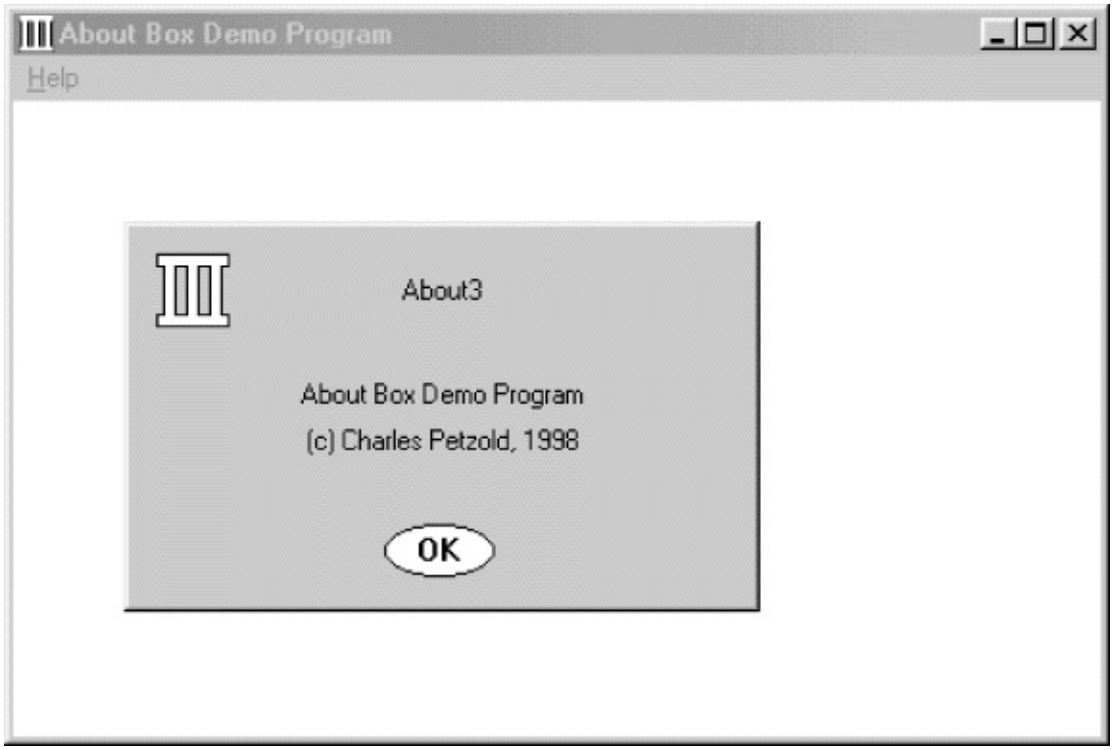
EllipPushWndProcWM_PAINTWM_KEYUPWM_LBUTTONUP
WM_PAINTGetClientRectGetWindowTextWindowsEllipse
DrawText

WM_KEYUPWM_LBUTTONUP

```
case      WM_KEYUP :
          if (wParam != VK_SPACE)
               break ;     // fall through
case WM_LBUTTONUP :
          SendMessage (GetParent (hwnd), WM_COMMAND,
               GetWindowLong (hwnd, GWL_ID), (LPARAM) hwnd) ;
```

```
        return 0 ;
```

GetParentWM_COMMANDwParamIDIDGetWindowLong
ABOUT311-3



11-3 ABOUT3

EllipPushWndProcWM_KEYDOWN
WM_LBUTTONDOWNWM_LBUTTONDOWN
WM_COMMAND

EllipPushWndProcWM_ENABLEEnableWindow

cbWndExtraSetWindowLongGetWindowLong

FindChangeFindFind

DialogBoxEndDialogDialogBoxCreateDialog
DialogBox

```
hDlgModeless = CreateDialog (      hInstance, szTemplate,
                                   hwndParent, DialogProc) ;
```

CreateDialog

DialogBoxCreateDialogCreateDialogCreateWindow

Developer StudioST

```
STYLE WS_POPUP | WS_CAPTION | WS_SYSMENU | WS_VISIBLE
```

STYLEWS_VISIBLE **D**
More StylesWS_VISIBLECreateDialogShowWindow

```
hDlgModeless = CreateDialog (  . . .  ) ;
   ShowWindow (hDlgModeless, SW_SHOW) ;
```

WS_VISIBLEShowWindow

CreateDialoghDlgModeless

```
while (GetMessage (&msg, NULL, 0, 0))
{
        if (hDlgModeless == 0 || !IsDialogMessage (hDlgModeless
   {
             TranslateMessage (&msg) ;
            DispatchMessage  (&msg) ;
   }
}
```

IsDialogMessageTRUE0FALSE0hDlgModeless0
TranslateMessageDispatchMessage

```
while (GetMessage (&msg, NULL, 0, 0))
{
        if (hDlgModeless == 0 || !IsDialogMessage (hDlgModeless
        {
             if (!TranslateAccelerator (hwnd, hAccel, &msg))
            {
                              TranslateMessage (&msg) ;
                              DispatchMessage  (&msg) ;
```

```
            }
    }
}
```

0hDlgModeless0IsDialogMessage

hDlgModelesshDlgModeless0

DestroyWindowEndDialogDestroyWindowhDlgModeless0

CloseCloseWindowsWM_CLOSE

```
case      WM_CLOSE :
          DestroyWindow (hDlg) ;
          hDlgModeless = NULL ;
          break ;
```

DestroyWindowhDlghDlgModelessCreateDialog

WM_CLOSECreateDialogParam

COLORS

COLORS1COLORS1WndProcCOLORS211-4

11-4　　COLORS2

```
COLORS2.C
```

```
/*------------------------------------------------------------------------
  COLORS2.C -- Version using Modeless Dialog Box
                                        (c) Charles Petzold, 1998
------------------------------------------------------------------------*/
#include <windows.h>
LRESULT     CALLBACK WndProc            (HWND, UINT, WPARAM,
BOOL             CALLBACK ColorScrDlg         (HWND, UINT, WPAR
HWND hDlgModeless ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                    PSTR szCmdLine, int iCmdShow)
{
          static TCHAR         szAppName[] = TEXT ("Colors2") ;
          HWND                                 hwnd ;
          MSG                                 msg ;
          WNDCLASS                           wndclass ;

          wndclass.style                        = CS_HREDRAW | CS_V
          wndclass.lpfnWndProc                     = WndProc ;
          wndclass.cbClsExtra                      = 0 ;
```

```
     wndclass.cbWndExtra                    = 0 ;
     wndclass.hInstance                     = hInstance ;
     wndclass.hIcon                         = LoadIcon (NULL, IDI_
     wndclass.hCursor                       = LoadCursor (NULL, I
     wndclass.hbrBackground                 = CreateSolidBrush (0L
     wndclass.lpszMenuName                  = NULL ;
     wndclass.lpszClassName                 = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (  NULL, TEXT ("This program requires W
                                             szAppName, MB_IC
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, TEXT ("Color Scroll")
               WS_OVERLAPPEDWINDOW | WS_CLIPCHILDREN,
               CW_USEDEFAULT, CW_USEDEFAULT,
```

```
                    CW_USEDEFAULT, CW_USEDEFAULT,
                    NULL, NULL, hInstance, NULL) ;

          ShowWindow (hwnd, iCmdShow) ;
          UpdateWindow (hwnd) ;

          hDlgModeless = CreateDialog (hInstance, TEXT ("ColorSc
            hwnd, ColorScrDlg) ;
          while (GetMessage (&msg, NULL, 0, 0))
          {
               if (hDlgModeless == 0 || !IsDialogMessage (hDlgMod
               {
                         TranslateMessage (&msg) ;
                         DispatchMessage  (&msg) ;
               }
          }
          return msg.wParam ;
}
```

```
LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
         switch (message)
         {
        case   WM_DESTROY :
               DeleteObject ((HGDIOBJ) SetClassLong (hwnd, GCL_H
            (LONG) GetStockObject (WHITE_BRUSH))) ;
               PostQuitMessage (0) ;
               return 0 ;
         }
         return DefWindowProc (hwnd, message, wParam, lParam)
}

BOOL CALLBACK ColorScrDlg (HWND hDlg, UINT message,WPAR
{
         static int          iColor[3] ;
         HWND                        hwndParent, hCtrl ;
         int                       iCtrlID, iIndex ;
```

```
        switch (message)
        {
        case   WM_INITDIALOG :
             for (iCtrlID = 10 ; iCtrlID < 13 ; iCtrlID++)
             {
                              hCtrl = GetDlgItem (hDlg, iCtrlID) ;
                              SetScrollRange (hCtrl, SB_CTL, 0, 255,
                              SetScrollPos  (hCtrl, SB_CTL, 0, FALSE)
             }
             return TRUE ;

        case   WM_VSCROLL :
             hCtrl               = (HWND) lParam ;
             iCtrlID             = GetWindowLong (hCtrl, GWL_ID) ;
             iIndex              = iCtrlID - 10 ;
             hwndParent          = GetParent (hDlg) ;

             switch (LOWORD (wParam))
                  {
```

```
            case   SB_PAGEDOWN :
                        iColor[iIndex] += 15 ;        // fall through
                    case SB_LINEDOWN :
                        iColor[iIndex] = min (255, iColor[iIndex
                        break ;
            case   SB_PAGEUP :
                        iColor[iIndex] -= 15 ;     // fall through
                    case SB_LINEUP :
                        iColor[iIndex] = max (0, iColor[iIndex] -
                        break ;
            case   SB_TOP :
                        iColor[iIndex] = 0 ;
                        break ;
            case   SB_BOTTOM :
                        iColor[iIndex] = 255 ;
                        break ;
                 case   SB_THUMBPOSITION :
                case   SB_THUMBTRACK :
```

```
                                iColor[iIndex] = HIWORD (wParam
                                break ;
                        default :
                                return FALSE ;
                }
                        SetScrollPos  (hCtrl, SB_CTL,       iColor[iIndex], T
                        SetDlgItemInt (hDlg,  iCtrlID + 3, iColor[iIndex],

                        DeleteObject ((HGDIOBJ) SetClassLong (hwndPa
                                (LONG) CreateSolidBrush (
                                RGB (iColor[0], iColor[1], iColor[2])))) ;

                InvalidateRect (hwndParent, NULL, TRUE) ;
                return TRUE ;
        }
                return FALSE ;
}
```

COLORS2.RC

```
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
/////////////////////////////////////////////////////////////////////////////
// Dialog
COLORSCRDLG DIALOG DISCARDABLE  16, 16, 120, 141
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTIO
CAPTION "Color Scroll Scrollbars"
FONT 8, "MS Sans Serif"
BEGIN
    CTEXT                       "&Red",IDC_STATIC,8,8,24
    SCROLLBAR                   10,8,20,24,100,SBS_VERT | WS_T
    CTEXT                   "0",13,8,124,24,8,NOT WS_GROUP
    CTEXT                       "&Green",IDC_STATIC,48,8,2
    SCROLLBAR                   11,48,20,24,100,SBS_VERT | WS_
    CTEXT                   "0",14,48,124,24,8,NOT WS_GROUP
    CTEXT                       "&Blue",IDC_STATIC,89,8,
    SCROLLBAR                   12,89,20,24,100,SBS_VERT | WS_
```

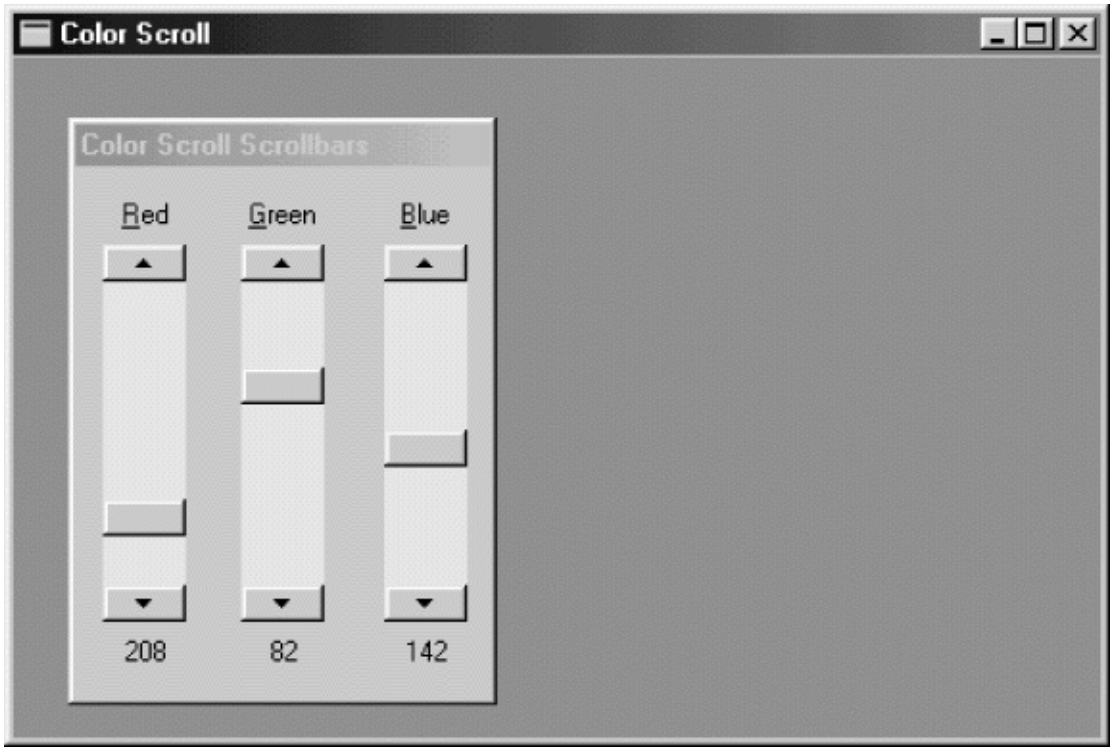
```
    CTEXT                          "0",15,89,124,24,8,NOT WS_GROUP
END
```

RESOURCE.H

```
// Microsoft Developer Studio generated include file.
// Used by Colors2.rc
#define IDC_STATIC      -1
```

COLORS111-4

ID101112ID131415Tab



11-4 COLORS2

COLORS2WinMainShowWindowWS_CLIPCHILDREN

CreateDialoghDlgModelessIsDialogMessage

```
while      (GetMessage (&msg, NULL, 0, 0))
{
        if (!IsDialogMessage (hDlgModeless, &msg))
   {
              TranslateMessage      (&msg) ;
              DispatchMessage       (&msg) ;
```

```
   }
}
```

hDlgModeless

```
case      WM_CLOSE :
          DestroyWindow (hDlg) ;
          hDlgModeless = NULL ;
          break ;
```

COLORS1SetWindowTextwsprintf

```
wsprintf (szBuffer, TEXT ("%i"), color[i]) ;
SetWindowText (hwndValue[i], szBuffer) ;
```

iIDhwndValue

SetDlgItemInt

```
SetDlgItemInt (hDlg, iCtrlID + 3, color [iCtrlID], FALSE) ;
```

SetDlgItemIntGetDlgItemIntiCtrlIDIDID3IDTRUE
FALSE0256

COLORS1COLORS2WindowsCreateWindow
CreateDialogCreateWindow

HEXCALC

HEXCALC11-5CreateWindowWM_PAINT15010
11-5

11-5 HEXCALC

```
HEXCALC.C

/*------------------------------------------------------------------------
  HEXCALC.C -- Hexadecimal Calculator
                          (c) Charles Petzold, 1998
------------------------------------------------------------------------*/

#include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                         PSTR szCmdLine, int iCmdSh
{
     static TCHAR szAppName[] = TEXT ("HexCalc") ;
     HWND         hwnd ;
     MSG          msg ;
     WNDCLASS     wndclass ;

     wndclass.style                    = CS_HREDRAW | CS_V
```

```
     wndclass.lpfnWndProc                  = WndProc ;
     wndclass.cbClsExtra                   = 0 ;
     wndclass.cbWndExtra                   = DLGWINDOWEXT
     wndclass.hInstance                    = hInstance ;
     wndclass.hIcon                        = LoadIcon (hInstance
     wndclass.hCursor                      = LoadCursor (NULL,
     wndclass.hbrBackground                = (HBRUSH) (COLOR_B
     wndclass.lpszMenuName                 = NULL ;
     wndclass.lpszClassName                = szAppName ;

     if (!RegisterClass (&wndclass))
{
          MessageBox (  NULL, TEXT ("This program requires W
                                          szAppName, MB_IC
          return 0 ;
     }

     hwnd = CreateDialog (hInstance, szAppName, 0, NULL) ;
```

```
        ShowWindow (hwnd, iCmdShow) ;
        while (GetMessage (&msg, NULL, 0, 0))
        {
                TranslateMessage (&msg) ;
                DispatchMessage (&msg) ;
   }
        return msg.wParam ;
}

void ShowNumber (HWND hwnd, UINT iNumber)
{
        TCHAR szBuffer[20] ;
        wsprintf (szBuffer, TEXT ("%X"), iNumber) ;
        SetDlgItemText (hwnd, VK_ESCAPE, szBuffer) ;
}

DWORD CalcIt (UINT iFirstNum, int iOperation, UINT iNum)
{
   switch (iOperation)
        {
```

```
        case '=': return iNum ;
        case '+': return iFirstNum +  iNum ;
        case '-': return iFirstNum -  iNum ;
        case '*': return iFirstNum *  iNum ;
        case '&': return iFirstNum &  iNum ;
        case '|': return iFirstNum |  iNum ;
        case '^': return iFirstNum ^  iNum ;
        case '<': return iFirstNum << iNum ;
        case '>': return iFirstNum >> iNum ;
        case '/': return iNum ? iFirstNum / iNum: MAXDWORD ;
        case '%': return iNum ? iFirstNum % iNum: MAXDWORD ;
        default : return 0 ;
        }
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
        static BOOL  bNewNumber = TRUE ;
        static int   iOperation = '=' ;
```

```
        static UINT   iNumber, iFirstNum ;
        HWND                            hButton ;

        switch (message)
        {
        case WM_KEYDOWN:                    // left arrow --> backspa
              if (wParam != VK_LEFT)
                                    break ;
              wParam = VK_BACK ;
        // fall through
case   WM_CHAR:
              if    ((wParam = (WPARAM) CharUpper ((TCHAR *) wP
                                    wParam = '=' ;

              if    (hButton = GetDlgItem (hwnd, wParam))
              {
                          SendMessage (hButton, BM_SETSTATE, 1, 0
                          Sleep (100) ;
```

```
                    SendMessage (hButton, BM_SETSTATE, 0, 0
          }
          else
          {
                    MessageBeep (0) ;
                     break ;
          }
        // fall through
     case   WM_COMMAND:
          SetFocus (hwnd) ;

          if (LOWORD (wParam) == VK_BACK)              //back
                         ShowNumber (hwnd, iNumber /= 16) ;

          else if (LOWORD (wParam) == VK_ESCAPE)
                         ShowNumber (hwnd, iNumber = 0) ;

          else if (isxdigit (LOWORD (wParam)))              // hex
     {
```

```
                    if (bNewNumber)
                    {
                        iFirstNum = iNumber ;
                        iNumber = 0 ;
                    }
                    bNewNumber = FALSE ;
                if  (iNumber <= MAXDWORD >> 4)
                    ShowNumber (hwnd, iNumber = 16 * iN
                    (isdigit (wParam) ? '0': 'A' - 10)) ;
                else
                    MessageBeep (0) ;
            }
            else    // operation
        {
                    if (!bNewNumber)
            ShowNumber (hwnd, iNumber =
                CalcIt (iFirstNum, iOperation, iNumber)) ;
                    bNewNumber = TRUE ;
```

```
                        iOperation = LOWORD (wParam) ;
                }
                return 0 ;
        case   WM_DESTROY:
                PostQuitMessage (0) ;
                return 0 ;
  }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

HEXCALC.RC

```
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
//////////////////////////////////////////////////////////////////////////////
// Icon
HEXCALC                         ICON    DISCARDABLE

//////////////////////////////////////////////////////////////////////////////
```

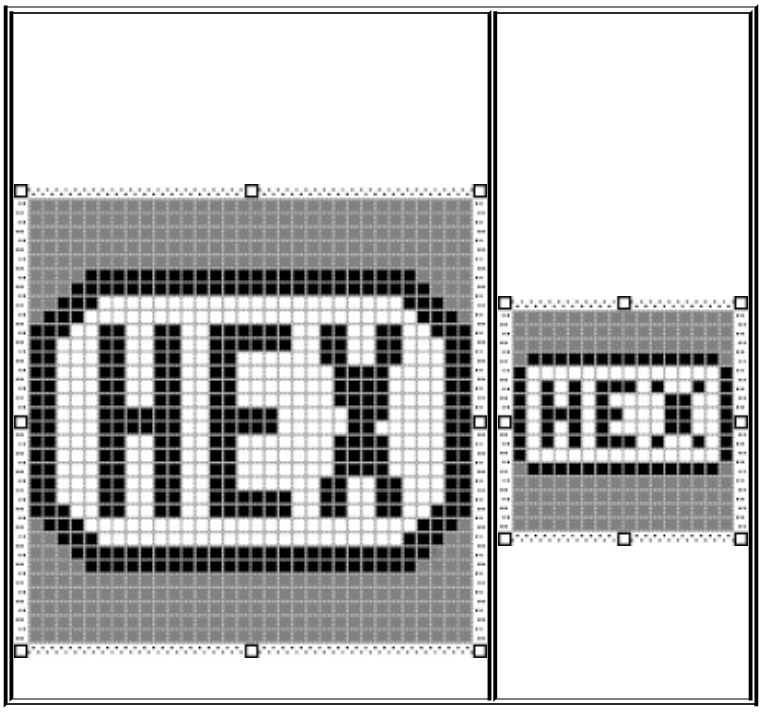
```
#include "hexcalc.dlg"
```

```
HEXCALC.DLG

/*--------------------------------

  HEXCALC.DLG dialog script

---------------------------------*/

HexCalc DIALOG -1, -1, 102, 122

STYLE WS_OVERLAPPED | WS_CAPTION | WS_SYSMENU | WS_MIN

CLASS "HexCalc"

CAPTION "Hex Calculator"

{

   PUSHBUTTON "D",        68,  8,  24, 14, 14

         PUSHBUTTON "A",        65,  8,  40, 14, 14

     PUSHBUTTON "7",        55,  8,  56, 14, 14

         PUSHBUTTON "4",     52,  8,  72, 14, 14

         PUSHBUTTON "1",            49,  8,  88, 14, 14

         PUSHBUTTON "0",        48,  8,  104,14, 14

         PUSHBUTTON "0",      27,  26, 4,  50, 14

         PUSHBUTTON "E",      69,  26, 24, 14, 14
```

```
 PUSHBUTTON "B",       66,  26, 40, 14, 14
PUSHBUTTON "8",             56,  26, 56, 14, 14
 PUSHBUTTON "5",       53,  26, 72, 14, 14
 PUSHBUTTON "2",       50,  26, 88, 14, 14
 PUSHBUTTON "Back",    8,   26, 104,32, 14
 PUSHBUTTON "C",       67,  44, 40, 14, 14
 PUSHBUTTON "F",       70,  44, 24, 14, 14
 PUSHBUTTON "9",        57,  44, 56, 14, 14
 PUSHBUTTON "6",        54,  44, 72, 14, 14
 PUSHBUTTON "3",        51,  44, 88, 14, 14
 PUSHBUTTON "+",        43,  62, 24, 14, 14
 PUSHBUTTON "-",        45,  62, 40, 14, 14
 PUSHBUTTON "*",        42,  62, 56, 14, 14
 PUSHBUTTON "/",        47,  62, 72, 14, 14
 PUSHBUTTON "%",        37,  62, 88, 14, 14
 PUSHBUTTON "Equals",    61,  62, 104,32, 14
 PUSHBUTTON "&&",38,  80, 24, 14, 14
 PUSHBUTTON "|",      124, 80, 40, 14, 14
```

```
        PUSHBUTTON "^",        94,  80, 56, 14, 14
        PUSHBUTTON "<",      60,  80, 72, 14, 14
        PUSHBUTTON ">",      62,  80, 88, 14, 14
}
```

HEXCALC.ICO



11-5 HEXCALC

HEXCALCC32AND,
FFFFFFFF

HEXCALC8EqualsEnterBack
BackspacedisplayEsc

HEXCALCHEXCALCWndProcWM_DESTROY
DefWindowProcWinMainCreateDialogHEXCALC.DLG
HEXCALC

WindowsWindowsHEXCALCWindows

Developer StudioDialog EditorHEXCALC.DLG
FileNew Files Text File ViewResource Includes
Compile-time Directives

```
#include "hexcalc.dlg"
```

HEXCALC.RC

HEXCALC.DLGHEXCALC

```
HexCalc DIALOG -1, -1, 102, 122

STYLE WS_OVERLAPPED | WS_CAPTION | WS_SYSMENU | WS_MI

CLASS "HexCalc"

CAPTION "Hex Calculator"
```

WS_OVERLAPPEDWS_MINIMIZEBOXCreateWindowCLASS Developer StudioDialog EditorWindowsCLASSWindows HexCalc

HexCalcHEXCALCWinMainWNDCLASScbWndExtra DLGWINDOWEXTRA

WinMainCreateDialog

```
hwnd = CreateDialog (hInstance, szAppName, 0, NULL) ;
```

HexCaEc0Windows

CreateDialogWindowsCreateWindowCreateWindow

```
hwnd =      CreateWindow (TEXT ("HexCalc"), TEXT ("Hex Calcula

            WS_OVERLAPPED | WS_CAPTION | WS_SYSMENU | WS

            CW_USEDEFAULT, CW_USEDEFAULT,

            102 * 4 / cxChar, 122 * 8 / cyChar,

            NULL, NULL, hInstance, NULL) ;
```

cxCharcyChar

WindowsCreateWindowWindows129CreateWindow
WM_COMMANDWndProc

HEXCALCHEXCALCIDASCIIWndProc
WM_COMMANDWM_CHARwParamASCII

WndProcWM_KEYDOWNBackspaceWM_CHARWndProc
EnterASCII

WM_CHARGetDlgItemGetDlgItem0IDIDBM_SETSTATE

```
if (hButton = GetDlgItem (hwnd, wParam))

{

        SendMessage (hButton, BM_SETSTATE, 1, 0) ;

        Sleep (100) ;

        SendMessage (hButton, BM_SETSTATE, 0, 0) ;

}
```

HEXCALCSleep100

WndProcWM_COMMAND

```
case      WM_COMMAND :

        SetFocus (hwnd) ;
```

WindowsAlt-File-Open

Windows　3.1

CCOMMDLG.H/Platform
Input/Common Dialog Box Library

POPPAD

POPPADPOPPADPOPPAD

POPPAD311-6

11-6　　POPPAD3

```
POPPAD.C

/*------------------------------------------------------------------------
  POPPAD.C -- Popup Editor
                                        (c) Charles Petzold, 1998
------------------------------------------------------------------------*/

#include <windows.h>

#include <commdlg.h>

#include "resource.h"


#define     EDITID   1
```

```
#define     UNTITLED TEXT ("(untitled)")

LRESULT     CALLBACK WndProc      (HWND, UINT, WPARAM, LPAR
BOOL             CALLBACK AboutDlgProc (HWND, UINT, WPARAM,

                // Functions in POPFILE.C

void        PopFileInitialize                (HWND) ;
BOOL        PopFileOpenDlg                    (HWND, PTSTR, PTSTR)
BOOL        PopFileSaveDlg                    (HWND, PTSTR, PTSTR)
BOOL        PopFileRead                      (HWND, PTSTR) ;
BOOL        PopFileWrite                     (HWND, PTSTR) ;

                // Functions in POPFIND.C

HWND        PopFindFindDlg                    (HWND) ;
HWND        PopFindReplaceDlg                 (HWND) ;
BOOL        PopFindFindText                  (HWND, int *, LPFINDRE
BOOL        PopFindReplaceText                (HWND, int *, LPFINDR
BOOL        PopFindNextText                   (HWND, int *) ;
BOOL        PopFindValidFind                 (void) ;
```

```
                // Functions in POPFONT.C

void        PopFontInitialize            (HWND) ;
BOOL        PopFontChooseFont            (HWND) ;
void        PopFontSetFont               (HWND) ;
void PopFontDeinitialize (void) ;
                        // Functions in POPPRNT.C

BOOL PopPrntPrintFile (HINSTANCE, HWND, HWND, PTSTR) ;

                        // Global variables

static HWND  hDlgModeless ;
static TCHAR szAppName[] = TEXT ("PopPad") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                        PSTR szCmdLine, int iCmdShow)
{
          MSG       msg ;
          HWND      hwnd ;
```

```
     HACCEL    hAccel ;
     WNDCLASS  wndclass ;

     wndclass.style                    = CS_HREDRAW |
     wndclass.lpfnWndProc                 = WndProc ;
     wndclass.cbClsExtra                  = 0 ;
     wndclass.cbWndExtra                  = 0 ;
     wndclass.hInstance                   = hInstance ;
     wndclass.hIcon                    = LoadIcon (hInst
     wndclass.hCursor                     = LoadCursor (N
     wndclass.hbrBackground            = (HBRUSH) GetS
     wndclass.lpszMenuName             = szAppName ;
     wndclass.lpszClassName            = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (  NULL, TEXT ("This program requires W
                        szAppName, MB_ICONERROR) ;
          return 0 ;
```

```
}

hwnd = CreateWindow (szAppName, NULL,
                     WS_OVERLAPPEDWINDOW,
                     CW_USEDEFAULT, CW_USEDEFAULT,
                    CW_USEDEFAULT, CW_USEDEFAULT,
                     NULL, NULL, hInstance, szCmdLine) ;

ShowWindow (hwnd, iCmdShow) ;
UpdateWindow (hwnd) ;
hAccel = LoadAccelerators (hInstance, szAppName) ;

while (GetMessage (&msg, NULL, 0, 0))
{
     if (hDlgModeless == NULL || !IsDialogMessage (hDlgI
     {
                  if (!TranslateAccelerator (hwnd, hAccel, &m
  {
                       TranslateMessage (&msg) ;
```

```
                        DispatchMessage (&msg) ;
                    }
                }
    }
        return msg.wParam ;
}

void DoCaption (HWND hwnd, TCHAR * szTitleName)
{
        TCHAR szCaption[64 + MAX_PATH] ;
        wsprintf (szCaption, TEXT ("%s - %s"), szAppName,
                        szTitleName[0] ? szTitleName : UNTITL
        SetWindowText (hwnd, szCaption) ;
}

void OkMessage (HWND hwnd, TCHAR * szMessage, TCHAR * szT
{
        TCHAR szBuffer[64 + MAX_PATH] ;
        wsprintf (szBuffer, szMessage, szTitleName[0] ? szTitleNa
```

```
        MessageBox (hwnd, szBuffer, szAppName, MB_OK | MB_IC
}

short AskAboutSave (HWND hwnd, TCHAR * szTitleName)
{
        TCHAR        szBuffer[64 + MAX_PATH] ;
        int   iReturn ;

        wsprintf (szBuffer, TEXT ("Save current changes in %s?"),
                                szTitleName[0] ? szTitleName : UNTITL

        iReturn = MessageBox (hwnd, szBuffer, szAppName,
                        MB_YESNOCANCEL | MB_ICONQUESTION) ;
        if (iReturn == IDYES)
              if (!SendMessage (hwnd, WM_COMMAND, IDM_FILE_S
                                iReturn = IDCANCEL ;

        return iReturn ;
}
```

```
LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
        static BOOL                     bNeedSave = FALSE ;
        static HINSTANCE hInst ;
        static HWND                      hwndEdit ;
        static int                     iOffset ;
        static TCHAR                     szFileName[MAX_PATH], szTi
        static UINT                     messageFindReplace ;
        int                           iSelBeg, iSelEnd, iEnable ;
        LPFINDREPLACE                     pfr ;

        switch (message)
        {
        case WM_CREATE:
              hInst = ((LPCREATESTRUCT) lParam) -> hInstance ;
                              // Create the edit control child window
              hwndEdit = CreateWindow (TEXT ("edit"), NULL,
                   WS_CHILD | WS_VISIBLE | WS_HSCROLL | WS_VS
```

```
                    WS_BORDER | ES_LEFT | ES_MULTILINE |
                    ES_NOHIDESEL | ES_AUTOHSCROLL | ES_AUTOVSC
                    0, 0, 0, 0,
                    hwnd, (HMENU) EDITID, hInst, NULL) ;

               SendMessage (hwndEdit, EM_LIMITTEXT, 32000, 0L) ;
                         // Initialize common dialog box stuff
               PopFileInitialize (hwnd) ;
               PopFontInitialize (hwndEdit) ;

               messageFindReplace = RegisterWindowMessage (FIN
               DoCaption (hwnd, szTitleName) ;
               return 0 ;
          case   WM_SETFOCUS:
               SetFocus (hwndEdit) ;
               return 0 ;

     case   WM_SIZE:
               MoveWindow (hwndEdit, 0, 0, LOWORD (lParam), HIW
```

```
        return 0 ;

    case   WM_INITMENUPOPUP:
        switch (lParam)
       {
        case 1:              // Edit menu

                                // Enable Undo if edit control can

                            EnableMenuItem ((HMENU) wParam, ID
                    SendMessage (hwndEdit, EM_CANUNDO, 0, 0L)
                            MF_ENABLED : MF_GRAYED) ;

                                // Enable Paste if text is in the clip

                            EnableMenuItem ((HMENU) wParam, ID
                        IsClipboardFormatAvailable (CF_TEXT) ?
                            MF_ENABLED : MF_GRAYED) ;
```

```
                    // Enable Cut, Copy, and Del if text is s

               SendMessage (hwndEdit, EM_GETSEL,    (WPARA
                   (LPARAM) &iSelEnd) ;

               iEnable = iSelBeg != iSelEnd ? MF_ENABLED : M

               EnableMenuItem ((HMENU) wParam, IDM_EDIT_
               EnableMenuItem ((HMENU) wParam, IDM_EDIT_
               EnableMenuItem ((HMENU) wParam, IDM_EDIT_
                    break ;

          case 2:                         // Search menu

                    // Enable Find, Next, and Replace if mo
                    //   dialogs are not already active

                    iEnable = hDlgModeless == NULL ?
```

```
                MF_ENABLED : MF_GRAYED ;
                EnableMenuItem ((HMENU) wParam, IDM_SEAR
                        EnableMenuItem ((HMENU) wParam, ID
                        EnableMenuItem ((HMENU) wParam, ID
                         break ;
        }
        return 0 ;

    case   WM_COMMAND:
                                    // Messages from edit contro

        if (lParam && LOWORD (wParam) == EDITID)
        {
                        switch (HIWORD (wParam))
              {
              case   EN_UPDATE :
                        bNeedSave = TRUE ;
                        return 0 ;
```

```
        case   EN_ERRSPACE :
                 case   EN_MAXTEXT :
                MessageBox (hwnd, TEXT ("Edit control out of spa
                  szAppName, MB_OK | MB_ICONSTOP) ;
                            return 0 ;
                          }
                 break ;
               }

switch (LOWORD (wParam))
              {
                          // Messages from File menu
               case   IDM_FILE_NEW:
                            if (bNeedSave && IDCANCEL == AskAb
                                        return 0 ;

                            SetWindowText (hwndEdit, TEXT ("\0")
                            szFileName[0]  = '\0' ;
                            szTitleName[0] = '\0' ;
```

```
                    DoCaption (hwnd, szTitleName) ;
                    bNeedSave = FALSE ;
                   return 0 ;

     case   IDM_FILE_OPEN:
if (bNeedSave && IDCANCEL == AskAboutSave (hwnd, s
   return 0 ;
if (PopFileOpenDlg (hwnd, szFileName, szTitleName))
     {
  if (!PopFileRead (hwndEdit, szFileName))
                                  {
   OkMessage (hwnd, TEXT ("Could not read file %s!"),
            szTitleName) ;
           szFileName[0]  = '\0' ;
          szTitleName[0] = '\0' ;
                                  }
                    }
```

```
                        DoCaption (hwnd, szTitleName) ;

        bNeedSave = FALSE ;

        return 0 ;


case   IDM_FILE_SAVE:

      if     (szFileName[0])

       {

            if (PopFileWrite (hwndEdit, szFileName))

            {

                    bNeedSave = FALSE ;

                   return 1 ;

            }

            else

            {

                    OkMessage (hwnd, TEXT ("Could not write

                            szTitleName) ;

                                return 0 ;

                   }

                }
```

```
        //fall through
    case   IDM_FILE_SAVE_AS:
                if (PopFileSaveDlg (hwnd, szFileName, szTitleNa
     {
                                DoCaption (hwnd, szTitleName) ;

                                if (PopFileWrite (hwndEdit, szFileN
                                {
                                                bNeedSave =
                                               return 1 ;
                                }
                                      else
                               {
            OkMessage (hwnd, TEXT ("Could not write file %s"
                      szTitleName) ;
                                                    return 0 ;
                                }
                                }
```

```
                    return 0 ;

case   IDM_FILE_PRINT:
                if (!PopPrntPrintFile (hInst, hwnd, hwndEdit, szTi
            OkMessage (    hwnd, TEXT ("Could not print file %s"
                    szTitleName) ;
                return 0 ;

      case   IDM_APP_EXIT:
                SendMessage (hwnd, WM_CLOSE, 0, 0) ;
                return 0 ;

                                        // Messages from Edit m

      case   IDM_EDIT_UNDO:
                SendMessage (hwndEdit, WM_UNDO, 0, 0) ;
                return 0 ;

      case   IDM_EDIT_CUT:
```

```
                SendMessage (hwndEdit, WM_CUT, 0, 0) ;
                return 0 ;

        case   IDM_EDIT_COPY:
                SendMessage (hwndEdit, WM_COPY, 0, 0) ;
                return 0 ;

        case   IDM_EDIT_PASTE:
                SendMessage (hwndEdit, WM_PASTE, 0, 0) ;
                return 0 ;

        case   IDM_EDIT_CLEAR:
                SendMessage (hwndEdit, WM_CLEAR, 0, 0) ;
                return 0 ;

        case   IDM_EDIT_SELECT_ALL:
                SendMessage (hwndEdit, EM_SETSEL, 0, -1) ;
                return 0 ;
```

```
                                        // Messages from Search me
    case   IDM_SEARCH_FIND:
                    SendMessage (hwndEdit, EM_GETSEL, 0, (LPARA
                    hDlgModeless = PopFindFindDlg (hwnd) ;
                    return 0 ;

          case   IDM_SEARCH_NEXT:
                    SendMessage (hwndEdit, EM_GETSEL, 0, (LPARA

                   if (PopFindValidFind ())
                              PopFindNextText (hwndEdit, &iOffset) ;
                    else
                              hDlgModeless = PopFindFindDlg (hwnd

                    return 0 ;

          case   IDM_SEARCH_REPLACE:
                    SendMessage (hwndEdit, EM_GETSEL, 0, (LPARA
```

```
                hDlgModeless = PopFindReplaceDlg (hwnd) ;
                return 0 ;

        case   IDM_FORMAT_FONT:
                if (PopFontChooseFont (hwnd))
                        PopFontSetFont (hwndEdit) ;

                return 0 ;

                                        // Messages from Help menu

case   IDM_HELP:
                OkMessage (hwnd,      TEXT ("Help not yet imple
            TEXT ("\0")) ;
                return 0 ;

        case   IDM_APP_ABOUT:
                DialogBox (hInst, TEXT ("AboutBox"), hwnd, Abo
                return 0 ;
```

```
   }
        break ;
case       WM_CLOSE:
        if (!bNeedSave || IDCANCEL != AskAboutSave (hwnd, szTi
                DestroyWindow (hwnd) ;

             return 0 ;
        case   WM_QUERYENDSESSION :
             if (!bNeedSave || IDCANCEL != AskAboutSave (hwnd,
                  return 1 ;

             return 0 ;

        case   WM_DESTROY:
             PopFontDeinitialize () ;
             PostQuitMessage (0) ;
             return 0 ;
```

```
        default:
                        // Process "Find-Replace" messages
        if (message == messageFindReplace)
        {
                pfr = (LPFINDREPLACE) lParam ;
                 if     (pfr->Flags & FR_DIALOGTERM)
                        hDlgModeless = NULL ;

                 if     (pfr->Flags & FR_FINDNEXT)
              if (!PopFindFindText (hwndEdit, &iOffset, pfr))
              OkMessage (hwnd,      TEXT ("Text not found!"),
           TEXT ("\0")) ;

                 if (pfr->Flags & FR_REPLACE || pfr->Flags &
                        if (!PopFindReplaceText (hwndEdit, &iO
                        OkMessage (hwnd,     TEXT ("Text not f
           TEXT ("\0")) ;

                 if (pfr->Flags & FR_REPLACEALL)
```

```
                                    while (PopFindReplaceText (hwnd
                                    return 0 ;
          }
          break ;
    }
    return DefWindowProc (hwnd, message, wParam, lParam) ;
}

BOOL CALLBACK AboutDlgProc (HWND hDlg, UINT message,WPA
{
          switch (message)
          {
          case   WM_INITDIALOG:
                return TRUE ;

          case   WM_COMMAND:
                switch (LOWORD (wParam))
                {
```

```
                case IDOK:
                                EndDialog (hDlg, 0) ;
                                return TRUE ;
                }
        break ;
        }
        return FALSE ;
}
```

POPFILE.C

```
/*------------------------------------------------------------------------
  POPFILE.C -- Popup Editor File Functions
------------------------------------------------------------------------*/
#include <windows.h>
#include <commdlg.h>

static OPENFILENAME ofn ;
void PopFileInitialize (HWND hwnd)
{
```

```
static TCHAR szFilter[] =     TEXT ("Text Files (*.TXT)\0*.txt
                         TEXT ("ASCII Files (*.ASC)\0*.asc\0") \
                          TEXT ("All Files (*.*)\0*.*\0\0") ;


ofn.lStructSize                  = sizeof (OPENFILENAME) ;
ofn.hwndOwner                      = hwnd ;
ofn.hInstance                     = NULL ;
ofn.lpstrFilter                  = szFilter ;
ofn.lpstrCustomFilter = NULL ;
ofn.nMaxCustFilter    = 0 ;
ofn.nFilterIndex      = 0 ;
ofn.lpstrFile        = NULL ;            // Set in Open and Clos
ofn.nMaxFile              = MAX_PATH ;
ofn.lpstrFileTitle          = NULL ;            // Set in Open ar
ofn.nMaxFileTitle           = MAX_PATH ;
ofn.lpstrInitialDir         = NULL ;
ofn.lpstrTitle              = NULL ;
ofn.Flags                  = 0 ;                    // Set in Open a
```

```
          ofn.nFileOffset             = 0 ;
          ofn.nFileExtension          = 0 ;
          ofn.lpstrDefExt             = TEXT ("txt") ;
          ofn.lCustData               = 0L ;
          ofn.lpfnHook                = NULL ;
          ofn.lpTemplateName          = NULL ;
}

BOOL PopFileOpenDlg (HWND hwnd, PTSTR pstrFileName, PTSTR
{
          ofn.hwndOwner               = hwnd ;
          ofn.lpstrFile               = pstrFileName ;
          ofn.lpstrFileTitle          = pstrTitleName ;
          ofn.Flags                   = OFN_HIDEREADONLY | OFN_CREA

          return GetOpenFileName (&ofn) ;
}

BOOL PopFileSaveDlg (HWND hwnd, PTSTR pstrFileName, PTSTR
{
```

```
        ofn.hwndOwner         = hwnd ;
        ofn.lpstrFile         = pstrFileName ;
        ofn.lpstrFileTitle    = pstrTitleName ;
        ofn.Flags             = OFN_OVERWRITEPROMPT ;

        return GetSaveFileName (&ofn) ;
}

BOOL PopFileRead (HWND hwndEdit, PTSTR pstrFileName)
{
        BYTE          bySwap ;
        DWORD         dwBytesRead ;
        HANDLE        hFile ;
        int           i, iFileLength, iUniTest ;
        PBYTE         pBuffer, pText, pConv ;

                      // Open the file.
        if (INVALID_HANDLE_VALUE ==
                    (hFile = CreateFile (pstrFileName, GENERIC_REA
```

```
                    NULL, OPEN_EXISTING, 0, NULL)))
          return FALSE ;
               // Get file size in bytes and allocate memory for read
               // Add an extra two bytes for zero termination.

          iFileLength = GetFileSize (hFile, NULL) ;
          pBuffer = malloc (iFileLength + 2) ;

               // Read file and put terminating zeros at end.
          ReadFile (hFile, pBuffer, iFileLength, &dwBytesRead, NULL
          CloseHandle (hFile) ;
          pBuffer[iFileLength] = '\0' ;
          pBuffer[iFileLength + 1] = '\0' ;

               // Test to see if the text is Unicode
   iUniTest = IS_TEXT_UNICODE_SIGNATURE | IS_TEXT_UNICODE_
   if (IsTextUnicode (pBuffer, iFileLength, &iUniTest))
{
               pText = pBuffer + 2 ;
               iFileLength -= 2 ;
```

```
        if (iUniTest & IS_TEXT_UNICODE_REVERSE_SIGNATURE)
    {
                    for (i = 0 ; i < iFileLength / 2 ; i++)
                    {
                            bySwap = ((BYTE *) pText) [2 * i] ;
                ((BYTE *) pText) [2 * i] = ((BYTE *) pText) [2 * i + 1] ;
                ((BYTE *) pText) [2 * i + 1] = bySwap ;
                    }
    }

                            // Allocate memory for possibly converted s
                pConv = malloc (iFileLength + 2) ;
                            // If the edit control is not Unicode, convert
                           // non-Unicode (i.e., in general, wide charac
#ifndef UNICODE
                WideCharToMultiByte (CP_ACP, 0, (PWSTR) pText, -1,
                   iFileLength + 2, NULL, NULL) ;
                            // If the edit control is Unicode, just copy th
```

```
#else
        lstrcpy ((PTSTR) pConv, (PTSTR) pText) ;
#endif

   }
        else                // the file is not Unicode
    {
         pText = pBuffer ;
                            // Allocate memory for possibly converted s
              pConv = malloc (2 * iFileLength + 2) ;
                            // If the edit control is Unicode, convert ASC
#ifdef UNICODE
        MultiByteToWideChar (CP_ACP, 0, pText, -1, (PTSTR) pCon
                                 iFileLength + 1) ;
                                      // If not, just copy buffer
#else
              lstrcpy ((PTSTR) pConv, (PTSTR) pText) ;
#endif
        }
```

```
          SetWindowText (hwndEdit, (PTSTR) pConv) ;
          free (pBuffer) ;
          free (pConv) ;

          return TRUE ;
}

BOOL PopFileWrite (HWND hwndEdit, PTSTR pstrFileName)
{
          DWORD        dwBytesWritten ;
          HANDLE    hFile ;
          int          iLength ;
         PTSTR        pstrBuffer ;
          WORD         wByteOrderMark = 0xFEFF ;
                    // Open the file, creating it if necessary

          if (INVALID_HANDLE_VALUE ==
                    (hFile = CreateFile (pstrFileName, GENERIC_WRI
```

```
            NULL, CREATE_ALWAYS, 0, NULL)))
               return FALSE ;
               // Get the number of characters in the edit control an
               // memory for them.

          iLength = GetWindowTextLength (hwndEdit) ;
          pstrBuffer = (PTSTR) malloc ((iLength + 1) * sizeof (TCHA

          if (!pstrBuffer)
          {
               CloseHandle (hFile) ;
               return FALSE ;
          }

               // If the edit control will return Unicode text, write the
               // byte order mark to the file.

#ifdef UNICODE
          WriteFile (hFile, &wByteOrderMark, 2, &dwBytesWritten, N
#endif
```

```
            // Get the edit buffer and write that out to the file.

        GetWindowText (hwndEdit, pstrBuffer, iLength + 1) ;

        WriteFile (hFile, pstrBuffer, iLength * sizeof (TCHAR),

                                &dwBytesWritten, NULL) ;

        if ((iLength * sizeof (TCHAR)) != (int) dwBytesWritten)

   {

            CloseHandle (hFile) ;

            free (pstrBuffer) ;

            return FALSE ;

        }


        CloseHandle (hFile) ;

        free (pstrBuffer) ;


        return TRUE ;

}
```

POPFIND.C

```
```

```
/*------------------------------------------------------------------------
  POPFIND.C -- Popup Editor Search and Replace Functions
------------------------------------------------------------------------*/
#include <windows.h>
#include <commdlg.h>
#include <tchar.h>                    // for _tcsstr (strstr for Unicod

#define MAX_STRING_LEN   256

static TCHAR szFindText [MAX_STRING_LEN] ;
static TCHAR szReplText [MAX_STRING_LEN] ;

HWND PopFindFindDlg (HWND hwnd)
{
     static FINDREPLACE fr ;       // must be static for modeless


     fr.lStructSize             = sizeof (FINDREPLACE) ;
     fr.hwndOwner               = hwnd ;
     fr.hInstance               = NULL ;
     fr.Flags                   = FR_HIDEUPDOWN | FR_HIDEMATCH
```

```
          fr.lpstrFindWhat      = szFindText ;
          fr.lpstrReplaceWith   = NULL ;
          fr.wFindWhatLen       = MAX_STRING_LEN ;
          fr.wReplaceWithLen    = 0 ;
          fr.lCustData          = 0 ;
          fr.lpfnHook           = NULL ;
          fr.lpTemplateName     = NULL ;

          return FindText (&fr) ;
}

HWND PopFindReplaceDlg (HWND hwnd)
{
          static FINDREPLACE fr ;       // must be static for modeless

          fr.lStructSize        = sizeof (FINDREPLACE) ;
          fr.hwndOwner          = hwnd ;
          fr.hInstance          = NULL ;
          fr.Flags              = FR_HIDEUPDOWN | FR_HIDEMATCH
```

```
          fr.lpstrFindWhat             = szFindText ;
          fr.lpstrReplaceWith          = szReplText ;
          fr.wFindWhatLen              = MAX_STRING_LEN ;
          fr.wReplaceWithLen           = MAX_STRING_LEN ;
          fr.lCustData                 = 0 ;
          fr.lpfnHook                  = NULL ;
          fr.lpTemplateName            = NULL ;

          return ReplaceText (&fr) ;
}

BOOL PopFindFindText (HWND hwndEdit, int * piSearchOffset, LP
{
          int    iLength, iPos ;
          PTSTR  pstrDoc, pstrPos ;

                    // Read in the edit document

          iLength = GetWindowTextLength (hwndEdit) ;
```

```
     if (NULL == (pstrDoc = (PTSTR) malloc ((iLength + 1) * si
          return FALSE ;

     GetWindowText (hwndEdit, pstrDoc, iLength + 1) ;

               // Search the document for the find string

     pstrPos = _tcsstr (pstrDoc + * piSearchOffset, pfr->lpstrFi
free (pstrDoc) ;

               // Return an error code if the string cannot be fo

     if (pstrPos == NULL)
          return FALSE ;

               // Find the position in the document and the nev

     iPos = pstrPos - pstrDoc ;
```

```
        * piSearchOffset = iPos + lstrlen (pfr->lpstrFindWhat) ;


                    // Select the found text
        SendMessage (hwndEdit, EM_SETSEL, iPos, * piSearchOffs
        SendMessage (hwndEdit, EM_SCROLLCARET, 0, 0) ;


        return TRUE ;
}
BOOL PopFindNextText (HWND hwndEdit, int * piSearchOffset)
{
        FINDREPLACE fr ;
   fr.lpstrFindWhat = szFindText ;
   return PopFindFindText (hwndEdit, piSearchOffset, &fr) ;
}

BOOL PopFindReplaceText (HWND hwndEdit, int * piSearchOffset
{
       // Find the text
   if (!PopFindFindText (hwndEdit, piSearchOffset, pfr))
```

```
        return FALSE ;

        // Replace it
    SendMessage (hwndEdit, EM_REPLACESEL, 0, (LPARAM) pfr->
lpstrReplaceWith) ;
    return TRUE ;
}

BOOL PopFindValidFind (void)
{
    return * szFindText != '\0' ;
}
```

POPFONT.C

```
/*------------------------------------------------
  POPFONT.C -- Popup Editor Font Functions
------------------------------------------------*/
#include <windows.h>
#include <commdlg.h>
```

```
static LOGFONT logfont ;
static HFONT   hFont ;

BOOL PopFontChooseFont (HWND hwnd)
{
   CHOOSEFONT cf ;
         cf.lStructSize             = sizeof (CHOOSEFONT) ;
         cf.hwndOwner               = hwnd ;
         cf.hDC                     = NULL ;
         cf.lpLogFont               = &logfont ;
         cf.iPointSize              = 0 ;
         cf.Flags                   = CF_INITTOLOGFONTSTRUCT |
         cf.rgbColors                 = 0 ;
         cf.lCustData                 = 0 ;
         cf.lpfnHook                  = NULL ;
         cf.lpTemplateName              = NULL ;
         cf.hInstance                 = NULL ;
         cf.lpszStyle                 = NULL ;
```

```
        cf.nFontType                = 0 ;                   // Return
        cf.nSizeMin                 = 0 ;
        cf.nSizeMax                 = 0 ;

        return ChooseFont (&cf) ;
}

void PopFontInitialize (HWND hwndEdit)
{
        GetObject (GetStockObject (SYSTEM_FONT), sizeof (LOGF
                                (PTSTR) &logfont) ;
        hFont = CreateFontIndirect (&logfont) ;
        SendMessage (hwndEdit, WM_SETFONT, (WPARAM) hFont
}

void PopFontSetFont (HWND hwndEdit)
{
  HFONT hFontNew ;
   RECT  rect ;
```

```
          hFontNew = CreateFontIndirect (&logfont) ;
          SendMessage (hwndEdit, WM_SETFONT, (WPARAM) hFontN
          DeleteObject (hFont) ;
          hFont = hFontNew ;
          GetClientRect (hwndEdit, &rect) ;
          InvalidateRect (hwndEdit, &rect, TRUE) ;
}

void       PopFontDeinitialize (void)
{
          DeleteObject (hFont) ;
}
```

POPPRNT0.C

```
/*------------------------------------------------------------------
  POPPRNT0.C -- Popup Editor Printing Functions (dummy version
------------------------------------------------------------------*/
#include <windows.h>
BOOL PopPrntPrintFile (    HINSTANCE hInst, HWND hwnd, HWND
```

```
                                                    PTSTR pstrTitleNam
{
          return FALSE ;
}
```

POPPAD.RC

```
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
//////////////////////////////////////////////////////////////////////////////
// Dialog
ABOUTBOX DIALOG DISCARDABLE  32, 32, 180, 100
STYLE DS_MODALFRAME | WS_POPUP
FONT 8, "MS Sans Serif"
BEGIN
   DEFPUSHBUTTON "OK",IDOK,66,80,50,14
   ICON                                          "POPPAD",IDC_STATIC,7
   CTEXT                                          "PopPad",IDC_STATIC,4
```

```
    CTEXT           "Popup Editor for Windows",IDC_STATIC,7,40,166,8
    CTEXT           "(c) Charles Petzold, 1998",IDC_STATIC,7,52,166,8
END

PRINTDLGBOX DIALOG DISCARDABLE  32, 32, 186, 95
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTIO
CAPTION "PopPad"
FONT 8, "MS Sans Serif"
BEGIN
    PUSHBUTTON      "Cancel",IDCANCEL,67,74,50,14
    CTEXT                                   "Sending",IDC_STATIC,8,8
    CTEXT           "",IDC_FILENAME,8,28,172,8
    CTEXT           "to print spooler.",IDC_STATIC,8,48,172,8
END

//////////////////////////////////////////////////////////////////////
// Menu

POPPAD MENU DISCARDABLE
BEGIN
    POPUP           "&File"
```

```
    BEGIN
    MENUITEM      "&New\tCtrl+N",   IDM_FILE_NEW
  MENUITEM       "&Open...\tCtrl+O",IDM_FILE_OPEN
  MENUITEM      "&Save\tCtrl+S",   IDM_FILE_SAVE
  MENUITEM      "Save &As...",     IDM_FILE_SAVE_AS
  MENUITEM      SEPARATOR
  MENUITEM      "&Print\tCtrl+P",  IDM_FILE_PRINT
  MENUITEM      SEPARATOR
MENUITEM      "E&xit",          IDM_APP_EXIT
END
  POPUP "&Edit"
BEGIN
  MENUITEM      "&Undo\tCtrl+Z",   IDM_EDIT_UNDO
  MENUITEM      SEPARATOR
  MENUITEM      "Cu&t\tCtrl+X",    IDM_EDIT_CUT
  MENUITEM      "&Copy\tCtrl+C",   IDM_EDIT_COPY
  MENUITEM      "&Paste\tCtrl+V",  IDM_EDIT_PASTE
  MENUITEM      "De&lete\tDel",    IDM_EDIT_CLEAR
  MENUITEM      SEPARATOR
```

```
  MENUITEM      "&Select All",     IDM_EDIT_SELECT_ALL
END
    POPUP     "&Search"
BEGIN
  MENUITEM      "&Find...\tCtrl+F",IDM_SEARCH_FIND
  MENUITEM      "Find &Next\tF3",  IDM_SEARCH_NEXT
  MENUITEM      "&Replace...\tCtrl+R", IDM_SEARCH_REPLACE
END
    POPUP    "F&ormat"
BEGIN
  MENUITEM      "&Font...",
END
    POPUP "&Help"
   BEGIN
   MENUITEM      "&Help",               IDM_HELP
  MENUITEM      "&About PopPad...",  IDM_APP_ABOUT
    END
END
```

```
////////////////////////////////////////////////////////////////////////
// Accelerator
POPPAD ACCELERATORS DISCARDABLE
BEGIN
  VK_BACK,     IDM_EDIT_UNDO,   VIRTKEY,    ALT,    NOINVERT
  VK_DELETE,  IDM_EDIT_CLEAR,  VIRTKEY,      NOINVERT
  VK_DELETE,  IDM_EDIT_CUT,    VIRTKEY,      SHIFT,  NOINVERT
  VK_F1,     IDM_HELP,       VIRTKEY,       NOINVERT
  VK_F3,     IDM_SEARCH_NEXT, VIRTKEY,      NOINVERT
  VK_INSERT,  IDM_EDIT_COPY,   VIRTKEY,      CONTROL,  NOINV
  VK_INSERT,                IDM_EDIT_PASTE,      VIRTKEY,     SHIF
  "^C",       IDM_EDIT_COPY,       ASCII,  NOINVERT
  "^F",       IDM_SEARCH_FIND,     ASCII,       NOINVERT
   "^N",      IDM_FILE_NEW,        ASCII,       NOINVERT
    "^O",     IDM_FILE_OPEN,       ASCII,       NOINVERT
    "^P",     IDM_FILE_PRINT,      ASCII,  NOINVERT
    "^R",     IDM_SEARCH_REPLACE,   ASCII,  NOINVERT
    "^S",     IDM_FILE_SAVE,       ASCII,  NOINVERT
    "^V",     IDM_EDIT_PASTE,      ASCII,  NOINVERT
```

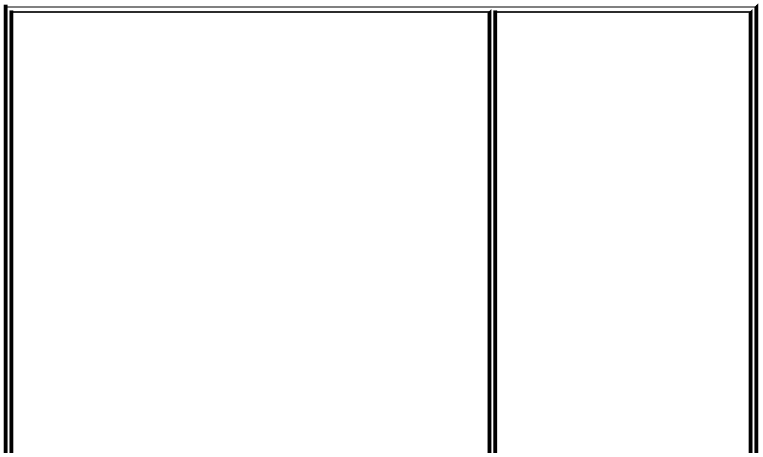
```
    "^X",        IDM_EDIT_CUT,         ASCII,  NOINVERT
    "^Z",        IDM_EDIT_UNDO,        ASCII,  NOINVERT
END

//////////////////////////////////////////////////////////////////////
// Icon
POPPAD                                          ICON    DISCARDABLE
```

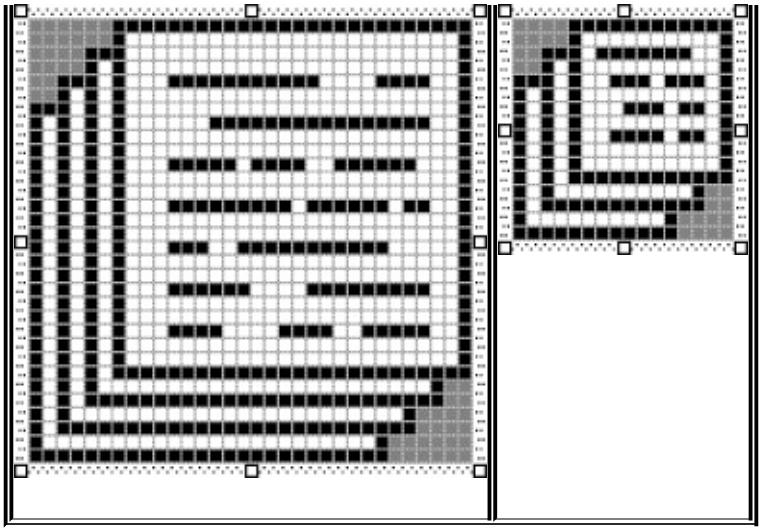
RESOURCE.H

```
// Microsoft Developer Studio generated include file.
// Used by poppad.rc
#define IDC_FILENAME          1000
#define IDM_FILE_NEW          40001
#define IDM_FILE_OPEN         40002
#define IDM_FILE_SAVE         40003
#define IDM_FILE_SAVE_AS      40004
#define IDM_FILE_PRINT        40005
#define IDM_APP_EXIT          40006
#define IDM_EDIT_UNDO         40007
```

```
#define IDM_EDIT_CUT         40008
#define IDM_EDIT_COPY        40009
#define IDM_EDIT_PASTE       40010
#define IDM_EDIT_CLEAR       40011
#define IDM_EDIT_SELECT_ALL  40012
#define IDM_SEARCH_FIND      40013
#define IDM_SEARCH_NEXT      40014
#define IDM_SEARCH_REPLACE   40015
#define IDM_FORMAT_FONT      40016
#define IDM_HELP             40017
#define IDM_APP_ABOUT        40018
```

POPPAD.ICO





POPPAD.RC

POPPAD.CPOPFILE.CFile OpenFile SaveI/OPOPFIND.C
POPFONT.CPOPPRNT0.C POPPRNT.CPO

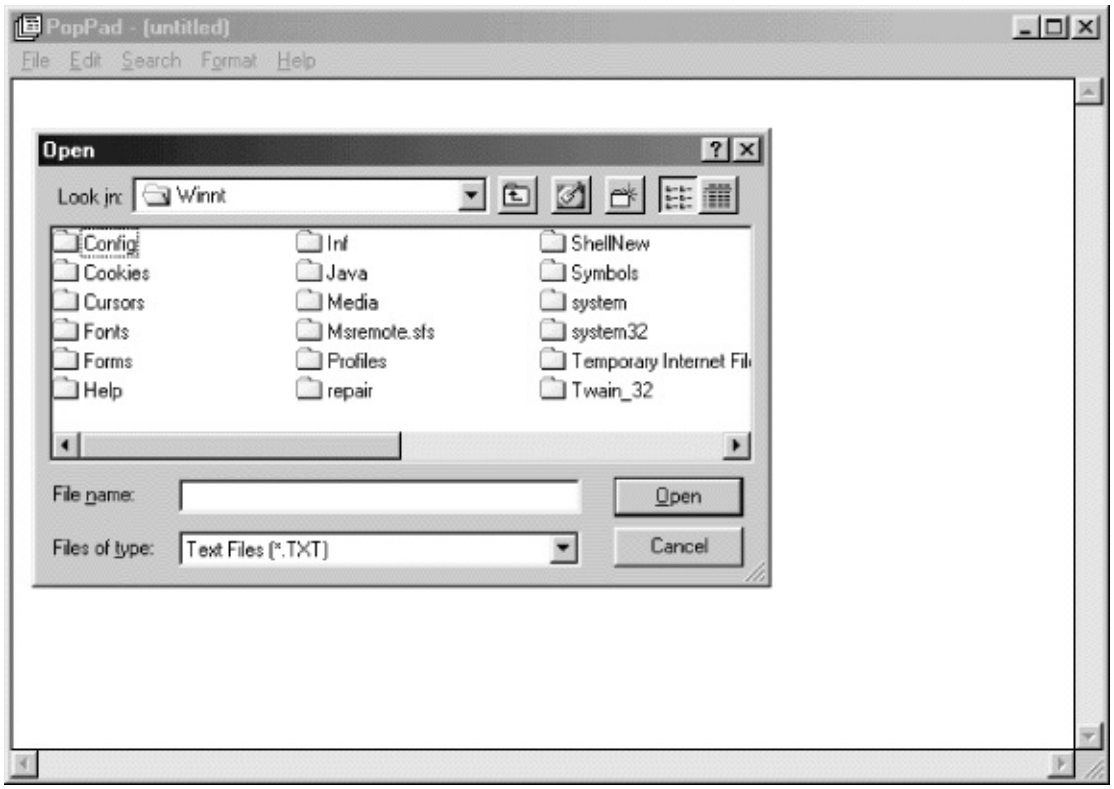
POPPAD.CPOPPAD.CWndProcszFileName
szTitleNamePOPPAD3DoCaptionOKMessageAskAboutSave

POPFILE.CFile OpenFile SaveI/OGetOpenFileName
GetSaveFileNameOPENFILENAMECOMMDLG.HPOPFILE.C
ofnofnPopFileInitializePOPPAD.CWndProcWM_CREATE

ofnGetOpenFileNameGetSaveFileName

hookPOPFILE.CFile
lStructSizehwndOwnerlpstrFilterlpstrFile
nMaxFilelpstrFileTitlenMaxFileTitleFlags
lpstrDefExt

FileOpenPOPPAD3POPFILEPopFileOpenDlg
PopFileOpenDlgOPENFILENAMEhwndOwnerlpstrFilelpstrFileTitleFlags
OFN_ CREATEPROMPTGetOpenFileName11-6



11-6 File Open

GetOpenFileNameOFN_CREATEPROMPTGetOpenFileName

POPFILE.CPopFileInitializeszFilter.TXT
.ASCASCIIOPENFILENAMElpstrFilter

OPENFILENAMEnFilterIndex

POPFILE.CPopFileSaveDlgFlagsOFN_OVERWRITEPROMPT
GetSaveFileNameFile SaveOFN_OVERWRITEPROMPT

UnicodeI/O

UnicodeUnicodePOPPAD3UnicodeUnicodeUnicode
Unicode

POPPAD3I/OUnicodePOPPAD3UnicodeUnicode

POPPAD3UnicodeUnicode

POPFILE.CPopFileWriteUnicode0xFEFFUnicode

PopFileReadIsTextUnicodeUnicodeMacintoshIntel
UnicodeUnicodePOPPAD3WideCharToMultiChar
WideCharToMultiCharANSIWindows

UnicodeUnicodeMultiCharToWideChar

`

WM_CREATEPOPFONT.CPOPPADPopFontInitializeLOGFONT
WM_SETFONTPopFontInitialize

POPPADWM_COMMANDPopFontChooseFontCHOOSEFONT
ChooseFontOKChooseFontTRUEPOPPADPopFontSetFont

WM_DESTROYPOPPADPopFontDeinitializePopFontSetFont

FindTextReplaceTextFINDREPLACE10-11POPFIND.C
PopFindFindDlgPopFindReplaceDlg

IsDialogMessageFindTextReplaceTextFINDREPLACE

FindTextReplaceTextFINDMSGSTRINGRegisterWindowMessage
WndProcWM_CREATE

WndProcRegisterWindowMessagelParamFINDREPLACEFlags
POPPAD3POPFIND.CPopFindFindTextPopFindReplaceText

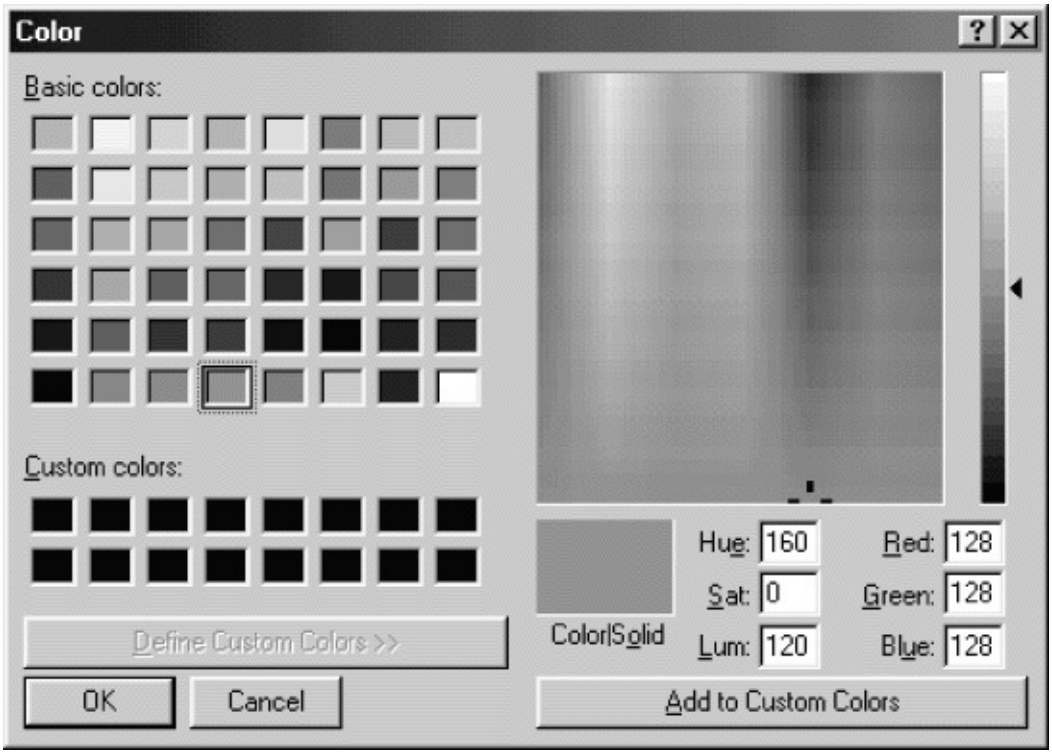
Windows

CO

COLORS3WindowsChooseColor11-7COLORS1COLORS2

```
11-7  COLORS3

COLORS3.C

/*------------------------------------------------------------------------
  COLORS3.C -- Version using Common Dialog Box
                                        (c) Charles Petzold, 1998
------------------------------------------------------------------------*/

#include <windows.h>
#include <commdlg.h>

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                          PSTR szCmdLine, int iCmdSh
{
          static CHOOSECOLOR    cc ;
          static COLORREF                crCustColors[16] ;

          cc.lStructSize                = sizeof (CHOOSECOLOR) ;
          cc.hwndOwner                  = NULL ;
          cc.hInstance                  = NULL ;
```

```
        cc.rgbResult                    = RGB (0x80, 0x80, 0x80) ;
        cc.lpCustColors                 = crCustColors ;
        cc.Flags                        = CC_RGBINIT | CC_FULLOPEN
        cc.lCustData                    = 0 ;
        cc.lpfnHook                     = NULL ;
    cc.lpTemplateName = NULL ;

        return ChooseColor (&cc) ;
}
```



11-7 COLORS3

ChooseColorCHOOSECOLOR16DWORDrgbResultFlags
CC_RGBINITrgbResult

ColorhwndOwnerNULLChooseColorDialogBoxDialogBoxNULL

Windows

Microsoft WindowsWindows

EditCutCopyPasteCutCopy
metafilePaste

Ctrl-XCtrl-C

POPPADEdit

WindowsWINUSER.HCF

- **CF_TEXT**NULLANSIcarriage ret

- **CF_OEMTEXT**CF_TEXTOEMWindowsMS-DOS

- **CF_UNICODETEXT**UnicodeCF_TEXTcarriage
  NULL0CF_UNICODETEXTWindows

- **CF_LOCALE**

CF_TEXTNULL

- **CF_SYLK**Microsoft MicrosoftMultiplanChartExcel ASCIIcarriage returnlinefeed
- **CF_DIF**(DIF)Software ArtsV linefeed

- **CF_BITMAP**
- **CF_DIB**
- **CF_PALETTE**CF_DIB

TIFF

- **CF_TIFF**(TIFF)MicrosoftAldusHewlett-PackardHewlett-Packard

metafile metafile

- **CF_METAFILEPICT**metafile
- **CF_ENHMETAFILE**metafile32Windows

- **CF_PENDATA**Windows
- **CF_WAVE**
- **CF_RIFF**Resource Interchange File Format
- **CF_HDROP**

CmallocWindowsmalloc

Windows16Windows

Windows API

```
hGlobal = GlobalAlloc (uiFlags, dwSize) ;
```

HGLOBALNULL

GlobalAlloc320GMEM_FIXEDGlobalAlloc

0GMEM,_ZEROINITWindowsGPTRGMEM_FIXED
GMEM_ZEROINIT

```
#define GPTR (GMEM_FIXED | GMEM_ZEROINIT)
```

```
hGlobal = GlobalReAlloc (hGlobal, dwSize, uiFlags) ;
```

GMEM_ZEROINIT0

```
dwSize = GlobalSize (hGlobal) ;
```

```
GlobalFree (hGlobal) ;
```

16WindowsWindowsGMEM_FIXED32Windows GMEM_FIXED16WindowsGlobalAllocGMEM_MOVEABLE Windows0

```
#define GHND (GMEM_MOVEABLE | GMEM_ZEROINIT)
```

GMEM_MOVEABLEWindows

GMEM_MOVEABLE16Windows

intGLOBALHANDLE

```
int * p ;

        GLOBALHANDLE hGlobal ;
```

```
hGlobal = GlobalAlloc (GHND, 1024) ;
```

Windows

```
p = (int *) GlobalLock (hGlobal) ;
```

Windows

```
GlobalUnlock (hGlobal) ;
```

WindowsWindows

GlobalFree

```
hGlobal = GlobalHandle (p) ;
```

WindowsWindows32Windows

GMEM_MOVEABLEGMEM_SHAREGlobalAllocGMEM_SHARE

ANSI(pString)iLengthNULLNULL

GlobalAllocNULL

```
hGlobal = GlobalAlloc (GHND | GMEM_SHARE, iLength + 1) ;
```

hGlobalNULL

```
pGlobal = GlobalLock (hGlobal) ;
```

```
for (i = 0 ; i < wLength ; i++)

   *pGlobal++ = *pString++ ;
```

GlobalAllocGHNDNULL

```
GlobalUnlock (hGlobal) ;
```

NULL

```
OpenClipboard (hwnd) ;

EmptyClipboard () ;
```

CF_TEXT

```
SetClipboardData (CF_TEXT, hGlobal) ;

CloseClipboard () ;
```

- OpenClipboardCloseClipboard
-
- SetClipboardDataSetClipboardData
  CloseClipboardSetClipboardDataCloseClipboard

CF_TEXT

```
bAvailable = IsClipboardFormatAvailable (CF_TEXT) ;
```

CF_TEXTTRUEPOPPAD2EditPaste
IsClipboardFormatAvailableCF_TEXT

```
OpenClipboard (hwnd) ;
```

```
hGlobal = GetClipboardData (CF_TEXT) ;
```

CF_TEXTNULLGetClipboardDataNULL

GetClipboardDataGetClipboardDataCloseClipboard

```
pText = (char *) malloc (GlobalSize (hGlobal)) ;
```

hGlobal hGlobalGetClipboardData

```
pGlobal = GlobalLock (hGlobal) ;
```

```
strcpy (pText, pGlobal) ;
```

C

```
while (*pText++ = *pGlobal++) ;
```

```
GlobalUnlock (hGlobal) ;

CloseClipboard () ;
```

pText

OpenClipboardOpenClipboardBOOL

Paste

Unicode

ANSICF_TEXTCF_OEMTEXTCF_UNICODETEXT

SetClipboardDataGetClipboardDataWindowsWindows SetClipboardDataCF_TEXTCF_OEMTEXTGetClipboardData CF_OEMTEXTCF_TEXT

Windows NTCF_UNICODETEXTCF_TEXTCF_OEMTEXT SetClipboardData GetClipboardDataUNICODEUNICODE CF_UNICODETEXTSetClipboardDataGetClipboardDataCF_TEXT

CLIPTEXT12-1

```
12-1  CLIPTEXT

CLIPTEXT.C

/*------------------------------------------------------------------------

  CLIPTEXT.C --        The Clipboard and Text

                                                (c) Charles Petzold, 199

------------------------------------------------------------------------*/
```

```
#include <windows.h>
#include "resource.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
#ifdef UNICODE
#define CF_TCHAR CF_UNICODETEXT
TCHAR szDefaultText[]             = TEXT ("Default Text - Unicode V
TCHAR szCaption[]                 = TEXT ("Clipboard Text Transfers
#else
#define CF_TCHAR CF_TEXT
TCHAR szDefaultText[] = TEXT ("Default Text - ANSI Version") ;
TCHAR szCaption[]                 = TEXT ("Clipboard Text Transfers

#endif
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                    PSTR szCmdLine, int iCmdShow)
{
          static TCHAR szAppName[] = TEXT ("ClipText") ;
          HACCEL                   hAccel ;
```

```
     HWND                  hwnd ;
     MSG                   msg ;
     WNDCLASS              wndclass ;

     wndclass.style                = CS_HREDRAW | CS_V
     wndclass.lpfnWndProc            = WndProc ;
     wndclass.cbClsExtra             = 0 ;
     wndclass.cbWndExtra             = 0 ;
     wndclass.hInstance              = hInstance ;
     wndclass.hIcon                  = LoadIcon (NULL, IDI_
     wndclass.hCursor                = LoadCursor (NULL,
     wndclass.hbrBackground          = (HBRUSH) GetStockO
     wndclass.lpszMenuName           = szAppName ;
     wndclass.lpszClassName          = szAppName ;

     if (!RegisterClass (&wndclass))
{
          MessageBox (  NULL, TEXT ("This program requires W
                                        szAppName, MB_IC
```

```
            return 0 ;
}

      hwnd = CreateWindow (szAppName, szCaption,
                  WS_OVERLAPPEDWINDOW,
                  CW_USEDEFAULT, CW_USEDEFAULT,
                  CW_USEDEFAULT, CW_USEDEFAULT,
                 NULL, NULL, hInstance, NULL) ;

      ShowWindow (hwnd, iCmdShow) ;
      UpdateWindow (hwnd) ;
hAccel = LoadAccelerators (hInstance, szAppName) ;

      while (GetMessage (&msg, NULL, 0, 0))
      {
            if (!TranslateAccelerator (hwnd, hAccel, &msg))
    {
                        TranslateMessage (&msg) ;
                        DispatchMessage (&msg) ;
```

```
            }
  }
        return msg.wParam ;
}

LRESULT CALLBACK WndProc ( HWND hwnd, UINT message, WPA
{
        static PTSTR         pText ;
        BOOL                 bEnable ;
        HGLOBAL              hGlobal ;
        HDC                  hdc ;
        PTSTR                pGlobal ;
        PAINTSTRUCT          ps ;
        RECT                 rect ;

  switch (message)
        {
        case   WM_CREATE:
             SendMessage (hwnd, WM_COMMAND, IDM_EDIT_RES
```

```
            return 0 ;

case   WM_INITMENUPOPUP:
            EnableMenuItem       ((HMENU)     wParam,IDM_EDIT
                  IsClipboardFormatAvailable    (CF_TCHAR) ? MF_

            bEnable = pText ? MF_ENABLED : MF_GRAYED ;

            EnableMenuItem       ((HMENU)     wParam,IDM_EDIT
            EnableMenuItem       ((HMENU)     wParam,IDM_EDIT
            EnableMenuItem       ((HMENU)     wParam,IDM_EDIT
            break ;

      case   WM_COMMAND:
            switch (LOWORD (wParam))
            {
            case   IDM_EDIT_PASTE:
                        OpenClipboard (hwnd) ;

                        if     (hGlobal = GetClipboardData (CF_
                        {
```

```
                    pGlobal = GlobalLock (hGlob
                    if (pText)
                    {
                              free (pText) ;
                              pText = NULL ;
                    }
                    pText = malloc (GlobalSize (hGlob
                    lstrcpy (pText, pGlobal) ;
                    InvalidateRect (hwnd, NULL, TRUE
               }
               CloseClipboard () ;
               return 0 ;

     case   IDM_EDIT_CUT:
     case   IDM_EDIT_COPY:
               if (!pText)
                    return 0 ;

               hGlobal = GlobalAlloc (GHND | GMEM_
```

```
            (lstrlen (pText) + 1) * sizeof (TCHAR)) ;

                    pGlobal = GlobalLock (hGlobal) ;

                    lstrcpy (pGlobal, pText) ;

                   GlobalUnlock (hGlobal) ;


                    OpenClipboard (hwnd) ;

                    EmptyClipboard () ;

                    SetClipboardData (CF_TCHAR, hGlobal

                    CloseClipboard () ;


                if (   LOWORD (wParam) == IDM_EDIT_COP

                        return 0 ;

        // fall through for IDM_EDIT_CUT

    case   IDM_EDIT_CLEAR:

                    if (pText)

                    {

                        free (pText) ;

                        pText = NULL ;

                    }

                InvalidateRect (hwnd, NULL, TRUE) ;
```

```
                return 0 ;

        case   IDM_EDIT_RESET:
                        if (pText)
                        {
                                free (pText) ;
                                pText = NULL ;
                        }
                pText = malloc ((lstrlen (szDefaultText) + 1) * si
                        lstrcpy (pText, szDefaultText) ;
                       InvalidateRect (hwnd, NULL, TRUE) ;
                        return 0 ;
        }
        break ;

case   WM_PAINT:
        hdc = BeginPaint (hwnd, &ps) ;

        GetClientRect (hwnd, &rect) ;
```

```
            if (pText != NULL)
            DrawText      (hdc, pText, -1, &rect, DT_EXPANDTABS

            EndPaint (hwnd, &ps) ;
            return 0 ;

        case   WM_DESTROY:
            if (   pText)
                        free (pText) ;

                    PostQuitMessage (0) ;
                    return 0 ;
        }
   return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

CLIPTEXT.RC

```
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
```

```
//////////////////////////////////////////////////////////////////////////
// Menu
CLIPTEXT MENU DISCARDABLE
BEGIN
    POPUP "&Edit"
   BEGIN
          MENUITEM "Cu&t\tCtrl+X",                  IDM_EDIT_CUT
      MENUITEM "&Copy\tCtrl+C",               IDM_EDIT_COPY
      MENUITEM "&Paste\tCtrl+V",              IDM_EDIT_PASTE
MENUITEM "De&lete\tDel",                IDM_EDIT_CLEAR
      MENUITEM SEPARATOR
      MENUITEM "&Reset",                IDM_EDIT_RESET
   END
END

//////////////////////////////////////////////////////////////////////////
// Accelerator
CLIPTEXT ACCELERATORS DISCARDABLE
BEGIN
```

```
  "C",          IDM_EDIT_COPY,    VIRTKEY, CONTROL, NOINVERT
  "V",          IDM_EDIT_PASTE,   VIRTKEY, CONTROL, NOINVERT
  VK_DELETE,                IDM_EDIT_CLEAR,   VIRTKEY, NOINVE
  "X",          IDM_EDIT_CUT,     VIRTKEY, CONTROL, NOINVERT
END
```

```
RESOURCE.H

// Microsoft Developer Studio generated include file.

// Used by ClipText.rc

#define IDM_EDIT_CUT          40001

#define IDM_EDIT_COPY         40002

#define IDM_EDIT_PASTE        40003

#define IDM_EDIT_CLEAR        40004

#define IDM_EDIT_RESET        40005
```

Windows NTUnicodeANSICLIPTEXT.C#ifdefUNICODE CF_TCHARCF_UNICODETEXTCF_TEXT IsClipboardFormatAvailableGetClipboardDataSetClipboardDataCF_TCHAR

EditResetpTextUnicodeUnicodeDefault Unicode versionUnicodeDefault Text - ANSI versio CopyCutDeletePastepTextWM_PAINTpText

UnicodeCLIPTEXTCopyUnicodePasteUnicodeANSI
ANSIUnicode

```
OpenClipboard                (hwnd) ;

EmptyClipboard                () ;

SetClipboardData          (iFormat, hGlobal) ;

CloseClipboard                () ;
```

```
OpenClipboard (hwnd) ;

hGlobal = GetClipboardData (iFormat) ;



CloseClipboard () ;
```

GetClipboardDataCloseClipboard

EmptyClipboardWindows

EmptyClipboardCloseClipboardSetClipboardDatametafile
metafilemetafile

SetClipboardData

```
OpenClipboard           (hwnd) ;

EmptyClipboard           () ;

SetClipboardData   (CF_TEXT, hGlobalText) ;

SetClipboardData   (CF_BITMAP, hBitmap) ;

SetClipboardData   (CF_METAFILEPICT, hGlobalMFP) ;

CloseClipboard           () ;
```

CF_TEXTCF_BITMAPCF_METAFILEPICTIsClipboardFormatAvailable TRUE

```
hGlobalText = GetClipboardData (CF_TEXT) ;
```

```
hBitmap = GetClipboardData (CF_BITMAP) ;
```

```
hGlobalMFP = GetClipboardData (CF_METAFILEPICT) ;
```

EmptyClipboardWindows

metafilemetafileWindowsCF_TEXTCF_OEMTEXT CF_UNICODETEXTCF_BITMAPCF_DIBCF_METAFILEPICT CF_ENHMETAFILE

EnumClipboardFormatsiFormat0

```
iFormat = 0 ;

OpenClipboard (hwnd) ;
```

0EnumClipboardFormatsiFormat0

```
while (iFormat = EnumClipboardFormats (iFormat))

{

  iFormat

}

CloseClipboard () ;
```

```
iCount = CountClipboardFormats () ;
```

WindowsSetClipboardDataNULL

```
OpenClipboard             (hwnd) ;

EmptyClipboard             () ;

SetClipboardData   (iFormat, NULL) ;

CloseClipboard            () ;
```

iFormatSetClipboardDataNULL

GetClipboardDataWindowsNULLWindows

OpenClipboardWindowsEmptyClipboard
Windows

WM_RENDERFORMATWM_RENDERALLFORMATS
WM_DESTROYCLIPBOARDGetClipboardDataWindows
WM_RENDERFORMATwParamWM_RENDERFORMAT
wParamSetClipboardDataWM_RENDERFORMAT
EmptyClipboardWindowsWM_DESTROYCLIPBOARD

SetClipboardDataNULLWM_RENDERALLFORMATS
SetClipboardDataWM_RENDERALLFORMATS
WM_DESTROYCLIPBOARDWM_DESTROY

WM_RENDERALLFORMATSWM_RENDERFORMAT

```
case      WM_RENDERALLFORMATS :

        OpenClipboard (hwnd) ;

        EmptyClipboard () ;

                                        // fall through

case      WM_RENDERFORMAT :

  //

        SetClipboardData (CF_TEXT, hGlobal) ;

        if (message == WM_RENDERALLFORMATS)

             CloseClipboard () ;
```

```
        return 0 ;
```

wParamWM_ REN

Windows

metafileSetClipboardDataGetClipboardDatawFormat
CF_DSPTEXTCF_DSPBITMAPCF_DSPMETAFILEPICT
CF_DSPENHMETAFILEDSPWindowsmetafile
CF_TEXTCF_BITMAPCF_DIBCF_METAFILEPICT
CF_ENHMETAFILEGetClipboardData

```
hwndClipOwner = GetClipboardOwner () ;
```

```
TCHAR szClassName [32] ;

//

GetClassName (hwndClipOwner, szClassName, 32) ;
```

CF_OWNERDISPLAYSetClipboardDataNULL

```
SetClipboardData (CF_OWNERDISPLAY, NULL) ;
```

WindowsCF_OWNERDISPLAY

NULLCF_OWNERDISPLAYSetClipboardDataWindows55

- **WM_ASKCBFORMATNAME**lParamwParam
- **WM_SIZECLIPBOARD**wParamlParamRECTRECT0 WindowswParam
- **WM_PAINTCLIPBOARD**wParamlParamPAINTSTRUCT hdc
- **WM_HSCROLLCLIPBOARDWM_VSCROLLCLIPBOARD** wParamlParamSB_THUMBPOSITIONlParam

WindowsWindowsSetClipboardDataGetClipboardData CF_OWNERDISPLAY

metafilemetafilemetafilemetafile

```
iFormat = RegisterClipboardFormat (szFormatName) ;
```

iFormat0xC0000xFFFFEnumClipboardFormatsASCII

```
GetClipboardFormatName (iFormat, psBuffer, iMaxCount) ;
```

WindowsiMaxCountpsBuffer

Windows

WindowsWindowsWindows

WindowsWindows

SetClipboardViewerWM_CREATE

```
static HWND hwndNextViewer ;

//

case      WM_CREATE :

   //

        hwndNextViewer = SetClipboardViewer (hwnd) ;
```

WindowshwndNextViewerNULL

WindowsWM_DRAWCLIPBOARDSendMessage
hwndNextViewerNULLhwndNextViewerNULL
WM_DRAWCLIPBOARDWM_PAINTCLIPBOARD
WM_PAINTCLIPBOARDCF_OWNERDISPLAYWM_
DRAWCLIPBOARDWindows

WM_DRAWCLIPBOARDhwndNextViewerNULL

```
case      WM_DRAWCLIPBOARD :

        if    (     hwndNextViewer)
```

```
                    SendMessage (hwndNextViewer, message, wPar
        InvalidateRect (hwnd, NULL, TRUE) ;
        return 0 ;
```

WM_PAINTOpenClipboardGetClipboardDataCloseClipboard

ChangeClipboardChain

```
ChangeClipboardChain (hwnd, hwndNextViewer) ;
```

ChangeClipboardChainWindowsWM_CHANGECBCHAINwParam ChangeClipboardChainlParamChangeClipboardChain

WM_CHANGECBCHAINwParamhwndNextViewer hwndNextViewerlParamWM_DRAWCLIPBOARDwParam hwndNextViewer hwndNextViewerNULL

```
case      WM_CHANGECBCHAIN :
        if ((HWND) wParam == hwndNextViewer)
            hwndNextViewer = (HWND) lParam ;
        else if (hwndNextViewer)
                    SendMessage (hwndNextViewer, message,
        return 0 ;
```

else ifhwndNextViewerNULLhwndNextViewerNULL

WM_DESTROYChangeClipboardChain

```
case       WM_DESTROY :
   ChangeClipboardChain (hwnd, hwndNextViewer) ;
        PostQuitMessage (0) ;
        return 0 ;
```

Windows

```
hwndViewer = GetClipboardViewer () ;
```

NULL

WindowsNULL

NULL

hwnd1SetClipboardViewerNULLhwndNextViewer

hwnd1

hwnd1NULL

hwnd2SetClipboardViewer hwnd1

hwnd2

hwnd2hwnd1

hwnd1NULL

(hwnd3)(hwnd4) SetClipboardViewer hwnd2hwnd3

hwnd4

hwnd4hwnd3

hwnd3hwnd2

hwnd2hwnd1

hwnd1NULL

WindowsWM_DRAWCLIPBOARDhwnd4hwnd4hwnd3hwnd3hwnd2 hwnd2hwnd1hwnd1

hwnd2

ChangeClipboardChain (hwnd2, hwnd1) ;

WindowswParamhwnd2lParamhwnd1WM_CHANGECBCHAINhwnd4 hwnd4hwnd3hwnd4hwnd3hwnd3wParam(hwnd2)lParam

hwnd4

hwnd4hwnd3

hwnd3hwnd1

hwnd1NULL

Windows12-2CLIPVIEWCF_TEXT

```
12-2  CLIPVIEW

CLIPVIEW.C

/*------------------------------------------------------------------------

  CLIPVIEW.C --Simple Clipboard Viewer
```

```
                                        (c) Charles Petzold, 1998
----------------------------------------------------------------------*/
#include <windows.h>
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
int WINAPI WinMain(     HINSTANCE hInstance, HINSTANCE hPrev
                PSTR szCmdLine, int iCmdShow)
{
        static TCHAR  szAppName[] = TEXT ("ClipView") ;
        HWND                    hwnd ;
        MSG                     msg ;
        WNDCLASS                wndclass ;

        wndclass.style                  = CS_HREDRAW | CS_VR
        wndclass.lpfnWndProc                = WndProc ;
        wndclass.cbClsExtra                 = 0 ;
        wndclass.cbWndExtra                 = 0 ;
        wndclass.hInstance                  = hInstance ;
        wndclass.hIcon                  = LoadIcon (NULL, IDI_
        wndclass.hCursor                    = LoadCursor (NULL,
```

```
     wndclass.hbrBackground          = (HBRUSH) GetStockO
     wndclass.lpszMenuName           = NULL ;
     wndclass.lpszClassName          = szAppName ;

     if (!RegisterClass (&wndclass))
{
          MessageBox (  NULL, TEXT ("This program requires W
                        szAppName, MB_ICONERROR) ;
          return 0 ;
}

     hwnd = CreateWindow (szAppName,
                          TEXT ("Simple Clipboard Viewer (Text C
               WS_OVERLAPPEDWINDOW,
               CW_USEDEFAULT, CW_USEDEFAULT,
               CW_USEDEFAULT, CW_USEDEFAULT,
              NULL, NULL, hInstance, NULL) ;
```

```
        ShowWindow (hwnd, iCmdShow) ;
        UpdateWindow (hwnd) ;

        while (GetMessage (&msg, NULL, 0, 0))
        {
             TranslateMessage (&msg) ;
             DispatchMessage (&msg) ;
        }
        return msg.wParam ;
}

LRESULT CALLBACK WndProc   (      HWND hwnd, UINT message,
{
        static HWND   hwndNextViewer ;
        HGLOBAL                  hGlobal ;
        HDC                      hdc ;
        PTSTR                    pGlobal ;
        PAINTSTRUCT              ps ;
        RECT                     rect ;
```

```
 switch (message)
 {
 case   WM_CREATE:
        hwndNextViewer = SetClipboardViewer (hwnd) ;
       return 0 ;

 case   WM_CHANGECBCHAIN:
        if ((HWND) wParam == hwndNextViewer)
                        hwndNextViewer = (HWND) lParam ;

        else if(hwndNextViewer)
                                SendMessage (hwndNextViewer,

return 0 ;
 case   WM_DRAWCLIPBOARD:
        if (hwndNextViewer)
                        SendMessage (hwndNextViewer, mess
```

```
                InvalidateRect (hwnd, NULL, TRUE) ;
                return 0 ;

          case   WM_PAINT:
                hdc = BeginPaint (hwnd, &ps) ;
               GetClientRect (hwnd, &rect) ;
                OpenClipboard (hwnd) ;

#ifdef UNICODE
                hGlobal = GetClipboardData (CF_UNICODETEXT) ;
#else
                hGlobal = GetClipboardData (CF_TEXT) ;
#endif
                if (hGlobal != NULL)
               {
                              pGlobal = (PTSTR) GlobalLock (hGlobal) ;
                              DrawText (hdc, pGlobal, -1, &rect, DT_EXPA
                              GlobalUnlock (hGlobal) ;
```

```
            }


            CloseClipboard () ;
            EndPaint (hwnd, &ps) ;
            return 0 ;


        case   WM_DESTROY:
            ChangeClipboardChain (hwnd, hwndNextViewer) ;
            PostQuitMessage (0) ;
            return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam)
}
```

CLIPVIEWWM_CREATEWM_CHANGECBCHAIN
WM_DRAWCLIPBOARDWM_DESTROYWM_PAINTCF_TEXT
GetClipboardDataCLIPVIEWDrawText

WindowsEnumClipboardFormatsGetClipboardFormatName
CF_OWNERDISPLAY

WM_PAINTCLIPBOARD	WM_VSCROLLCLIPBOARD

WM_SIZECLIPBOARD	WM_HSCROLLCLIPBOARD

GetClipboardOwner

WindowsGDI
off line

WindowsWindows

-
-
-

GDIWindowsStartDocEndDocStartPageEndPageGDI

metafile

WindowsGDI32.DRVWindows

CreateDCPrintDlgStartDocStartDocGDIGDI
Control

StartDocEndDocGDIStartPageEndPage

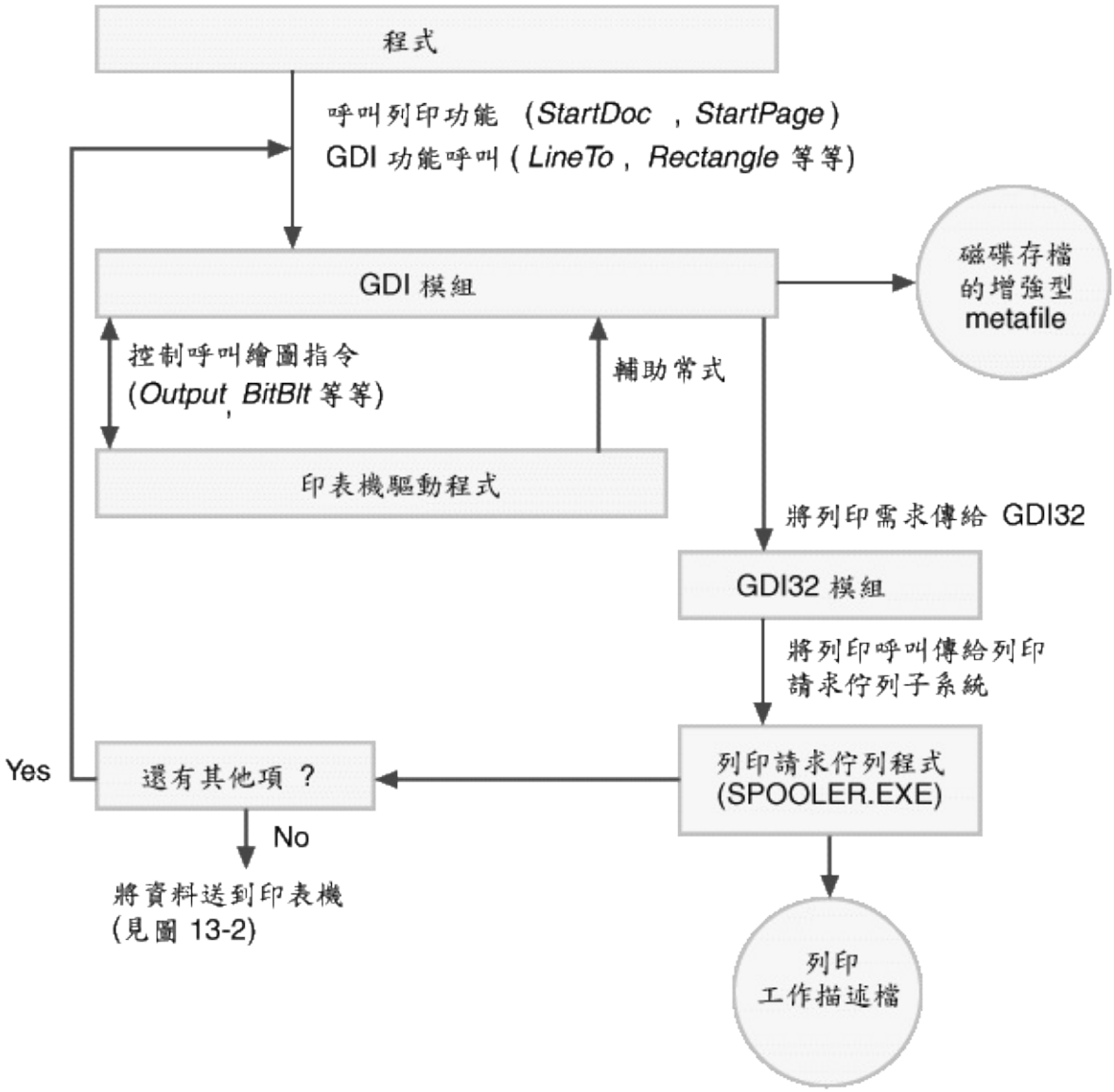
StartDocStartPageEllipseGDIGDImetafile~EMF
metafile.TMP

GDIEndPagemetafile6008.5×114

GDImetafileOutputmetafileGDI
metafilemetafile

GDIhelperPostScript

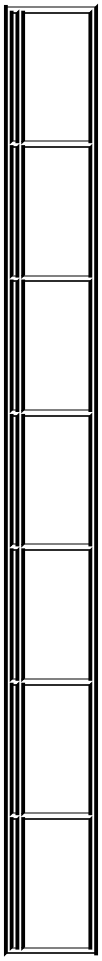
GDIGDI~SPL.TMPGDIEndDoc
13-1GDI



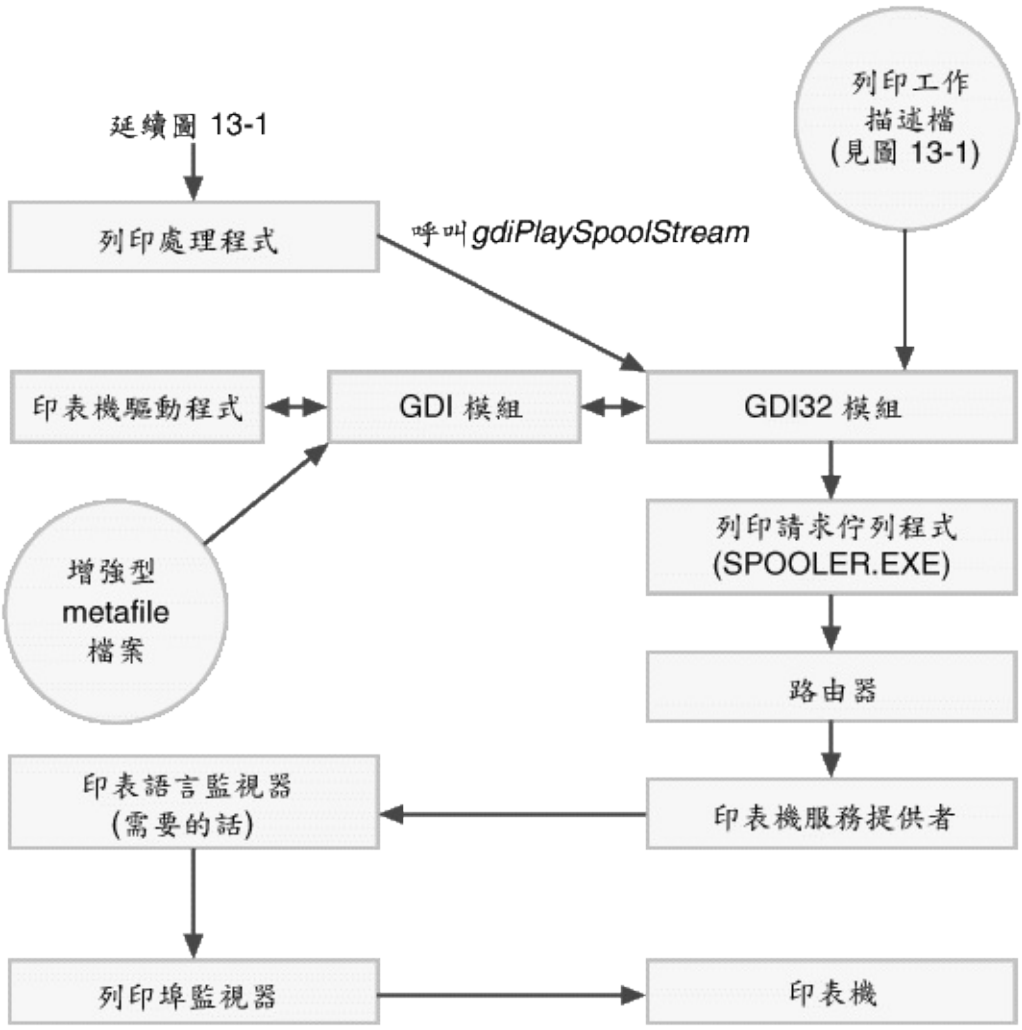
13-1 GDI

Windows13-1

13-1



WindowsGDIGDI13-2



13-2

GDI

Windows

WindowsWindows

WindowsGDIGDI

GDImetafilemetafilemetafileGDIGDI
metafileGDI

WindowsGDImetafile

StartDocStartPageGDI

PrintDlgNotepadCreateDC

```
hdc = CreateDC (TEXT ("DISPLAY"), NULL, NULL, NULL) ;
```

CreateDC

```
hdc = CreateDC (NULL, szDeviceName, NULL, pInitializationData
```

pInitializationDataNULLszDeviceNameWindows

Windows

EnumPrintersPRINTER_INFO_xx

Windows 98Windows NT13·

13-1 GETPRNDC

```
GETPRNDC.C

/*--------------------------------------------------------------------

  GETPRNDC.C -- GetPrinterDC function

--------------------------------------------------------------------*/
```

```
#include <windows.h>
HDC GetPrinterDC (void)
{
        DWORD                                   dwNeeded, dwR
        HDC                                     hdc ;
        PRINTER_INFO_4 *        pinfo4 ;
        PRINTER_INFO_5 *        pinfo5 ;

        if (GetVersion () & 0x80000000)     // Windows 98
        {
                EnumPrinters (PRINTER_ENUM_DEFAULT, NULL, 5
              0, &dwNeeded, &dwReturned) ;
                pinfo5 = malloc (dwNeeded) ;
               EnumPrinters (PRINTER_ENUM_DEFAULT, NULL, 5
              dwNeeded, &dwNeeded, &dwReturned) ;
                hdc = CreateDC (NULL, pinfo5->pPrinterName,
                free (pinfo5) ;
        }
        else
```

```
//Windows NT
  {
                EnumPrinters (PRINTER_ENUM_LOCAL, NULL, 4,
          0, &dwNeeded, &dwReturned) ;
                pinfo4 = malloc (dwNeeded) ;
                EnumPrinters (PRINTER_ENUM_LOCAL, NULL, 4,
          dwNeeded, &dwNeeded, &dwReturned) ;
                hdc = CreateDC (NULL, pinfo4->pPrinterName,
                free (pinfo4) ;
        }
        return hdc ;
}
```

GetVersionWindows 98Windows NTEnun 98PRINTER_INFO_5Windows NTPRINTER_INFO_4 EnumPrinters/Platform SDK/Graphics and Multimedia Services/GDI/Printi and Print Spooler/Printing and Print Spooler Reference/Printing and Print Spooler Functions/EnumPrinters

DEVCAPS

DEVCAPS1 GetDeviceCaps13-2

13-2 DEVCAPS2

```
DEVCAPS2.C
/*------------------------------------------------------------------------
  DEVCAPS2.C --        Displays Device Capability Information (Ve
                                                (c) Charles Petzold, 199
------------------------------------------------------------------------*/
#include <windows.h>
#include "resource.h"

LRESULT CALLBACK WndProc          (HWND, UINT, WPARAM, LPA
void DoBasicInfo                  (HDC, HDC, int, int) ;
void DoOtherInfo                  (HDC, HDC, int, int) ;
void DoBitCodedCaps                   (HDC, HDC, int, int, int) ;

typedef struct
{
          int                     iMask ;
          TCHAR *             szDesc ;
}
BITS ;
```

```
#define IDM_DEVMODE     1000

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                          PSTR szCmdLine, int iCr
{
          static TCHAR        szAppName[] = TEXT ("DevCaps2") ;
          HWND                                hwnd ;
          MSG                                 msg ;
          WNDCLASS                            wndclass ;

          wndclass.style                      = CS_HREDRAW | CS_V
          wndclass.lpfnWndProc                      = WndProc ;
          wndclass.cbClsExtra                       = 0 ;
          wndclass.cbWndExtra                       = 0 ;
          wndclass.hInstance                        = hInstance ;
          wndclass.hIcon                            = LoadIcon (NULL, IDI_
          wndclass.hCursor                          = LoadCursor (NULL,
          wndclass.hbrBackground              = (HBRUSH) GetStockO
          wndclass.lpszMenuName               = szAppName ;
          wndclass.lpszClassName              = szAppName ;
```

```
if (!RegisterClass (&wndclass))
{
     MessageBox (  NULL, TEXT ("This program requires W
                                          szAppName, MB_IC
     return 0 ;
}

hwnd = CreateWindow (szAppName, NULL,
           WS_OVERLAPPEDWINDOW,
           CW_USEDEFAULT, CW_USEDEFAULT,
           CW_USEDEFAULT, CW_USEDEFAULT,
          NULL, NULL, hInstance, NULL) ;

ShowWindow (hwnd, iCmdShow) ;
UpdateWindow (hwnd) ;

while (GetMessage (&msg, NULL, 0, 0))
```

```
        {
                TranslateMessage (&msg) ;
                DispatchMessage (&msg) ;
        }
        return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
  static TCHAR                  szDevice[32], szWindowText[64]
   static int                   cxChar, cyChar,nCurrentDevice
     nCurrentInfo       = IDM_BASIC ;
        static DWORD                    dwNeeded, dwReturned ;
        static PRINTER_INFO_4 * pinfo4 ;
        static PRINTER_INFO_5 * pinfo5 ;
        DWORD                i ;
        HDC                 hdc, hdcInfo ;
        HMENU               hMenu ;
        HANDLE              hPrint ;
```

```
        PAINTSTRUCT         ps ;
        TEXTMETRIC          tm ;

        switch (message)
        {
        case   WM_CREATE :
                    hdc =  GetDC (hwnd) ;
              SelectObject (hdc, GetStockObject (SYSTEM_FIXED_F
              GetTextMetrics (hdc, &tm) ;
                    cxChar = tm.tmAveCharWidth ;
                    cyChar = tm.tmHeight + tm.tmExternalLeading
                    ReleaseDC (hwnd, hdc) ;
// fall through
        case   WM_SETTINGCHANGE:
                    hMenu = GetSubMenu (GetMenu (hwnd), 0) ;

                    while (GetMenuItemCount (hMenu) > 1)
                          DeleteMenu (hMenu, 1, MF_BYPOSITION) ;
```

```
                    // Get a list of all local and remote printers
                    //
                   // First, find out how large an array we need
                    //   call will fail, leaving the required size in
                    //
                   // Next, allocate space for the info array and
                    //
                   // Put the printer names on the menu

if (GetVersion () & 0x80000000)                                    //
    {
                    EnumPrinters (PRINTER_ENUM_LOCAL, NUL
        0, &dwNeeded, &dwReturned) ;

                   pinfo5 = malloc (dwNeeded) ;

                    EnumPrinters (PRINTER_ENUM_LOCAL, NUL
        dwNeeded, &dwNeeded, &dwReturned) ;

                    for (i = 0 ; i < dwReturned ; i++)
                    {
```

```
                        AppendMenu (hMenu, (i+1) % 16 ? 0 : M
                                                    pinfo5[i].pP
                    }
                free (pinfo5) ;
        }
            else
// Windows NT
        {
                EnumPrinters (PRINTER_ENUM_LOCAL, NULL
            0, &dwNeeded, &dwReturned) ;
                pinfo4 = malloc (dwNeeded) ;
                EnumPrinters (PRINTER_ENUM_LOCAL, NUL
            dwNeeded, &dwNeeded, &dwReturned) ;
                for (i = 0 ; i < dwReturned ; i++)
                    {
                 AppendMenu (hMenu, (i+1) % 16 ? 0 : MF_
                    pinfo4[i].pPrinterName) ;
          }
```

```
                        free (pinfo4) ;
                }


                AppendMenu (hMenu, MF_SEPARATOR, 0, NULL) ;
                AppendMenu (hMenu, 0, IDM_DEVMODE, TEXT ("Prop


                wParam = IDM_SCREEN ;
// fall through
        case   WM_COMMAND :
                        hMenu = GetMenu (hwnd) ;


                        if (   LOWORD (wParam) == IDM_SCREEN ||     //
                                        LOWORD (wParam) < IDM_DEVMODE)
                        {
                        CheckMenuItem (hMenu, nCurrentDevice, MF_U
                        nCurrentDevice = LOWORD (wParam) ;
                        CheckMenuItem (hMenu, nCurrentDevice, MF_C
                        }
                        else if (LOWORD (wParam) == IDM_DEVMODE)
```

```
                {
                        GetMenuString (hMenu, nCurrent
                    sizeof (szDevice) / sizeof (TCHAR), MF_BYC


                        if (OpenPrinter (szDevice, &hPrint
                        {
                        PrinterProperties (hwnd, hPrint) ;
                        ClosePrinter (hPrint) ;
                        }
                }
                else
// info menu items
        {
                        CheckMenuItem (hMenu, nCurrentInfo
                        nCurrentInfo = LOWORD (wParam) ;
                        CheckMenuItem (hMenu, nCurrentInfo
                }
                InvalidateRect (hwnd, NULL, TRUE) ;
```

```
                return 0 ;

        case   WM_INITMENUPOPUP :
                if (lParam == 0)
                        EnableMenuItem (GetMenu (hwnd
                                nCurrentDevice == IDM_SCR
                return 0 ;

        case   WM_PAINT :
                lstrcpy (szWindowText, TEXT ("Device Capabiliti

                if (nCurrentDevice == IDM_SCREEN)
                {
                        lstrcpy (szDevice, TEXT ("DISPLAY")) ;
                        hdcInfo = CreateIC (szDevice, NULL, N
                }
                else
                {
                        hMenu = GetMenu (hwnd) ;
```

```
                GetMenuString (hMenu, nCurrentDevic
 sizeof (szDevice), MF_BYCOMMAND) ;
               hdcInfo = CreateIC (NULL, szDevice, NU
}

lstrcat (szWindowText, szDevice) ;
SetWindowText (hwnd, szWindowText) ;

hdc = BeginPaint (hwnd, &ps) ;
SelectObject (hdc, GetStockObject (SYSTEM_FIX

if (hdcInfo)
{
               switch (nCurrentInfo)
                     {
               case   IDM_BASIC :
DoBasicInfo (hdc, hdcInfo, cxChar, cyChar) ;
                              break ;
```

```
                case   IDM_OTHER :
            DoOtherInfo (hdc, hdcInfo, cxChar, cyChar) ;
                                break ;

                case   IDM_CURVE :
                case   IDM_LINE :
                case   IDM_POLY :
                case   IDM_TEXT :
            DoBitCodedCaps (hdc, hdcInfo, cxChar, cyChar,
                        nCurrentInfo - IDM_CURVE) ;
                                break ;
                          }
                DeleteDC (hdcInfo) ;
        }

        EndPaint (hwnd, &ps) ;
        return 0 ;
```

```
        case   WM_DESTROY :
                    PostQuitMessage (0) ;
                    return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam)
}

void DoBasicInfo (HDC hdc, HDC hdcInfo, int cxChar, int cyChar)
{
  static struct
        {
             int     nIndex ;
             TCHAR * szDesc ;
  }
        info[] =
        {
            HORZSIZE,                TEXT ("HORZSIZE
             VERTSIZE,               TEXT ("VERTSIZE                 H
```

```
            HORZRES,                TEXT ("HORZRES
            VERTRES,                TEXT ("VERTRES              H
            BITSPIXEL,              TEXT ("BITSPIXEL            C
            PLANES,                 TEXT ("PLANES
            NUMBRUSHES,             TEXT ("NUMBRUSHES
            NUMPENS,                TEXT ("NUMPENS
            NUMMARKERS,             TEXT ("NUMMARKERS
            NUMFONTS,               TEXT   ("NUMFONTS
            NUMCOLORS,              TEXT   ("NUMCOLORS
            PDEVICESIZE, TEXT("PDEVICESIZESize of device struc
ASPECTX,      TEXT("ASPECTX Relative width of pixel:"),
ASPECTY,      TEXT("ASPECTY Relative height of pixel:"),
ASPECTXY,     TEXT("ASPECTXY Relative diagonal of pixel:"),
LOGPIXELSX,   TEXT("LOGPIXELSX Horizontal dots per inch:"),
LOGPIXELSY,   TEXT("LOGPIXELSY Vertical dots per inch:"),
SIZEPALETTE,  TEXT("SIZEPALETTE Number of palette entries:"),
NUMRESERVED,  TEXT("NUMRESERVED Reserved palette entries
COLORRES,     TEXT("COLORRES Actual color resolution:"),
PHYSICALWIDTH, TEXT("PHYSICALWIDTH Printer page pixel width
```

```
PHYSICALHEIGHT,TEXT("PHYSICALHEIGHT Printer page pixel heig
PHYSICALOFFSETX,TEXT("PHYSICALOFFSETX Printer page x offse
PHYSICALOFFSETY,TEXT("PHYSICALOFFSETY Printer page y offse
  } ;
   int   i ;
  TCHAR szBuffer[80] ;

          for (i = 0 ; i < sizeof (info) / sizeof (info[0]) ; i++)
            TextOut (hdc, cxChar, (i + 1) * cyChar, szBuffer,
             wsprintf (szBuffer, TEXT ("%-45s%8d"), info[i].szDesc,
                 GetDeviceCaps (hdcInfo, info[i].nIndex))) ;
}

void DoOtherInfo (HDC hdc, HDC hdcInfo, int cxChar, int cyChar)
{
          static BITS clip[] =
          {
       CP_RECTANGLE, TEXT ("CP_RECTANGLE  Can Clip To Rectan
```

```
      } ;


      static BITS raster[] =
{
  RC_BITBLT,    TEXT ("RC_BITBLT  Capable of simple BitBlt:"),
      RC_BANDING,   TEXT ("RC_BANDING Requires banding su
  RC_SCALING,   TEXT ("RC_SCALING Requires scaling support
  RC_BITMAP64,  TEXT ("RC_BITMAP64  Supports bitmaps >64
  RC_GDI20_OUTPUT, TEXT ("RC_GDI20_OUTPUT Has 2.0 outp
  RC_DI_BITMAP, TEXT ("RC_DI_BITMAP  Supports DIB to mem
      RC_PALETTE,   TEXT ("RC_PALETTE      Supports a palette:
     RC_DIBTODEV,  TEXT ("RC_DIBTODEV Supports bitmap co
     RC_BIGFONT,   TEXT ("RC_BIGFONT  Supports fonts >64K:
      RC_STRETCHBLT,TEXT ("RC_STRETCHBLT Supports Stretch
      RC_FLOODFILL, TEXT ("RC_FLOODFILL  Supports FloodFill:
     RC_STRETCHDIB,TEXT ("RC_STRETCHDIB Supports Stretch
      } ;


      static TCHAR * szTech[]=      {      TEXT ("DT_PLOTTER (Ve
```

```
                    TEXT ("DT_RASDISPLAY (Raster display)"),
                     TEXT ("DT_RASPRINTER (Raster printer)"),
                     TEXT ("DT_RASCAMERA (Raster camera)"),
                     TEXT ("DT_CHARSTREAM (Character stream)"),
                    TEXT ("DT_METAFILE (Metafile)"),
                     TEXT ("DT_DISPFILE (Display file)") } ;
int                              i ;
 TCHAR                               szBuffer[80] ;


        TextOut (hdc, cxChar, cyChar, szBuffer,
      wsprintf (szBuffer, TEXT ("%-24s%04XH"), TEXT ("DRIVERV
          GetDeviceCaps (hdcInfo, DRIVERVERSION))) ;
TextOut (hdc, cxChar, 2 * cyChar, szBuffer,
                     wsprintf (szBuffer, TEXT ("%-24s%-40s"), TEXT (
                                  szTech[GetDeviceCaps (hdcInfo, TECH
 TextOut (hdc, cxChar, 4 * cyChar, szBuffer,
                     wsprintf (szBuffer, TEXT ("CLIPCAPS (Clipping ca
        for (i = 0 ; i < sizeof (clip) / sizeof (clip[0]) ; i++)
```

```
                    TextOut (hdc, 9 * cxChar, (i + 6) * cyChar, szBuf
                                        wsprintf (szBuffer, TEXT ("%-45s %
                                                GetDeviceCaps (hdcInfo, CLI
                                         TEXT ("Yes") : TEXT ("No"))) ;
          TextOut (hdc, cxChar, 8 * cyChar, szBuffer,
                wsprintf (szBuffer, TEXT ("RASTERCAPS (Raster capa
   for (i = 0 ; i < sizeof (raster) / sizeof (raster[0]) ; i++)
          TextOut (hdc, 9 * cxChar, (i + 10) * cyChar, szBuffer,
                                        wsprintf (szBuffer, TEXT ("%-45s %
                                                GetDeviceCaps (hdcInfo, RA
                              TEXT ("Yes") : TEXT ("No"))) ;
}

void DoBitCodedCaps (      HDC hdc, HDC hdcInfo, int cxChar, int
{
          static BITS curves[] =
          {
                CC_CIRCLES,    TEXT ("CC_CIRCLES    Can do circles:"
                CC_PIE,        TEXT ("CC_PIE        Can do pie wedges:")
```

```
        CC_CHORD,      TEXT ("CC_CHORD      Can do chord a
        CC_ELLIPSES,  TEXT ("CC_ELLIPSES   Can do ellipses:
        CC_WIDE,       TEXT ("CC_WIDE       Can do wide bord
        CC_STYLED,     TEXT ("CC_STYLED     Can do styled b
       CC_WIDESTYLED, TEXT   ("CC_WIDESTYLED Can do w
        CC_INTERIORS,  TEXT ("CC_INTERIORS  Can do interi
    } ;

    static BITS lines[] =
    {
        LC_POLYLINE,   TEXT   ("LC_POLYLINE Can do polyline
        LC_MARKER,     TEXT   ("LC_MARKER Can do markers
        LC_POLYMARKER, TEXT   ("LC_POLYMARKER Can do p
        LC_WIDE,       TEXT   ("LC_WIDE Can do wide lines:"),
        LC_STYLED,     TEXT   ("LC_STYLED Can do styled line
        LC_WIDESTYLED, TEXT   ("LC_WIDESTYLED       Can d
        LC_INTERIORS,  TEXT ("LC_INTERIORS  Can do interi
    } ;
```

```
        static BITS poly[] =
        {
             PC_POLYGON,
                       TEXT ("PC_POLYGON     Can do alternate fill
             PC_RECTANGLE,        TEXT   ("PC_RECTANGLE Can do
        PC_WINDPOLYGON,
                        TEXT ("PC_WINDPOLYGON Can do winding
             PC_SCANLINE,         TEXT ("PC_SCANLINE    Can do s
             PC_WIDE,             TEXT ("PC_WIDE        Can do wide
             PC_STYLED,           TEXT ("PC_STYLED      Can do styl
             PC_WIDESTYLED,
                       TEXT ("PC_WIDESTYLED  Can do wide and s
            PC_INTERIORS,         TEXT ("PC_INTERIORS   Can do in
} ;


        static BITS text[] =
        {
             TC_OP_CHARACTER, TEXT ("TC_OP_CHARACTER      C
```

```
        TC_OP_STROKE,    TEXT ("TC_OP_STROKE  Can do str
        TC_CP_STROKE,    TEXT ("TC_CP_STROKE  Can do str
        TC_CR_90,        TEXT ("TC_CP_90      Can do 90 degre
       TC_CR_ANY,        TEXT ("TC_CR_ANY     Can do any cha
        TC_SF_X_YINDEP,  TEXT ("TC_SF_X_YINDEP  Can do sc
        TC_SA_DOUBLE,    EXT ("TC_SA_DOUBLE    Can do do
        TC_SA_INTEGER,   TEXT ("TC_SA_INTEGER   Can do in
        TC_SA_CONTIN,    TEXT ("TC_SA_CONTIN  Can do any
        TC_EA_DOUBLE,    TEXT ("TC_EA_DOUBLE   Can do dc
        TC_IA_ABLE,      TEXT ("TC_IA_ABLE     Can do italicizi
        TC_UA_ABLE,      TEXT ("TC_UA_ABLE     Can do under
        TC_SO_ABLE,      TEXT ("TC_SO_ABLE     Can do strike
      TC_RA_ABLE,      TEXT ("TC_RA_ABLE     Can do raster f
        TC_VA_ABLE,      TEXT ("TC_VA_ABLE     Can do vector
    } ;


    static struct
    {
        int                     iIndex ;
```

```
            TCHAR *             szTitle ;
            BITS                    (*pbits)[] ;
            int                     iSize ;
}
        bitinfo[] =
        {
            CURVECAPS,    TEXT ("CURVCAPS (Curve Capabilities
                        (BITS (*)[]) curves, sizeof (curves) / sizeof (
            LINECAPS,     TEXT ("LINECAPS (Line Capabilities)"),
                        (BITS (*)[]) lines, sizeof (lines) / sizeof (line
            POLYGONALCAPS, TEXT ("POLYGONALCAPS (Polygona
                        (BITS (*)[]) poly, sizeof (poly) / sizeof (poly[
            TEXTCAPS,     TEXT ("TEXTCAPS (Text Capabilities)"),
                        (BITS (*)[]) text, sizeof (text) / sizeof (text[(
 } ;

        static TCHAR szBuffer[80] ;
        BITS                (*pbits)[] = bitinfo[iType].pbits ;
```

```
        int                 i, iDevCaps = GetDeviceCaps (hdcInfo, bitir


        TextOut (hdc, cxChar, cyChar, bitinfo[iType].szTitle,
                                      lstrlen (bitinfo[iType].szTitle)) ;
        for (i = 0 ; i < bitinfo[iType].iSize ; i++)
        extOut (hdc, cxChar, (i + 3) * cyChar, szBuffer,
         wsprintf (szBuffer, TEXT ("%-55s %3s"), (*pbits)[i].szDesc
         iDevCaps & (*pbits)[i].iMask ? TEXT ("Yes") : TEXT ("No"))
}
```

DEVCAPS2.RC

```
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
//////////////////////////////////////////////////////////////////////////////
// Menu
DEVCAPS2 MENU DISCARDABLE
BEGIN
   POPUP "&Device"
```

```
  BEGIN
                    MENUITEM "&Screen",IDM_SCREEN, CHECKED
  END
  POPUP "&Capabilities"
  BEGIN
              MENUITEM "&Basic Information",IDM_BASIC
                    MENUITEM "&Other Information",IDM_OTHER
                    MENUITEM "&Curve Capabilities",IDM_CURVE
                    MENUITEM "&Line Capabilities",IDM_LINE
                    MENUITEM "&Polygonal Capabilities",IDM_POLY
                    MENUITEM "&Text Capabilities",IDM_TEXT
  END
END
```

RESOURCE.H

```
// Microsoft Developer Studio generated include file.
// Used by DevCaps2.rc
#define IDM_SCREEN      40001
```

```
#define IDM_BASIC        40002

#define IDM_OTHER        40003

#define IDM_CURVE        40004

#define IDM_LINE         40005

#define IDM_POLY         40006

#define IDM_TEXT         40007
```

DEVCAPS2DEVCAPS2

PrinterProperties

DEVCAPS2DeviceProperties
portraitlandscape
DEVCAPS2GetDeviceCapsProperties

ExtDeviceModeWindows

WindowsPrintDlgPropertiesPrintDlg

ExtDeviceModeExtDeveModePropSheetDEVCAPS2
PrinterProperties

PrinterPropertiesOpenPrinterPrinterPropertiesClosePrinter
DEVCAPS2

DeviceszDevice

```
GetMenuString (    hMenu, nCurrentDevice, szDevice,

                                    sizeof (szDevice) / sizeof (TCHAR)
```

OpenPrinterPrinterPropertiesClosePrinter

```
if (OpenPrinter (szDevice, &hPrint, NULL))
{
        PrinterProperties (hwnd, hPrint) ;
        ClosePrinter (hPrint) ;
}
```

BitBlt

GetDeviceCaps

GDIGDI

RASTERCAPSGetDeviceCapsRC_BITBLT
GDICreateCompatibleDCCreateCompatibleBitmapPatBltBitBlt
StretchBltGrayStringDrawIconSetPixelGetPixelFloodFill
ExtFloodFillFillRgnFrameRgnInvertRgnPaintRgnFillRectFrameRect
InvertRectGDI

13-3FORMFEED

13-3　FORMFEED

FORMFEED.C

```
/*--------------------------------------------------------------------
  FORMFEED.C -- Advances printer to next page
```

```
                                        (c) Charles Petzold, 1998
----------------------------------------------------------------------*/
#include <windows.h>
HDC GetPrinterDC (void) ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                    LPSTR lpszCmdLine, int iCmdShow)
{
     static DOCINFO di = { sizeof (DOCINFO), TEXT ("FormFee
     HDC                    hdcPrint = GetPrinterDC ()

     if (hdcPrint != NULL)
     {
    if (StartDoc (hdcPrint, &di) > 0)
                    if (StartPage (hdcPrint) > 0 && EndPag
                                EndDoc (hdcPrint) ;
          DeleteDC (hdcPrint) ;
    }
     return 0 ;
}
```

13-1GETPRNDC.C

FORMFEEDStartDocStartDoc

```
if (StartDoc (hdcPrint, &di) > 0)
```

StartDocDOCINFOFormFeedDocument

StartDocFORMFEEDStartPageEndPage

```
if (StartPage (hdcPrint) > 0 && EndPage (hdcPrint) > 0)
```

```
EndDoc (hdcPrint) ;
```

EndDocGDIGetLastError

MS-DOSASCII12CopenwriteASCII12
ASCII1212PostScript12showpage

WindowsWindows

WindowsFORMFEEDGDIFORMFEEDPRINT1
PRINT2PRINT3GETPRNDC.CPRINT.C13-4

13-4　PRINT

```
PRINT.C
```

```
/*--------------------------------------------------------------------
  PRINT.C -- Common routines for Print1, Print2, and Print3
--------------------------------------------------------------------*/
#include <windows.h>
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
BOOL PrintMyPage (HWND) ;

extern HINSTANCE   hInst ;
extern TCHAR                szAppName[] ;
extern TCHAR                szCaption[] ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                    PSTR szCmdLine, int iCmdShow)
{
          HWND              hwnd ;
          MSG               msg ;
          WNDCLASS          wndclass ;

          wndclass.style                    = CS_HREDRAW | CS_V
          wndclass.lpfnWndProc                   = WndProc ;
```

```
        wndclass.cbClsExtra                    = 0 ;
        wndclass.cbWndExtra                    = 0 ;
        wndclass.hInstance                     = hInstance ;
        wndclass.hIcon                         = LoadIcon (NULL, IDI_
        wndclass.hCursor                       = LoadCursor (NULL,
        wndclass.hbrBackground                 = (HBRUSH) GetStockO
        wndclass.lpszMenuName                  = NULL ;
        wndclass.lpszClassName                 = szAppName ;

        if (!RegisterClass (&wndclass))
{
             MessageBox (  NULL, TEXT ("This program requires W
                                    szAppName, MB_ICONERROR) ;
             return 0 ;
        }

        hInst = hInstance ;
        hwnd = CreateWindow (szAppName, szCaption,
```

```
                    WS_OVERLAPPEDWINDOW,
                    CW_USEDEFAULT, CW_USEDEFAULT,
                    CW_USEDEFAULT, CW_USEDEFAULT,
                   NULL, NULL, hInstance, NULL) ;

          ShowWindow (hwnd, iCmdShow) ;
          UpdateWindow (hwnd) ;

          while (GetMessage (&msg, NULL, 0, 0))
          {
                    TranslateMessage (&msg) ;
                    DispatchMessage (&msg) ;
          }
          return msg.wParam ;
}

void PageGDICalls (HDC hdcPrn, int cxPage, int cyPage)
{
          static TCHAR szTextStr[] = TEXT ("Hello, Printer!") ;
```

```
        Rectangle (hdcPrn, 0, 0, cxPage, cyPage) ;
        MoveToEx (hdcPrn, 0, 0, NULL) ;
        LineTo   (hdcPrn, cxPage, cyPage) ;
        MoveToEx (hdcPrn, cxPage, 0, NULL) ;
        LineTo   (hdcPrn, 0, cyPage) ;

        SaveDC (hdcPrn) ;

        SetMapMode                (hdcPrn, MM_ISOTROPIC) ;
        SetWindowExtEx            (hdcPrn, 1000, 1000, NULL) ;
      SetViewportExtEx            (hdcPrn, cxPage / 2, -cyPage / 2,
        SetViewportOrgEx          (hdcPrn, cxPage / 2,  cyPage /

        Ellipse (hdcPrn, -500, 500, 500, -500) ;
        SetTextAlign (hdcPrn, TA_BASELINE | TA_CENTER) ;
        TextOut (hdcPrn, 0, 0, szTextStr, lstrlen (szTextStr)) ;

        RestoreDC (hdcPrn, -1) ;
}
```

```
LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
        static int          cxClient, cyClient ;
        HDC                 hdc ;
        HMENU                 hMenu ;
        PAINTSTRUCT             ps ;

        switch (message)
        {
        case   WM_CREATE:
                    hMenu = GetSystemMenu (hwnd, FALSE) ;
                    AppendMenu (hMenu, MF_SEPARATOR, 0, NULL)
                    AppendMenu (hMenu, 0, 1, TEXT ("&Print")) ;
                    return 0 ;

        case   WM_SIZE:
                    cxClient = LOWORD (lParam) ;
                    cyClient = HIWORD (lParam) ;
```

```
            return 0 ;

        case   WM_SYSCOMMAND:
                if (wParam == 1)
            {
                        if (!PrintMyPage (hwnd))
                                MessageBox (hwnd, TEXT ("Could
                        szAppName, MB_OK | MB_ICONEXCLAM
                         return 0 ;
            }
            break ;

        case   WM_PAINT :
            hdc = BeginPaint (hwnd, &ps) ;

                PageGDICalls (hdc, cxClient, cyClient) ;

                EndPaint (hwnd, &ps) ;
                return 0 ;
```

```
        case   WM_DESTROY :
                    PostQuitMessage (0) ;
                    return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam)
}
```

PRINT.CWinMainWndProcPageGDICallsPageGDICalls
Hello, Printer!

WM_CREATEWndProcPrintPrintMyPagePrintMyPage
TRUEFALSEPrintMyPageFALSEWndProc

PRINT113-5PrintGDI

13-5 PRINT1

```
PRINT1.C
/*------------------------------------------------------------------
  PRINT1.C -- Bare Bones Printing
                                          (c) Charles Petzold, 1998
------------------------------------------------------------------*/
```

```
#include <windows.h>
HDC        GetPrinterDC (void) ;                    // in GETPRNDC.C
void       PageGDICalls (HDC, int, int) ;           // in PRINT.C

HINSTANCE hInst ;
TCHAR             szAppName[] = TEXT ("Print1") ;
TCHAR             szCaption[] = TEXT ("Print Program 1") ;

BOOL PrintMyPage (HWND hwnd)
{
          static DOCINFO di = { sizeof (DOCINFO), TEXT ("Print1: Pr
    BOOL                    bSuccess = TRUE ;
          HDC               hdcPrn ;
          int               xPage, yPage ;

          if     (NULL == (hdcPrn = GetPrinterDC ()))
                    return FALSE ;
   xPage = GetDeviceCaps (hdcPrn, HORZRES) ;
  yPage = GetDeviceCaps (hdcPrn, VERTRES) ;
```

```
  if (StartDoc (hdcPrn, &di) > 0)
        {
                if (StartPage (hdcPrn) > 0)
                 {
                        PageGDICalls (hdcPrn, xPage, yPage) ;

                        if (EndPage (hdcPrn) > 0)
                                EndDoc (hdcPrn) ;
                        else
                              bSuccess = FALSE ;
                }
        }
        else
             bSuccess = FALSE ;

        DeleteDC (hdcPrn) ;
        return bSuccess ;
}
```

```

```

PRINT1.CPrintMyPageFALSEWndProcGetDeviceCaps

```
xPage = GetDeviceCaps (hdcPrn, HORZRES) ;

yPage = GetDeviceCaps (hdcPrn, VERTRES) ;
```

PRINT1StartPageEndPagePageGDICallsPRINT1PrintMyPage
FORMFEEDStartDocStartPageEndPagePRINT1EndDoc

537537

SetAbortProcWindowsGDI

AbortProc

```
BOOL CALLBACK AbortProc (HDC hdcPrn, int iCode)

{

   //

}
```

SetAbortProc

```
SetAbortProc (hdcPrn, AbortProc) ;
```

StartDoc

EndPagemetafileGDIhdcPrniCode0GDIiCode

SP_OUTOFDISK

AbortProcTRUEFALSE

```
BOOL CALLBACK AbortProc (HDC hdcPrn, int iCode)
{
        MSG   msg ;
        while (PeekMessage (&msg, NULL, 0, 0, PM_REMOVE))
        {
             TranslateMessage (&msg) ;
             DispatchMessage (&msg) ;
        }
        return TRUE ;
}
```

PeekMessageGetMessageRANDRECTPeekMessage
PeekMessage

PeekMessageTRUEAbortProcPeekMessageTRUEPeekMessage
TranslateMessageDispatchMessagePeekMessageFALSEAbortProc
Windows

WindowsAbortProc

EndPageEndPageGDIGDImetafileGDIEndPageGDI
metafileGDIGDI

EndPageGDIiCode0GDIiCodeSP_OUTOFDISK

PeekMessage

PeekMessageFALSEGDITRUEGDIEndPage

GDICancelPRINT2PRINT3Cancel

```
BOOL CALLBACK AbortProc (HDC hdcPrn, int iCode)
{
        MSG  msg ;
        while (PeekMessage (&msg, NULL, 0, 0, PM_REMOVE))
        {
             TranslateMessage (&msg) ;
             DispatchMessage (&msg) ;
        }
        return TRUE ;
}
```

Windows

```
SetAbortProc (hdcPrn, AbortProc) ;
```

StartDoc

AbortProcPeekMessageAbortProcAbortProc
Print

```
EnableWindow (hwnd, FALSE) ;
```

```
EnableWindow (hwnd, TRUE) ;
```

AbortProcTranslateMessageDispatchMessageTranslateMessage
DispatchMessageWM_PAINTWM_PAINTBeginPaintEndPaint
PeekMessageFALSE

13-6PRINT2PRINT1AbortProcEnableWindow

13-6　PRINT2

```
PRINT2.C

/*--------------------------------------------------------------

  PRINT2.C --   Printing with Abort Procedure

                                        (c) Charles Petzold, 1998

--------------------------------------------------------------*/

#include <windows.h>

HDC GetPrinterDC (void) ;                       // in GETPRNDC.C
```

```
void PageGDICalls (HDC, int, int) ;              // in PRINT.C

HINSTANCE hInst ;
TCHAR            szAppName[] = TEXT ("Print2") ;
TCHAR            szCaption[] = TEXT ("Print Program 2 (Abort Proc

BOOL CALLBACK AbortProc (HDC hdcPrn, int iCode)
{
          MSG msg ;
          while (PeekMessage (&msg, NULL, 0, 0, PM_REMOVE)
     {
               TranslateMessage (&msg) ;
               DispatchMessage (&msg) ;
     }
     return TRUE ;
}

BOOL PrintMyPage (HWND hwnd)
{
     static DOCINFO di = { sizeof (DOCINFO), TEXT ("Print2: Pr
```

```
BOOL                    bSuccess = TRUE ;
HDC                     hdcPrn ;
short                   xPage, yPage ;

if (NULL == (hdcPrn = GetPrinterDC ()))
          return FALSE ;
xPage = GetDeviceCaps (hdcPrn, HORZRES) ;
yPage = GetDeviceCaps (hdcPrn, VERTRES) ;

EnableWindow (hwnd, FALSE) ;
SetAbortProc (hdcPrn, AbortProc) ;
if (StartDoc (hdcPrn, &di) > 0)
{
          if (StartPage (hdcPrn) > 0)
          {
                    PageGDICalls (hdcPrn, xPage, yPage) ;
                    if (EndPage (hdcPrn) > 0)
                              EndDoc (hdcPrn) ;
```

```
                                else
                                        bSuccess = FALSE ;
                        }
     }
          else
                        bSuccess = FALSE ;
          EnableWindow (hwnd, TRUE) ;
          DeleteDC (hdcPrn) ;
          return bSuccess ;
}
```

PRINT2PrintMyPagePRINT2

WindowsCancelGDI

AbortProcPrintDlgProc
Windows

AbortProcPeekMessagePrintDlgProcWM_COMMAND
CancelCancelbUserAbortTRUEAbortProcbUserAbort
AbortProcTRUEFALSEPRINT2TRUECancelFALSE13-
7PRINT3

13-7　PRINT3

```
```

```
PRINT3.C

/*------------------------------------------------------------

  PRINT3.C --   Printing with Dialog Box

                                          (c) Charles Petzold, 1998

------------------------------------------------------------*/

#include <windows.h>

HDC GetPrinterDC (void) ;                       // in GETPRNDC.C

voidPageGDICalls (HDC, int, int) ;              // in PRINT.C

HINSTANCE hInst ;

TCHAR           szAppName[] = TEXT ("Print3") ;

TCHAR           szCaption[] = TEXT ("Print Program 3 (Dialog Box

BOOL bUserAbort ;

HWND hDlgPrint ;

BOOL CALLBACK PrintDlgProc (HWND hDlg, UINT message,

                    WPARAM wParam, LPARAM lParam)

{

        switch (message)
```

```
        {
   case   WM_INITDIALOG:
                    SetWindowText (hDlg, szAppName) ;
                    EnableMenuItem (GetSystemMenu (hDlg, FALSE
                    return TRUE ;

        case   WM_COMMAND:
                    bUserAbort = TRUE ;
                    EnableWindow (GetParent (hDlg), TRUE) ;
                    DestroyWindow (hDlg) ;
                    hDlgPrint = NULL ;
                   return TRUE ;
        }
        return FALSE ;
}

BOOL CALLBACK AbortProc (HDC hdcPrn, int iCode)
{
        MSG msg ;
```

```
        while (!bUserAbort && PeekMessage (&msg, NULL, 0, 0, P
        {
             if (!hDlgPrint || !IsDialogMessage (hDlgPrint, &msg))
             {
                        TranslateMessage (&msg) ;
                        DispatchMessage (&msg) ;
             }
        }
  return !bUserAbort ;
}

BOOL PrintMyPage (HWND hwnd)
{
        static DOCINFO di = { sizeof (DOCINFO), TEXT ("Print3: Pr
        BOOL                    bSuccess = TRUE ;
        HDC                     hdcPrn ;
        int                     xPage, yPage ;

        if (NULL == (hdcPrn = GetPrinterDC ()))
```

```
                return FALSE ;

        xPage = GetDeviceCaps (hdcPrn, HORZRES) ;

        yPage = GetDeviceCaps (hdcPrn, VERTRES) ;


        EnableWindow (hwnd, FALSE) ;

        bUserAbort = FALSE ;

        hDlgPrint = CreateDialog (hInst, TEXT ("PrintDlgBox"),

                                                hwnd, PrintDlgProc)
SetAbortProc (hdcPrn, AbortProc) ;

        if (StartDoc (hdcPrn, &di) > 0)

        {

             if (StartPage (hdcPrn) > 0)

             {

                        PageGDICalls (hdcPrn, xPage, yPage) ;

                             if (EndPage (hdcPrn) > 0)

                                        EndDoc (hdcPrn) ;

                        else

                             bSuccess = FALSE ;

             }
```

```
        }
        else
                        bSuccess = FALSE ;
        if (!bUserAbort)
   {
                    EnableWindow (hwnd, TRUE) ;
                   DestroyWindow (hDlgPrint) ;
   }

  DeleteDC (hdcPrn) ;
   return bSuccess && !bUserAbort ;
}
```

```
PRINT.RC
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
//////////////////////////////////////////////////////////////////////////////
// Dialog
```

```
PRINTDLGBOX DIALOG DISCARDABLE  20, 20, 186, 63
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTIO
FONT 8, "MS Sans Serif"
BEGIN
   PUSHBUTTON                    "Cancel",IDCANCEL,67,42,50,14
   CTEXT                              "Cancel Printing",IDC_STA
END
```

PRINT3PRINT3CancelCancel
CancelGDI

PRINT3bUserAborthDlgPrintPrintMyPagebUserAbortFALSE
PRINT2AbortProcSetAbortProcPrintDlgProcCreateDialog
CreateDialoghDlgPrint

AbortProc

```
while (!bUserAbort && PeekMessage (&msg, NULL, 0, 0, PM_REM
{
         if (!hDlgPrint || !IsDialogMessage (hDlgPrint, &msg))
        {
               TranslateMessage (&msg) ;
               DispatchMessage (&msg) ;
```

```
        }

}

return !bUserAbort ;
```

bUserAbortFALSEPeekMessageIsDialogMessageAbortProc
bUserAbortbUserAbortFALSEAbortProcTRUEbUserAbort
TRUE

PrintDlgProcWM_INITDIALOGCloseCancel
PrintDlgProcWM_COMMAND

```
case      WM_COMMAND :

        bUserAbort = TRUE ;

        EnableWindow (GetParent (hDlg), TRUE) ;

        DestroyWindow (hDlg) ;

        hDlgPrint = NULL ;

        return TRUE ;
```

bUserAbortTRUEWindowshDlgPrintNULL
IsDialogMessage

AbortProcPeekMessageIsDialogMessageGDIEndPage
AbortProcGDIAbortProcFALSEEndPagePrintMyPage
PrintMyPageEndDocGDIEndPage

PrintMyPage

```
if (!bUserAbort)
```

```
{
   EnableWindow (hwnd, TRUE) ;
   DestroyWindow (hDlgPrint) ;
}
```

bUserAbortbSuccessPrintMyPageANDWndProc

```
return bSuccess && !bUserAbort ;
```

POPPAD

POPPADPOPPAD POPPAD1

13-8 POPPRNT

```
POPPRNT.C
/*------------------------------------------------------------------
  POPPRNT.C -- Popup Editor Printing Functions
------------------------------------------------------------------*/
#include <windows.h>
#include <commdlg.h>
#include "resource.h"
```

```
BOOL bUserAbort ;
HWND hDlgPrint ;

BOOL CALLBACK PrintDlgProc (       HWND hDlg, UINT msg, WPAR
{
        switch (msg)
        {
        case   WM_INITDIALOG :
                   EnableMenuItem (GetSystemMenu (hDlg, FALSE
                   return TRUE ;

        case   WM_COMMAND :
                   bUserAbort = TRUE ;
                   EnableWindow (GetParent (hDlg), TRUE) ;
                   DestroyWindow (hDlg) ;
                   hDlgPrint = NULL ;
                   return TRUE ;
        }
        return FALSE ;
```

```
}

BOOL CALLBACK AbortProc (HDC hPrinterDC, int iCode)

{

        MSG msg ;

        while (!bUserAbort && PeekMessage (&msg, NULL, 0, 0, P

        {

             if (!hDlgPrint || !IsDialogMessage (hDlgPrint, &msg))

             {

                        TranslateMessage (&msg) ;

                        DispatchMessage (&msg) ;

             }

        }

        return !bUserAbort ;

}

BOOL PopPrntPrintFile (HINSTANCE hInst, HWND hwnd, HWND hw

                                                    PTSTR szTitleI

{

        static DOCINFO       di = { sizeof (DOCINFO) } ;
```

```
static PRINTDLG       pd ;
BOOL                  bSuccess ;
int                   yChar, iCharsPerLine, iLinesPerPage, iTotalL
                      iTotalPages, iPage, iLine, iLineNum ;
PTSTR                 pstrBuffer ;
TCHAR                 szJobName [64 + MAX_PATH] ;
TEXTMETRIC            tm ;
WORD                  iColCopy, iNoiColCopy ;

                      // Invoke Print common dialog box

pd.lStructSize              =      sizeof (PRINTDLG) ;
pd.hwndOwner                =      hwnd ;
pd.hDevMode                 =      NULL ;
pd.hDevNames                =      NULL ;
pd.hDC                      =      NULL ;
pd.Flags                    =      PD_ALLPAGES | PD_COLLATE |
                       PD_RETURNDC | PD_NOSELECTION ;
pd.nFromPage                =      0 ;
```

```
        pd.nToPage                =     0 ;
        pd.nMinPage               =     0 ;
        pd.nMaxPage               =     0 ;
        pd.nCopies                =     1 ;
        pd.hInstance              =     NULL ;
        pd.lCustData              =     0L ;
        pd.lpfnPrintHook          =     NULL ;
        pd.lpfnSetupHook          =     NULL ;
        pd.lpPrintTemplateName    =     NULL ;
        pd.lpSetupTemplateName    =     NULL ;
        pd.hPrintTemplate         =     NULL ;
        pd.hSetupTemplate         =     NULL ;

if    (!PrintDlg (&pd))
                return TRUE ;

if    (0 == (iTotalLines = SendMessage (hwndEdit, EM_GETLINE
                return TRUE ;
```

```
                    // Calculate necessary metrics for file

     GetTextMetrics (pd.hDC, &tm) ;
     yChar = tm.tmHeight + tm.tmExternalLeading ;

     iCharsPerLine = GetDeviceCaps (pd.hDC, HORZRES) / tm
     iLinesPerPage = GetDeviceCaps (pd.hDC, VERTRES) / yCh
iTotalPages   = (iTotalLines + iLinesPerPage - 1) / iLinesPerPag

                    // Allocate a buffer for each line of text

     pstrBuffer    = malloc (sizeof (TCHAR) * (iCharsPerLine +

                    // Display the printing dialog box

     EnableWindow (hwnd, FALSE) ;
     bSuccess                  = TRUE ;
     bUserAbort           = FALSE ;

     hDlgPrint            = CreateDialog (hInst, TEXT ("PrintDlgB
```

```
                                                    hwnd, PrintDl

 SetDlgItemText (hDlgPrint, IDC_FILENAME, szTitleName) ;

SetAbortProc (pd.hDC, AbortProc) ;

                    // Start the document

 GetWindowText (hwnd, szJobName, sizeof (szJobName)) ;

 di.lpszDocName = szJobName ;

 if (StartDoc (pd.hDC, &di) > 0)

 {

                    // Collation requires this loop and iNoiC

      for (iColCopy = 0 ;

                    iColCopy < ((WORD) pd.Flags & PD_CO

                    iColCopy++)

      {

                    for (iPage = 0 ; iPage < iTotalPages ; iPage+

                    {

    for (iNoiColCopy = 0 ;

        iNoiColCopy < (pd.Flags & PD_COLLATE ? 1 : pd.nCo
```

```
                                        iNoiColCopy++)
                    {
                                            // Start the page
                                if (StartPage (pd.hDC) < 0)
                                {
                                                    bSuccess = FA
                                                    break ;
                                }

// For each page, print the lines
for (iLine = 0 ; iLine < iLinesPerPage ; iLine++)
                                {
 iLineNum = iLinesPerPage * iPage + iLine ;
 if (iLineNum > iTotalLines)
                                        break ;
 *(int *) pstrBuffer = iCharsPerLine ;
 TextOut    (pd.hDC, 0, yChar * iLine, pstrBuffer,
 (int) SendMessage (hwndEdit, EM_GETLINE,
    (WPARAM) iLineNum, (LPARAM) pstrBuffer));
```

```
                                        }


if (EndPage (pd.hDC) < 0)
                                        {
 bSuccess = FALSE ;
 break ;
                                        }


 if (bUserAbort)
break ;
                                    }


 if (!bSuccess || bUserAbort)
 break ;
                            }


  if (!bSuccess || bUserAbort)
 break ;
        }
```

```
  }
        else
                bSuccess = FALSE ;
       if    (bSuccess)
                EndDoc (pd.hDC) ;

        if    (!bUserAbort)
   {
                EnableWindow (hwnd, TRUE) ;
                DestroyWindow (hDlgPrint) ;
        }

        free (pstrBuffer) ;
        DeleteDC (pd.hDC) ;

        return bSuccess && !bUserAbort ;
}
```

POPPADWindowsPOPPRNT.CPrintDlgcommon

PRINTDLG

FilePrintPrintPRINTDLGPrintDlg

PrintDlgPOPPADCollate3
123123123111222333

Properties

PrintDlgPRINTDLG

POPPRNT.CPopPrntPrintFileFilePrintPOPPAD
PrintDlgPopPrntPrintFileGetDeviceCapsGetTextMetrics

EM_GETLINECOUNTiTotalLinesEM_GETLINE
TextOutPOPPRNT.C

forforiColCopyforiNonColCopy

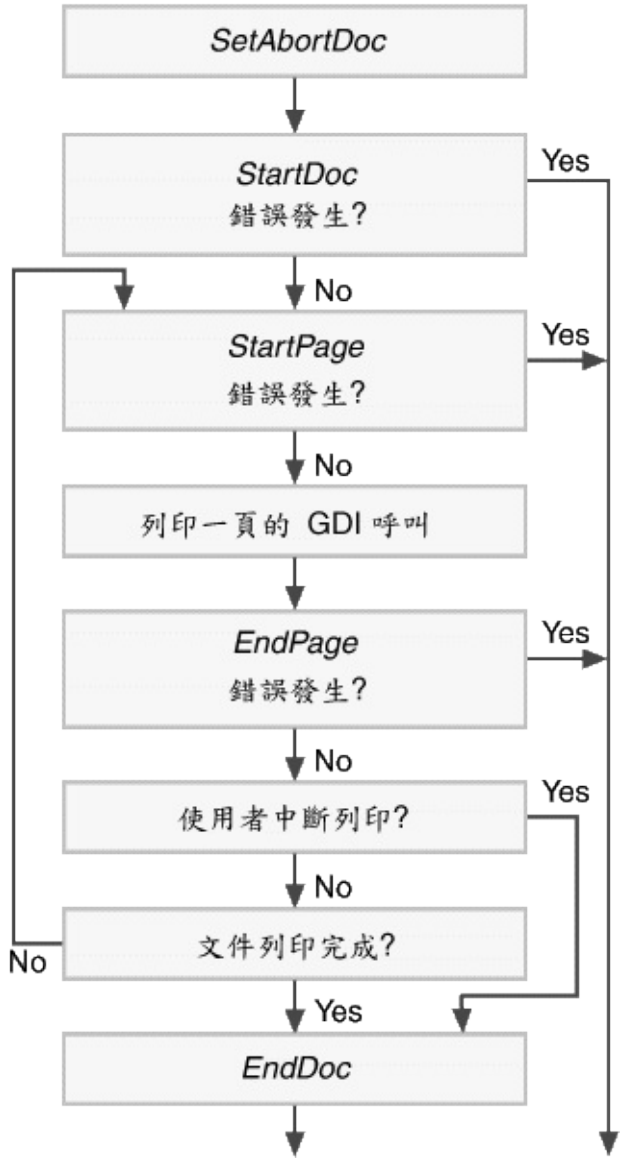
StartPageEndPagebUserAbortTRUEforFALSEEndPage
bUserAbortEndDoc

```
if (!bError)

        EndDoc (hdcPrn) ;
```

POPPADGDIEndPagePOPPADFALSE
DocumentCancel Printing POPPADI

WindowsAbortDocPOPPADPOPPADAbortDoc
POPPADAbortDocStartDocEndPageAbortDoc

13-3bUserAbortTRUEEndPageEndDoc


SetAbortDoc
StartDoc
錯誤發生?
Yes
No
StartPage
錯誤發生?
Yes
No
列印一頁的 GDI 呼叫
EndPage
錯誤發生?
Yes
No
使用者中斷列印?
Yes
No
文件列印完成?
No
Yes
EndDoc

Bitblt	

Windowsmetafile

Microsoft Windows 3.0DIBdevice-independent bitmap DIBGDI WindowsDIBDIBWindows

metafileMetafilemetafileWindows

metafile

metafile

640×48016VGAVideo
242 MBMetafilemetafileGDI

metafileInternet

Windows 98

metafile

CCD CCDCCDADCAnalog-to-digital

CCD

CCDADC

969×6

9×654cxcyccxcyxy

xy(8,

640×480300

DIB

1 bilevel bicolor monochrome0101224 41682561665,5362416,777,216

Windows

Windows 1.0IBMCGAColor Graphics Adap CardHGCCGAWindows

EGAEnhanced Graphics AdapterWindows164EGA

6416WindowsEGAEGA1616
Windows16Windows1616

16IRGBIntensity-Red-Green-BlueIBM
WindowsRGB

14-1

IRGB	RGB
0000	00-00-00
0001	00-00-80
0010	00-80-00
0011	00-80-80
0100	80-00-00
0101	80-00-80
0110	80-80-00
0111	C0-C0-C0

1000	80-80-80
1001	00-00-FF
1010	00-FF-00
1011	00-FF-FF
1100	FF-00-00
1101	FF-00-FF
1110	FF-FF-00
1111	FF-FF-FF

EGAWindowsAPI
GetDeviceCapsCreateBitmap

Windows 98Microsoft Windows NTVGA

1987IBMVideo Graphics ArrayVGAPS/2Windows640 48016256VGA320×240Windows

VGASuper-VGASVGA256640×48016

25688palette
Windows25620Windows236

IRGB	RGB
00000000	00-00-00
00000001	80-00-00
00000010	00-80-00
00000011	80-80-00
00000100	00-00-80
00000101	80-00-80
00000110	00-80-80
00000111	C0-C0-C0
00001000	C0-DC-C0
00001001	A6-CA-F0

11110110	FF-FB-F0
11110111	A0-A0-A4
11111000	80-80-80
11111001	FF-00-00
11111010	00-FF-00
11111011	FF-FF-00
11111100	00-00-FF
11111101	FF-00-FF
11111110	00-FF-FF
11111111	FF-FF-FF

True-Color1624161153232,7686
65,536PC32,76865,536Hi-Color

2416,777,216True Color3

GetDeviceCaps DEVCAPSBITSPIXELPLANES14-3

BITSPIXEL	PLANES	
1	1	2
1	4	16
8	1	256
1516	1	32,76865 536
2432	1	16 777 216

GDI

WindowsGDIGraphics Device Interface1.0Windows WindowsGDIGDI16VGA8256

Windows 3.0device-independent DIBDIB

DIBWindows 3.0Windows256

MicrosoftWindows 95Windows NT 4.0DIBWindows 98Windows NT 5.0ICMImage Color Mar

DIBGDIGDI

BitBlt

Bitbltbit blitbit-block transferBLTDEC
bitbltXerox Palo Alto Research CenterPARCSmallTalkSma
bitbltbltThen I wrote some code to blt the hap
and play a wave file.

BitBlttransferBitBlt

BitBlt

14-1BITBLTBitBltWindows

14-1 BITBLT

```
BITBLT.C

/*------------------------------------------------------------------

  BITBLT.C --   BitBlt Demonstration

                                          (c) Charles Petzold, 1998

------------------------------------------------------------------*/

#include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns

                    PSTR szCmdLine, int iCmdShow)
```

```
{
        static TCHAR szAppName [] = TEXT ("BitBlt") ;
        HWND                 hwnd ;
        MSG                  msg ;
        WNDCLASS             wndclass ;

        wndclass.style                  = CS_HREDRAW | CS_V
        wndclass.lpfnWndProc              = WndProc ;
        wndclass.cbClsExtra               = 0 ;
        wndclass.cbWndExtra               = 0 ;
        wndclass.hInstance                = hInstance ;
        wndclass.hIcon                  = LoadIcon (NULL, IDI_
        wndclass.hCursor                  = LoadCursor (NULL,
        wndclass.hbrBackground          = (HBRUSH) GetStockO
        wndclass.lpszMenuName           = NULL ;
        wndclass.lpszClassName          = szAppName ;

 if (!RegisterClass (&wndclass))
        {
```

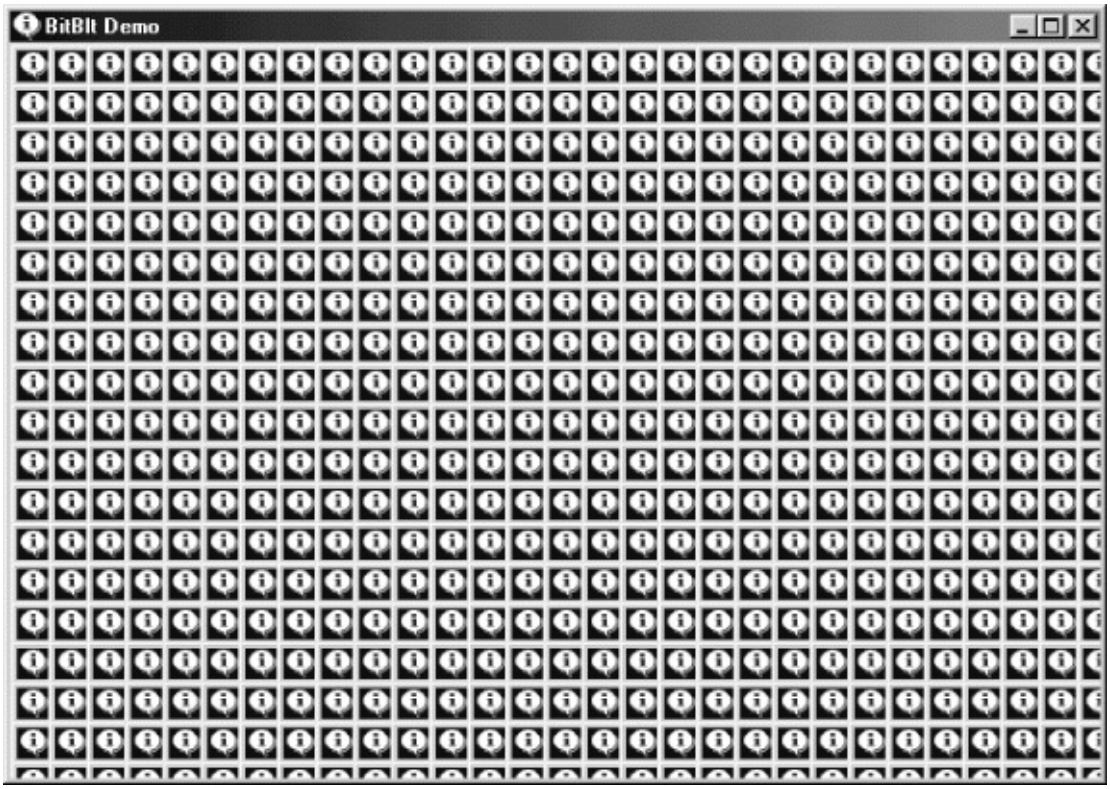
```
            MessageBox (NULL, TEXT ("This program requires Wi
                                                    szAppName, MB_IC
            return 0 ;
  }

       hwnd = CreateWindow (szAppName, TEXT ("BitBlt Demo"
                    WS_OVERLAPPEDWINDOW,
                  CW_USEDEFAULT, CW_USEDEFAULT,
                 CW_USEDEFAULT, CW_USEDEFAULT,
                  NULL, NULL, hInstance, NULL) ;

       ShowWindow (hwnd, iCmdShow) ;
       UpdateWindow (hwnd) ;
       while (GetMessage (&msg, NULL, 0, 0))
  {
                  TranslateMessage (&msg) ;
                  DispatchMessage (&msg) ;
  }
return msg.wParam ;
```

```
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
        static int          cxClient, cyClient, cxSource, cySource ;
        HDC                 hdcClient, hdcWindow ;
        int                 x, y ;
        PAINTSTRUCT         ps ;

        switch (message)
        {
        case WM_CREATE:
                cxSource = GetSystemMetrics (SM_CXSIZEFRAM
            GetSystemMetrics (SM_CXSIZE) ;
                cySource = GetSystemMetrics (SM_CYSIZEFRAM
            GetSystemMetrics (SM_CYCAPTION) ;
                return 0 ;
        case   WM_SIZE:
                cxClient = LOWORD (lParam) ;
```

```
                cyClient = HIWORD (lParam) ;
                return 0 ;

        case   WM_PAINT:
                hdcClient = BeginPaint (hwnd, &ps) ;
                hdcWindow = GetWindowDC (hwnd) ;


                for (y = 0 ; y < cyClient ; y += cySource)
                for (x = 0 ; x < cxClient ; x += cxSource)
                {
                      BitBlt (hdcClient, x, y, cxSource, cySource,
                                              hdcWindow, 0, 0, SRCC
                }

                ReleaseDC (hwnd, hdcWindow) ;
                EndPaint (hwnd, &ps) ;
                return 0 ;

        case   WM_DESTROY:
                PostQuitMessage (0) ;
```

```
                    return 0 ;

   }

        return DefWindowProc (hwnd, message, wParam, lParam)

}
```

BitBltBITBLTIDI_INFORMATION14-1



14-1 BITBLT

BitBlt(destination)

```
BitBlt (hdcDst, xDst, yDst, cx, cy, hdcSrc, xSrc, ySrc, dwROP) ;
```

BITBLTBeginPaintGetWindowDC

xSrcySrcBITBLT0cxcyBITBLTGetSytemMetrics

xDstyDstBITBLTBitBlt0

BitBlt

BitBltBITBLTBITBLTBITBLTBitBlt

BitBltBITBLTWM_PAINT

```
BitBlt (hdcClient, 0, 0, cxSource, cySource,
                hdcWindow, 0, 0, SRCCOPY) ;
for (y = 0 ; y < cyClient ; y += cySource)
for (x = 0 ; x < cxClient ; x += cxSource)
{
          if (x > 0 || y > 0)
                    BitBlt (hdcClient, x, y, cxSource, cySource,
               hdcClient, 0, 0, SRCCOPY) ;
}
```

BITBLT

BitBlt

BitBltStretchBltStretchBlt

```
StretchBlt (hdcDst, xDst, yDst, cxDst, cyDst,
                            hdcSrc, xSrc, ySrc, cxSrc, cySrc, dwROP) ;
```

STRETCHStretchBlt14-2

```
14-2  STRETCH

STRETCH.C

/*------------------------------------------------------------------------
  STRETCH.C -- StretchBlt Demonstration
                                              (c) Charles Petzold, 1998
------------------------------------------------------------------------*/

#include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                PSTR szCmdLine, int iCmdShow)

{
          static TCHAR szAppName [] = TEXT ("Stretch") ;
```

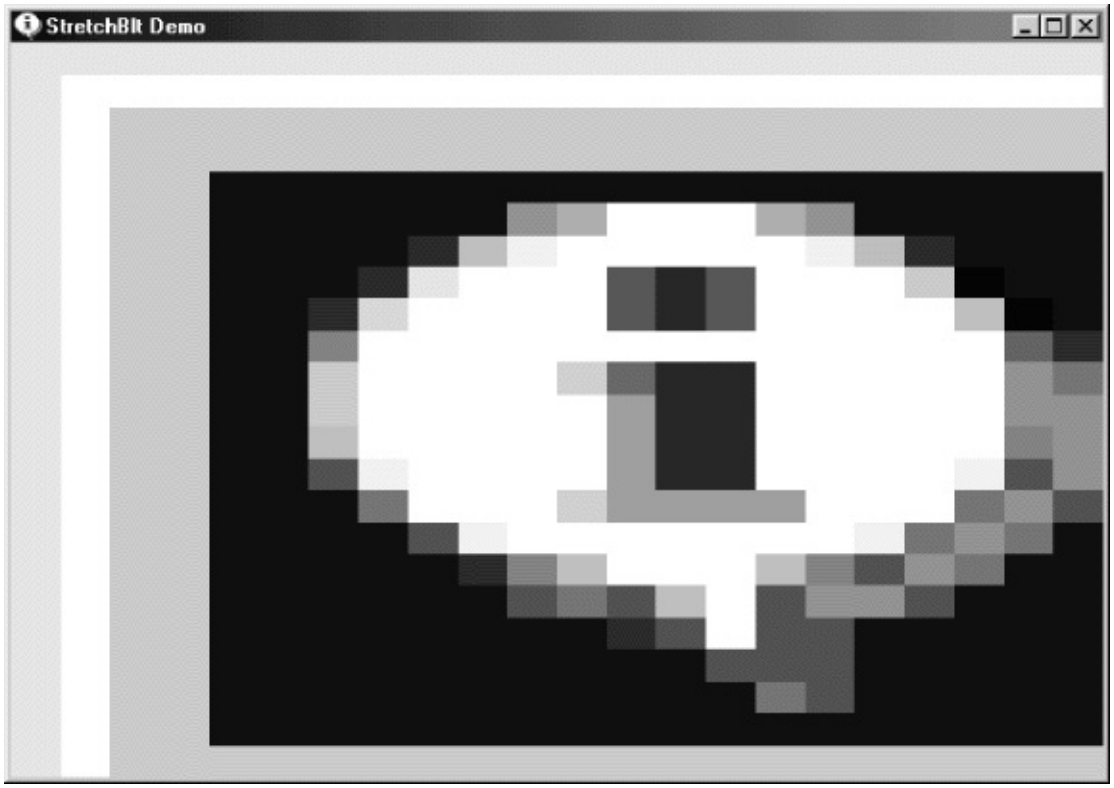
```
     HWND                 hwnd ;
     MSG                  msg ;
     WNDCLASS             wndclass ;

     wndclass.style                   = CS_HREDRAW | CS_V
     wndclass.lpfnWndProc               = WndProc ;
     wndclass.cbClsExtra                = 0 ;
     wndclass.cbWndExtra                = 0 ;
     wndclass.hInstance                 = hInstance ;
     wndclass.hIcon                   = LoadIcon (NULL, IDI_
     wndclass.hCursor                   = LoadCursor (NULL,
     wndclass.hbrBackground           = (HBRUSH) GetStockO
     wndclass.lpszMenuName             = NULL ;
     wndclass.lpszClassName           = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (  NULL, TEXT ("This program requires W
                                        szAppName, MB_IC
```

```
            return 0 ;
   }

   hwnd = CreateWindow (szAppName, TEXT ("StretchBlt Demo"
                  WS_OVERLAPPEDWINDOW,
                  CW_USEDEFAULT,
CW_USEDEFAULT,
                  CW_USEDEFAULT,
CW_USEDEFAULT,
                  NULL, NULL, hInstance, NULL) ;
        ShowWindow (hwnd, iCmdShow) ;
        UpdateWindow (hwnd) ;

        while (GetMessage (&msg, NULL, 0, 0))
        {
                        TranslateMessage (&msg) ;
                        DispatchMessage (&msg) ;
  }
        return msg.wParam ;
```

```
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
        static int          cxClient, cyClient, cxSource, cySource ;
        HDC                 hdcClient, hdcWindow ;
        PAINTSTRUCT         ps ;

        switch (message)
        {
        case   WM_CREATE:
                cxSource = GetSystemMetrics (SM_CXSIZEFRAM
             GetSystemMetrics (SM_CXSIZE) ;

                cySource = GetSystemMetrics (SM_CYSIZEFRAM
             GetSystemMetrics (SM_CYCAPTION) ;
                return 0 ;

        case   WM_SIZE:
                cxClient = LOWORD (lParam) ;
```

```
                cyClient = HIWORD (lParam) ;
                return 0 ;

        case   WM_PAINT:
                hdcClient = BeginPaint (hwnd, &ps) ;
                hdcWindow = GetWindowDC (hwnd) ;

                StretchBlt (hdcClient, 0, 0, cxClient, cyClient,
              hdcWindow, 0, 0, cxSource, cySource, MERGECOPY)

                ReleaseDC (hwnd, hdcWindow) ;
                EndPaint (hwnd, &ps) ;
                return 0 ;

        case   WM_DESTROY:
                PostQuitMessage (0) ;
                return 0 ;
   }
        return DefWindowProc (hwnd, message, wParam, lParam)
}
```

StretchBlt14-2



14-2 STRETCH

BitBltStretchBltBitBltBitBltcxcycxcy

MM_TEXTWindowsWindowsStretchBlt

StretchBltcxSrccxDstStretchBltSTRETCHxDst cxClientcxDst-cxClientcySrccyDstStretchBltSTRETCHyDst cyClientcyDst-cyClient

StretchBlt

StretchBltStretchBlt

StretchBltSetStretchBltMode

```
SetStretchBltMode (hdc, iMode) ;
```

iMode

- BLACKONWHITESTRETCH_ANDSCANSStretchBltAND
- WHITEONBLACKSTRETCH_ORSCANS StretchBltOR
- COLORONCOLORSTRETCH_DELETESCANS StretchBlt
- HALFTONESTRETCH_HALFTONE Windows

WindowsGetStretchBltMode

BITBLTSTRETCHSRCCOPYBitBltStretchBltSRCCOPY256
STRETCH

NOTSRCCOPYSRCCOPYSRCINVERT
BLACKNESSWHITENESS

StretchBlt

```
SelectObject (hdcClient, CreateHatchBrush (HS_DIAGCROSS, RG

StretchBlt (hdcClient, 0, 0, cxClient, cyClient,

                        hdcWindow, 0, 0, cxSource, cySource, MER

DeleteObject (hdcClient, GetStockObject (WHITE_BRUSH)) ;
```

BitBltStretchBlt

- Source
- Destination BitBltStretchBlt
- Pattern

160101

BitBltStretchBlt25625615WINGDI.H/Platform
SDK/Graphics and Multimedia Services/GDI/Raster Operation Codes/Ternary Raster Operations

15ROP14-4

14-4

P1 1 1 1 0 0 0 0 **s1 1 0 0 1 1 0 0** **D1 0 1 0 1 0 1 0**			**ROP**	
	0 0 0 0 0 0 0 0	0	0x000042	BLACKNESS
	0 0 0 1 0 0 0 1	~(S D)	0x1100A6	NOTSRCERASE

	0 0 1 1 0 0 1 1	~S	0x330008	NOTSRCCOPY
	0 1 0 0 0 1 0 0	S & ~D	0x440328	SRCERASE
	0 1 0 1 0 1 0 1	~D	0x550009	DSTINVERT
	0 1 0 1 1 0 1 0	P ^ D	0x5A0049	PATINVERT
	0 1 1 0 0 1 1 0	S ^ D	0x660046	SRCINVERT
	1 0 0 0 1 0 0 0	S & D	0x8800C6	SRCAND
	1 0 1 1 1 0 1 1	~S D	0xBB0226	MERGEPAINT
	1 1 0 0 0 0 0 0	P & S	0xC000CA	MERGECOPY
	1 1 0 0 1 1 0 0	S	0xCC0020	SRCCOPY
	1 1 1 0 1 1 1 0	S D	0xEE0086	SRCPAINT
	1 1 1 1 0 0 0 0	P	0xF00021	PATCOPY
	1 1 1 1 1 0 1 1	P ~S D	0xFB0A09	PATPAINT

	1 1 1 1 1 1 1 1	1	0xFF0062	WHITENESS

ROPBitBltStretchBltWINGDI.HROP02552
C

101BLACKNESSWHITENESS

PATCOPYPATCOPY

PATPAINT011

BitBltStretchBltSRCPAINTWindows

Blt

BitBltStretchBltWindowsPatBlt pattern block transf
BitBltStretchBltPatBlt

```
PatBlt (hdc, x, y, cx, cy, dwROP) ;
```

xycxcyx,ycxcyPatBltPatBltdwROPROPROP
PatBlt16

P1 **D1**	**1 0 0** **0 1 0**	**ROP**	

	0 0 0 0	0	0x000042	BLACKNESS
	0 0 0 1	~(P | D)	0x0500A9	
	0 0 1 0	~P & D	0x0A0329	
	0 0 1 1	~P	0x0F0001	
	0 1 0 0	P & ~D	0x500325	
	0 1 0 1	~D	0x550009	DSTINVERT
	0 1 1 0	P ^ D	0x5A0049	PATINVERT
	0 1 1 1	~(P & D)	0x5F00E9	
	1 0 0 0	P & D	0xA000C9	
	1 0 0 1	~(P ^ D)	0xA50065	
	1 0 1 0	D	0xAA0029	
	1 0 1 1	~P | D	0xAF0229	
	1 1 0 0	P	0xF00021	PATCOPY

	1 1 0 1	P | ~D	0xF50225	
	1 1 1 0	P | D	0xFA0089	
	1 1 1 1	1	0xFF0062	WHITENESS

PatBlt

```
PatBlt (hdc, x, y, cx, cy, BLACKNESS) ;
```

```
PatBlt (hdc, x, y, cx, cy, WHITENESS) ;
```

```
PatBlt (hdc, x, y, cx, cy, DSTINVERT) ;
```

WHITE_BRUSH

```
PatBlt (hdc, x, y, cx, cy, PATINVERT) ;
```

FillRect

```
FillRect (hdc, &rect, hBrush) ;
```

FillRect

```
hBrush = SelectObject (hdc, hBrush) ;

PatBlt (hdc,       rect.left, rect.top,

                                rect.right - rect.left,

                                rect.bottom - rect.top, PATCOPY) ;

SelectObject (hdc, hBrush) ;
```

WindowsFillRect

```
InvertRect (hdc, &rect) ;
```

Windows

```
PatBlt (hdc,       rect.left, rect.top,

                                rect.right - rect.left,

                                rect.bottom - rect.top, DSTINVERT) ;
```

PatBltx,ycxcyBitBltPatBltStretchBltGDIGDI
MM_TEXTPatBltcxcyx,yx,ycy

PatBltcxcyx,
x,   yx, y + cyx + cx, yx + cx, y + cy

MM_LOENGLISHPatBlt

```
PatBlt (hdc, 0, 0, 100, -100, dwROP) ;
```

```
PatBlt (hdc, 0, -100, 100, 100, dwROP) ;
```

```
PatBlt (hdc, 100, 0, -100, -100, dwROP) ;
```

```
PatBlt (hdc, 100, -100, -100, 100, dwROP) ;
```

PatBltxyycyxcx

GDI

Windows1.0GDIWindows 3.0GDI device-independent bitmapDDBDIB

Windows 3.0WindowsDIBDDBWindc DIBDDBDIBDDB

DIBDDBDDBWindows

DDB

DDBWindowsmetafileGDIDDB
HBITMAPhandle to a bitmap

```
HBITMAP hBitmap ;
```

DDBCreateBitmapGDI

```
DeleteObject (hBitmap) ;
```

```

```

DDB

CreateBitmap

```
hBitmap = CreateBitmap (cx, cy, cPlanes, cBitsPixel, bits) ;
```

DDBDDBNULLDDB

WindowsGDI795?3

```
hBitmap = CreateBitmap (7, 9, 5, 3, NULL) ;
```

Windows

Windows7×9×5×3945118

Windows

```
iWidthBytes = 2 * ((cx * cBitsPixel + 15) / 16) ;
```

C

```
iWidthBytes = (cx * cBitsPixel + 15) & ~15) >> 3 ;
```

DDB

```
iBitmapBytes = cy * cPlanes * iWidthBytes ;
```

iWidthBytes4iBitmapBytes180

53GDI

CreateBitmap

- cPlanescBitsPixel1
- cPlanescBitsPixelPLANESBITSPIXELGetDeviceCaps

CreateBitmapCreateCompatibleBitmap

```
hBitmap = CreateCompatibleBitmap (hdc, cx, cy) ;
```

CreateCompatibleBitmapGetDeviceCapsCreateBitmapDDB

CreateDiscardableBitmapCreateCompatibleBitmapWindows
CreateDiscardableBitmapWindows

CreateBitmapIndirect

```
hBitmap CreateBitmapIndirect (&bitmap) ;
```

bitmapBITMAPBITMAP

```
typedef struct _tagBITMAP

{

        LONG        bmType ;                // set to 0

        LONG        bmWidth ;                // width in pixels

        LONG        bmHeight ;               // height in pixels

        LONG        bmWidthBytes ;              // width of row in by
```

```
        WORD        bmPlanes ;                  // number of color pla
        WORD       bmBitsPixel ;               // number of bits per p
        LPVOIDbmBits ;                        // pointer to pixel bits
}
BITMAP, * PBITMAP ;
```

CreateBitmapIndirectbmWidthBytesWindowsbmBitsNULL

GetObjectBITMAPBITMAP

```
BITMAP bitmap ;
```

```
GetObject (hBitmap, sizeof (BITMAP), &bitmap) ;
```

WindowsBITMAPbmBitsNULL

DeleteObject

CreateBitmapCreateBitmapIndirectGDIWindows

```
SetBitmapBits (hBitmap, cBytes, &bits) ;
```

GetBitmapBits

```
GetBitmapBits (hBitmap, cBytes, &bits) ;
```

cBytesbitscBytes

DDB1110DDBDDB

8VGAWindowsCreateCompatibleBitmapGetDeviceCaps18
0x37

DDBDDB0x37RGB

DDB

CreateBitmapCreateBitmapIndirectSetBitmapBitsDDBDDB
GetBitmapBitsDDB

SetBitmapDimensionExGetBitmapDimensionEx0.1
GDIDDB

GDI

hdc

```
hdcMem = CreateCompatibleDC (hdc) ;
```

NULLWindowsDeleteDC

111

1GDI

```
SelectObject (hdcMem, hBitmap) ;
```

GDI

SelectObjectDDB53

SelectObjectDDBGDIBitBlt
BitBlt

GDILoadBitmapLoadBitmapLoadIconLoadCursor

```
hBitmap = LoadBitmap (hInstance, szBitmapName) ;
```

NULLWindowsOBMMAKEINTRESOURCE
LoadBitmapDeleteObject

LoadBitmapLoadBitmapGDILoadBitmap
LoadBitmap

14-3BRICKS1

```
14-3  BRICKS1

BRICKS1.C

/*------------------------------------------------------------------------

  BRICKS1.C -- LoadBitmap Demonstration

                                          (c) Charles Petzold, 1998

------------------------------------------------------------------------*/

#include <windows.h>
```

```
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                    PSTR szCmdLine, int iCmdShow)
{
          static TCHAR szAppName [] = TEXT ("Bricks1") ;
          HWND                 hwnd ;
          MSG                  msg ;
          WNDCLASS                 wndclass ;

          wndclass.style                    = CS_HREDRAW | CS_V
          wndclass.lpfnWndProc                   = WndProc ;
          wndclass.cbClsExtra                    = 0 ;
          wndclass.cbWndExtra                    = 0 ;
          wndclass.hInstance                     = hInstance ;
          wndclass.hIcon                    = LoadIcon (NULL, IDI_
          wndclass.hCursor                    = LoadCursor (NULL,
          wndclass.hbrBackground            = (HBRUSH) GetStockO
          wndclass.lpszMenuName              = NULL ;
          wndclass.lpszClassName             = szAppName ;
```

```
     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Wi
                                   szAppName, MB_ICONERROR
          return 0 ;
}

     hwnd = CreateWindow ( szAppName, TEXT ("LoadBitmap
                    WS_OVERLAPPEDWINDOW,
                    CW_USEDEFAULT, CW_USEDEFAULT,
                   CW_USEDEFAULT, CW_USEDEFAULT,
                    NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
               TranslateMessage (&msg) ;
```

```
                DispatchMessage (&msg) ;
        }
        return msg.wParam ;
}

LRESULT CALLBACK WndProc ( HWND hwnd, UINT message, WPA
{
        static HBITMAP hBitmap ;
        static int              cxClient, cyClient, cxSource, cySour
        BITMAP                  bitmap ;
        HDC                     hdc, hdcMem ;
        HINSTANCE               hInstance ;
        int                     x, y ;
        PAINTSTRUCT             ps ;

        switch (message)
        {
        case   WM_CREATE:
                hInstance = ((LPCREATESTRUCT) lParam)->hIns
```

```
                hBitmap = LoadBitmap (hInstance, TEXT ("Bricks
                GetObject (hBitmap, sizeof (BITMAP), &bitmap)
                cxSource = bitmap.bmWidth ;
                cySource = bitmap.bmHeight ;
                return 0 ;
        case   WM_SIZE:
                cxClient = LOWORD (lParam) ;
                cyClient = HIWORD (lParam) ;
                return 0 ;
        case   WM_PAINT:
                hdc = BeginPaint (hwnd, &ps) ;
                hdcMem = CreateCompatibleDC (hdc) ;
                SelectObject (hdcMem, hBitmap) ;
                for (y = 0 ; y < cyClient ; y += cySource)
```

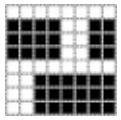
```
                    for (x = 0 ; x < cxClient ; x += cxSource)
                    {
                    BitBlt (hdc, x, y, cxSource, cySource, hdcMem, 0
          }

                    DeleteDC (hdcMem) ;
                    EndPaint (hwnd, &ps) ;
                    return 0 ;

     case   WM_DESTROY:
                    DeleteObject (hBitmap) ;
                    PostQuitMessage (0) ;
                    return 0 ;
          }
          return DefWindowProc (hwnd, message, wParam, lParam)
}
```

BRICKS1.RC

```
//Microsoft Developer Studio generated resource script.
#include "resource.h"
```

```
#include "afxres.h"

/////////////////////////////////////////////////////////////////////////////

// Bitmap

BRICKS                  BITMAP  DISCARDABLE     "Bricks.b
```

BRICKS.BMP

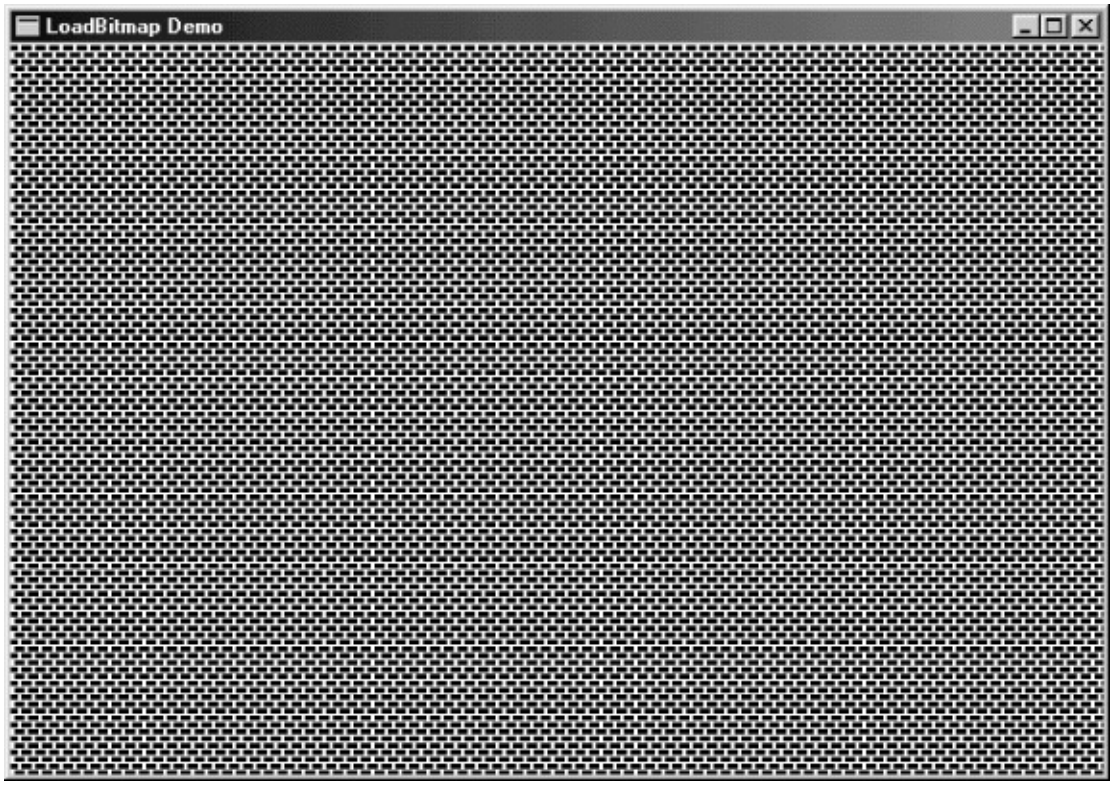


Visual C++ Developer Studio8BricksBRICKS1WM_CREATE GetObject8BRICKS1WM_DESTROY

WM_PAINTBRICKS1BitBlt14-3

Developer StudioBRICKS.BMPDeveloper

DIBGDI



14-3 BRICKS1

01816

```
0 1 0 1 0 0 0 1 0 1 1 1 0 1 1 1 0 0 0 1 = 51 77 10 00
```

```
0 1 0 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 0 1 = 57 77 50 00

0 0 0 1 0 0 1 1 0 1 1 1 0 1 1 1 0 1 0 1 = 13 77 50 00

0 1 0 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 0 1 = 57 77 50 00

0 1 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 = 51 11 10 00
```

2054BITMAP

```
static BITMAP bitmap              = { 0, 20, 5, 4, 1, 1 } ;
```

BYTE

```
static BYTE  bits []      = {    0x51, 0x77, 0x10, 0x00,

                          0x57, 0x77, 0x50, 0x00,

                               0x13, 0x77, 0x50, 0x00,

                     0x57, 0x77, 0x50, 0x00,

                       0x51, 0x11, 0x10, 0x00 } ;
```

CreateBitmapIndirect

```
bitmap.bmBits = (PSTR) bits ;

hBitmap = CreateBitmapIndirect (&bitmap) ;
```

```
hBitmap = CreateBitmapIndirect (&bitmap) ;

SetBitmapBits (hBitmap, sizeof bits, bits) ;
```

```
hBitmap = CreateBitmap (20, 5, 1, 1, bits) ;
```

14-4BRICKS2

14-4 BRICKS2

```
BRICKS2.C

/*------------------------------------------------------------------------

  BRICKS2.C -- CreateBitmap Demonstration

                                              (c) Charles Petzold, 1998

------------------------------------------------------------------------*/

#include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns

                    PSTR szCmdLine, int iCmdShow)

{

        static TCHAR szAppName [] = TEXT ("Bricks2") ;
```

```
HWND                    hwnd ;
MSG                     msg ;
WNDCLASS                wndclass ;

wndclass.style                  = CS_HREDRAW | CS_V
wndclass.lpfnWndProc              = WndProc ;
wndclass.cbClsExtra               = 0 ;
wndclass.cbWndExtra               = 0 ;
wndclass.hInstance                = hInstance ;
wndclass.hIcon                  = LoadIcon (NULL, IDI_
wndclass.hCursor                = LoadCursor (NULL,
wndclass.hbrBackground          = (HBRUSH) GetStockO
wndclass.lpszMenuName           = NULL ;
wndclass.lpszClassName          = szAppName ;

if (!RegisterClass (&wndclass))
{
     MessageBox (  NULL, TEXT ("This program requires W
                szAppName, MB_ICONERROR) ;
```

```
            return 0 ;
        }

            hwnd = CreateWindow (szAppName, TEXT ("CreateB
                WS_OVERLAPPEDWINDOW,
                CW_USEDEFAULT, CW_USEDEFAULT,
               CW_USEDEFAULT, CW_USEDEFAULT,
                NULL, NULL, hInstance, NULL) ;

        ShowWindow (hwnd, iCmdShow) ;
        UpdateWindow (hwnd) ;

        while (GetMessage (&msg, NULL, 0, 0))
 {
                TranslateMessage (&msg) ;
                DispatchMessage (&msg) ;
        }
        return msg.wParam ;
}

LRESULT CALLBACK WndProc ( HWND hwnd, UINT message, WPA
```

```
{
        static BITMA        Pbitmap = {   0, 8, 8, 2, 1, 1 } ;
        static BYTE                bits [8][2]={ 0xFF, 0, 0x0C, 0, 0x0
                                    0xFF, 0, 0xC0, 0, 0xC0, 0, 0xC0, 0
        static HBITMAP hBitmap ;
        static int                 cxClient, cyClient, cxSource, cySour
        HDC                        hdc, hdcMem ;
        int                       x, y ;
        PAINTSTRUCT                ps ;

        switch (message)
       {
        case   WM_CREATE:
                    bitmap.bmBits = bits ;
                    hBitmap      = CreateBitmapIndirect (&bitmap)
                    cxSource     = bitmap.bmWidth ;
                    cySource     = bitmap.bmHeight ;
                    return 0 ;
```

```
        case   WM_SIZE:
                    cxClient = LOWORD (lParam) ;
                    cyClient = HIWORD (lParam) ;
                    return 0 ;

        case   WM_PAINT:
                    hdc = BeginPaint (hwnd, &ps) ;

                    hdcMem = CreateCompatibleDC (hdc) ;
                    SelectObject (hdcMem, hBitmap) ;

                    for (y = 0 ; y < cyClient ; y += cySource)
                    for (x = 0 ; x < cxClient ; x += cxSource)
                    {
                                BitBlt (hdc, x, y, cxSource, cySource, h
                   }

                   DeleteDC (hdcMem) ;
                    EndPaint (hwnd, &ps) ;
                    return 0 ;
```

```
        case   WM_DESTROY:
                    DeleteObject (hBitmap) ;
                    PostQuitMessage (0) ;
                    return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam)
}
```

25614-2

BRICKSBRICKS314-5

14-5　BRICKS3

```
BRICKS3.C
/*------------------------------------------------------------------------
  BRICKS3.C -- CreatePatternBrush Demonstration
                                                (c) Charles Petzold, 1998
------------------------------------------------------------------------*/
#include <windows.h>
```

```
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                    PSTR szCmdLine, int iCmdShow)
{
          static TCHAR         szAppName [] = TEXT ("Bricks3") ;
          HBITMAP              hBitmap ;
          HBRUSH               hBrush ;
          HWND                 hwnd ;
          MSG                  msg ;
          WNDCLASS             wndclass ;

          hBitmap = LoadBitmap (hInstance, TEXT ("Bricks")) ;
          hBrush = CreatePatternBrush (hBitmap) ;
          DeleteObject (hBitmap) ;

          wndclass.style                    = CS_HREDRAW | CS_VR
          wndclass.lpfnWndProc                = WndProc ;
          wndclass.cbClsExtra                 = 0 ;
          wndclass.cbWndExtra                 = 0 ;
          wndclass.hInstance                  = hInstance ;
```

```
        wndclass.hIcon                          = LoadIcon (NULL, IDI_
        wndclass.hCursor                         = LoadCursor (NULL,
        wndclass.hbrBackground                  = hBrush ;
        wndclass.lpszMenuName                   = NULL ;
        wndclass.lpszClassName                  = szAppName ;

if (!RegisterClass (&wndclass))
        {
             MessageBox (  NULL, TEXT ("This program requires W
          szAppName, MB_ICONERROR) ;
             return 0 ;
  }

        hwnd = CreateWindow (szAppName, TEXT ("CreatePatter
                    WS_OVERLAPPEDWINDOW,
                   CW_USEDEFAULT, CW_USEDEFAULT,
                   CW_USEDEFAULT, CW_USEDEFAULT,
                    NULL, NULL, hInstance, NULL) ;
```

```
        ShowWindow (hwnd, iCmdShow) ;
        UpdateWindow (hwnd) ;

        while (GetMessage (&msg, NULL, 0, 0))
        {
             TranslateMessage (&msg) ;
             DispatchMessage (&msg) ;
   }

        DeleteObject (hBrush) ;
        return msg.wParam ;
}

LRESULT CALLBACK WndProc ( HWND hwnd, UINT message, WPA
{
        switch (message)
        {
       case   WM_DESTROY:
                    PostQuitMessage (0) ;
                     return 0 ;
```

```
   }
        return DefWindowProc (hwnd, message, wParam, lParam)
}
```

```
BRICKS3.RC
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
//////////////////////////////////////////////////////////////////////////
// Bitmap
BRICKS                          BITMAP  DISCARDABLE     "Bricks.b
```

BRICKS1BRICKS.BMP

BRICKS3WNDCLASShbrBackground

GDI8LOGBRUSHlbStyleBS_PATTERNCreatePatternBrush
CreateBrushIndirect8Windows

GDIWindowsCreatePatternBrush
CreateBrushIndirectBRICKS3

BitBltStretchBltGDI

14-6HELLOBITHello,

```
HELLOBIT.C

/*------------------------------------------------------------------------
  HELLOBIT.C --         Bitmap Demonstration
                                                    (c) Charles Petzold, 199
------------------------------------------------------------------------*/

#include <windows.h>
#include "resource.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                    PSTR szCmdLine, int iCmdShow)
{
          static TCHAR         szAppName [] = TEXT ("HelloBit") ;
          HWND                          hwnd ;
          MSG                            msg ;
          WNDCLASS                            wndclass ;

          wndclass.style                          = CS_HREDRAW | CS_V
```

```
        wndclass.lpfnWndProc                   = WndProc ;
        wndclass.cbClsExtra                    = 0 ;
        wndclass.cbWndExtra                    = 0 ;
        wndclass.hInstance                     = hInstance ;
        wndclass.hIcon                         = LoadIcon (NULL, IDI_
        wndclass.hCursor                       = LoadCursor (NULL,
        wndclass.hbrBackground                 = (HBRUSH) GetStockO
       wndclass.lpszMenuName                   = szAppName ;
        wndclass.lpszClassName                 = szAppName ;

        if (!RegisterClass (&wndclass))
        {
 MessageBox (  NULL, TEXT ("This program requires Windows N
                                              szAppName, MB_ICONER
return 0 ;
        }

 hwnd = CreateWindow (szAppName, TEXT ("HelloBit"),
                  WS_OVERLAPPEDWINDOW,
```

```
                    CW_USEDEFAULT, CW_USEDEFAULT,
                   CW_USEDEFAULT, CW_USEDEFAULT,
                    NULL, NULL, hInstance, NULL) ;

          ShowWindow (hwnd, iCmdShow) ;
          UpdateWindow (hwnd) ;

          while (GetMessage (&msg, NULL, 0, 0))
     {
               TranslateMessage (&msg) ;
               DispatchMessage (&msg) ;
          }
               return msg.wParam ;
}

LRESULT CALLBACK WndProc ( HWND hwnd, UINT message, WPA
{
          static HBITMAP       hBitmap ;
          static HDC           hdcMem ;
          static       int     cxBitmap, cyBitmap, cxClient, cyClient, i$
```

```
static TCHAR * szText = TEXT (" Hello, world! ") ;
HDC                 hdc ;
HMENU               hMenu ;
int                 x, y ;
PAINTSTRUCT         ps ;
SIZE                size ;

switch (message)
{
case   WM_CREATE:
            hdc = GetDC (hwnd) ;
            hdcMem  = CreateCompatibleDC (hdc) ;

            GetTextExtentPoint32 (hdc, szText, lstrlen (szTex
            cxBitmap = size.cx ;
            cyBitmap = size.cy ;
            hBitmap = CreateCompatibleBitmap (hdc, cxBit

            ReleaseDC (hwnd, hdc) ;
```

```
                SelectObject (hdcMem, hBitmap) ;
                TextOut (hdcMem, 0, 0, szText, lstrlen (szText)) ;
                return 0 ;

        case   WM_SIZE:
                cxClient = LOWORD (lParam) ;
                cyClient = HIWORD (lParam) ;
               return 0 ;

        case   WM_COMMAND:
                hMenu = GetMenu (hwnd) ;

                switch (LOWORD (wParam))
                {
                case   IDM_BIG:
                case   IDM_SMALL:
                                CheckMenuItem (hMenu, iSize, M
                               iSize = LOWORD (wParam) ;
                                CheckMenuItem (hMenu, iSize, M
                                InvalidateRect (hwnd, NULL, TRUI
```

```
                    break ;
            }
            return 0 ;

    case   WM_PAINT:
            hdc = BeginPaint (hwnd, &ps) ;

            switch (iSize)
            {
        case   IDM_BIG:
                    StretchBlt (hdc, 0, 0, cxClient, cy
                hdcMem, 0, 0, cxBitmap, cyBitmap, SRCCO
                                        break ;

            case   IDM_SMALL:
                    for (y = 0 ; y < cyClient ; y += cy
                    for (x = 0 ; x < cxClient ; x += cx
                    {
                BitBlt (hdc, x, y, cxBitmap, cyBitmap,
                hdcMem, 0, 0, SRCCOPY) ;
```

```
                                    }
                                     break ;
                    }

                    EndPaint (hwnd, &ps) ;
                    return 0 ;

          case   WM_DESTROY:
                    DeleteDC (hdcMem) ;
                    DeleteObject (hBitmap) ;
                    PostQuitMessage (0) ;
                   return 0 ;
          }
          return DefWindowProc (hwnd, message, wParam, lParam)
}
```

HELLOBIT.RC

```
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
```

```
////////////////////////////////////////////////////////////////////////////
// Menu
HELLOBIT MENU DISCARDABLE
BEGIN
   POPUP "&Size"
   BEGIN
                     MENUITEM "&Big",              IDM_BIG, CHECKED
                     MENUITEM "&Small",            IDM_SMALL
   END
END
```

```
RESOURCE.H
// Microsoft Developer Studio generated include file.
// Used by HelloBit.rc
#define IDM_BIG           40001
#define IDM_SMALL         40002
```

GetTextExtentPoint32TextOutWM_DESTROY
HELLOBIT

HELLOBIT14-4

14-4 HELLOBIT

HELLOBITWM_DISPLAYCHANGE
Windows

shadow bitmapWM_PAIN

14-7SKETCH

14-7 SKETCH SKETCH.C

```
/*------------------------------------------------------------------------
  SKETCH.C -- Shadow Bitmap Demonstration
                                              (c) Charles Petzold, 1998
------------------------------------------------------------------------*/
#include <windows.h>
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                                PSTR szCmdLine, int iCr
{
          static TCHAR         szAppName [] = TEXT ("Sketch") ;
          HWND                          hwnd ;
          MSG                           msg ;
          WNDCLASS                      wndclass ;

         wndclass.style                      = CS_HREDRAW | CS_VR
          wndclass.lpfnWndProc                    = WndProc ;
          wndclass.cbClsExtra                    = 0 ;
          wndclass.cbWndExtra                     = 0 ;
          wndclass.hInstance                     = hInstance ;
```

```
     wndclass.hIcon                          = LoadIcon (NULL, IDI_
     wndclass.hCursor                        = LoadCursor (NULL,
     wndclass.hbrBackground                  = (HBRUSH) GetStockO
    wndclass.lpszMenuName                    = NULL ;
     wndclass.lpszClassName                  = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (  NULL, TEXT ("This program requires W
        szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, TEXT ("Sketch"),
                    WS_OVERLAPPEDWINDOW,
                    CW_USEDEFAULT, CW_USEDEFAULT,
                    CW_USEDEFAULT, CW_USEDEFAULT,
                   NULL, NULL, hInstance, NULL) ;
```

```
        if (hwnd == NULL)
        {
             MessageBox (  NULL, TEXT ("Not enough memory to
                          szAppName, MB_ICONERROR) ;
             return 0 ;
   }

        ShowWindow (hwnd, iCmdShow) ;
       UpdateWindow (hwnd) ;

        while (GetMessage (&msg, NULL, 0, 0))
   {
                          TranslateMessage (&msg) ;
                          DispatchMessage (&msg) ;
   }
  return msg.wParam ;
}

void GetLargestDisplayMode (int * pcxBitmap, int * pcyBitmap)
{
```

```
        DEVMODE      devmode ;
        int          iModeNum = 0 ;

        * pcxBitmap = * pcyBitmap = 0 ;

        ZeroMemory (&devmode, sizeof (DEVMODE)) ;
        devmode.dmSize = sizeof (DEVMODE) ;

        while (EnumDisplaySettings (NULL, iModeNum++, &devn
        {
             * pcxBitmap = max (* pcxBitmap, (int) devmode.dml
             * pcyBitmap = max (* pcyBitmap, (int) devmode.dml
        }
}
LRESULT CALLBACK WndProc ( HWND hwnd, UINT message, WPA
{
        static BOOL   fLeftButtonDown, fRightButtonDown ;
        static HBITMAP hBitmap ;
        static HDC           hdcMem ;
        static int         cxBitmap, cyBitmap, cxClient, cyClient, xM
```

```
HDC                         hdc ;
PAINTSTRUCT         ps ;

switch (message)
{
case   WM_CREATE:
          GetLargestDisplayMode (&cxBitmap, &cyBitmap

           hdc = GetDC (hwnd) ;
           hBitmap = CreateCompatibleBitmap (hdc, cxBit
           hdcMem  = CreateCompatibleDC (hdc) ;
          ReleaseDC (hwnd, hdc) ;

     if (!hBitmap)                          // no memory for bitr
      {
                DeleteDC (hdcMem) ;
                return -1 ;
      }

           SelectObject (hdcMem, hBitmap) ;
```

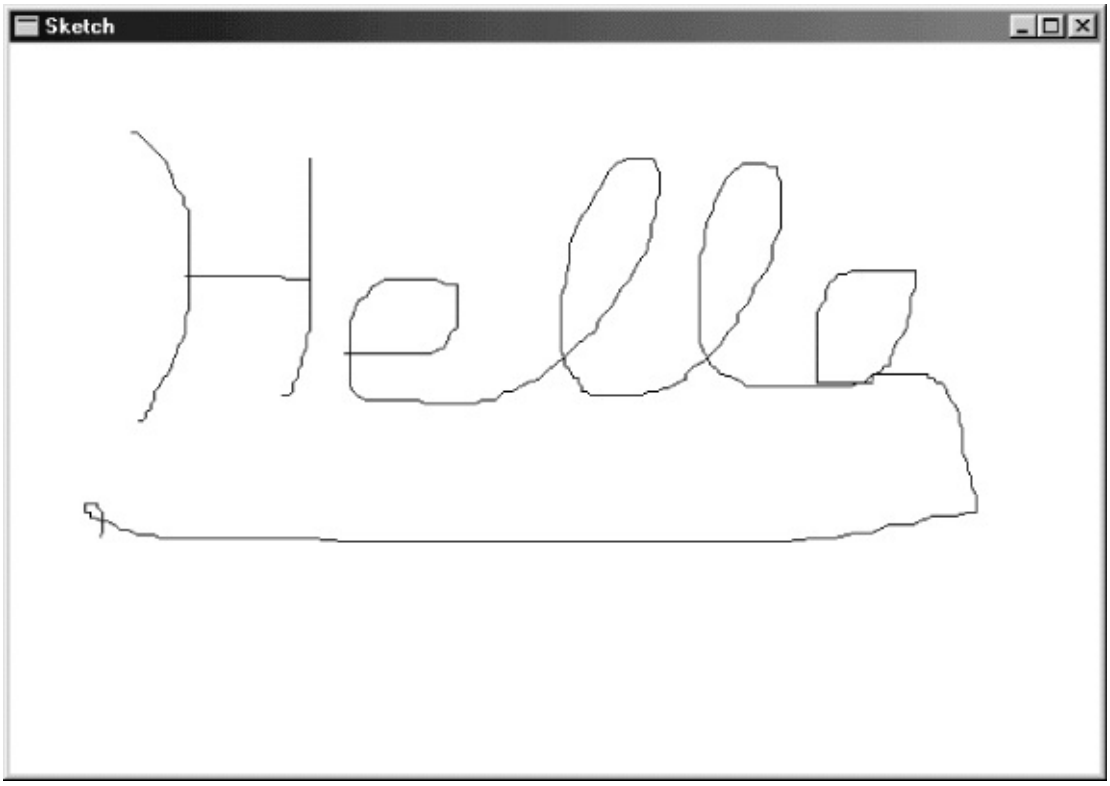
```
                PatBlt (hdcMem, 0, 0, cxBitmap, cyBitmap, WHI
                return 0 ;

        case   WM_SIZE:
                cxClient = LOWORD (lParam) ;
                cyClient = HIWORD (lParam) ;
                return 0 ;

        case   WM_LBUTTONDOWN:
                if (!fRightButtonDown)
                                SetCapture (hwnd) ;

               xMouse = LOWORD (lParam) ;
                yMouse = HIWORD (lParam) ;
                fLeftButtonDown = TRUE ;
                return 0 ;

        case   WM_LBUTTONUP:
                if (fLeftButtonDown)
                                SetCapture (NULL) ;
```

```
                fLeftButtonDown = FALSE ;
                return 0 ;

        case   WM_RBUTTONDOWN:
                if (!fLeftButtonDown)
                                SetCapture (hwnd) ;

                xMouse = LOWORD (lParam) ;
                yMouse = HIWORD (lParam) ;
                fRightButtonDown = TRUE ;
               return 0 ;

        case   WM_RBUTTONUP:
                if (fRightButtonDown)
                        SetCapture (NULL) ;

                fRightButtonDown = FALSE ;
                return 0 ;
```

```
case   WM_MOUSEMOVE:
            if (!fLeftButtonDown && !fRightButtonDown)
                 return 0 ;

            hdc = GetDC (hwnd) ;

            SelectObject (hdc,
                GetStockObject (fLeftButtonDown ? BLACK_P

            SelectObject (hdcMem,
                   GetStockObject (fLeftButtonDown ? BLACK_

            MoveToEx (hdc,    xMouse, yMouse, NULL) ;
           MoveToEx (hdcMem, xMouse, yMouse, NULL) ;

            xMouse = (short) LOWORD (lParam) ;
            yMouse = (short) HIWORD (lParam) ;

            LineTo (hdc,    xMouse, yMouse) ;
            LineTo (hdcMem, xMouse, yMouse) ;
```

```
                    ReleaseDC (hwnd, hdc) ;
                    return 0 ;

          case   WM_PAINT:
                    hdc = BeginPaint (hwnd, &ps) ;
                    BitBlt (hdc, 0, 0, cxClient, cyClient, hdcMem, 0,

                    EndPaint (hwnd, &ps) ;
                    return 0 ;

          case   WM_DESTROY:
                    DeleteDC (hdcMem) ;
                    DeleteObject (hBitmap) ;
                    PostQuitMessage (0) ;
                    return 0 ;
          }
          return DefWindowProc (hwnd, message, wParam, lParam)
}
```

SKETCH…14-5SKETCH



14-5 SKETCH

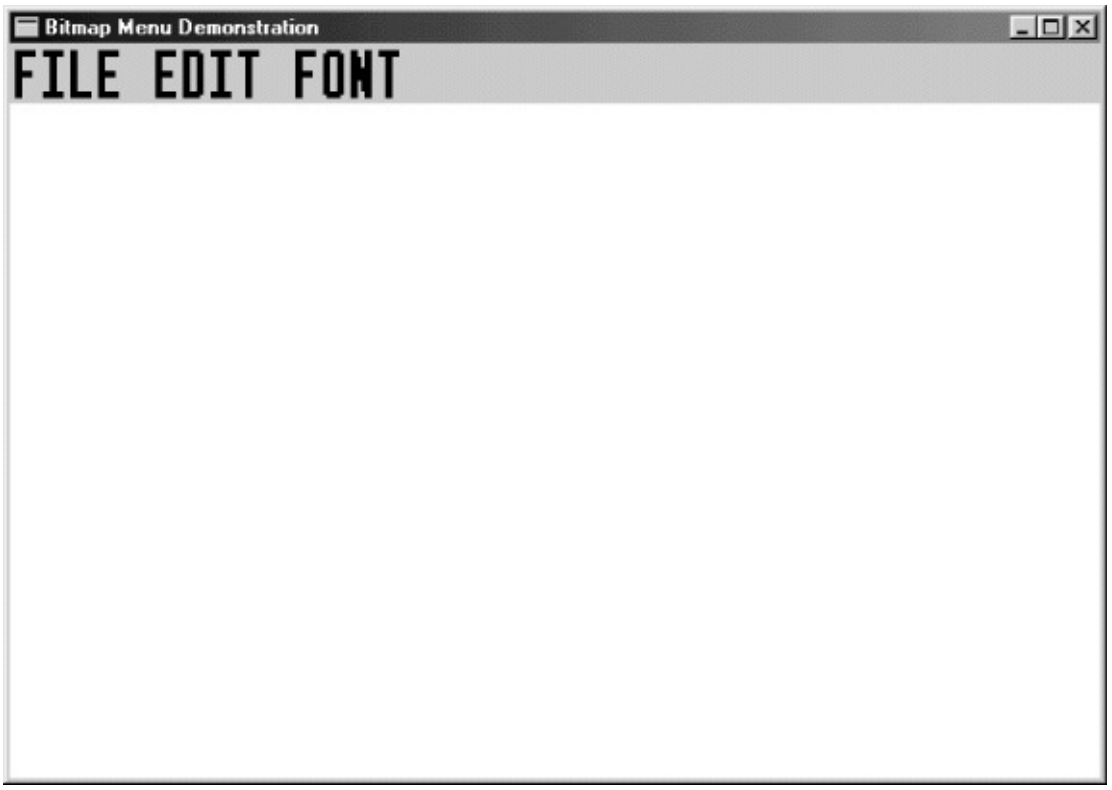
GetSystemMetricsSKETCHEnumDisplaySettings
DEVMODEEnumDisplaySettings0EnumDisplaySettingsFALSE

SKETCHSKETCHWM_CREATE-1

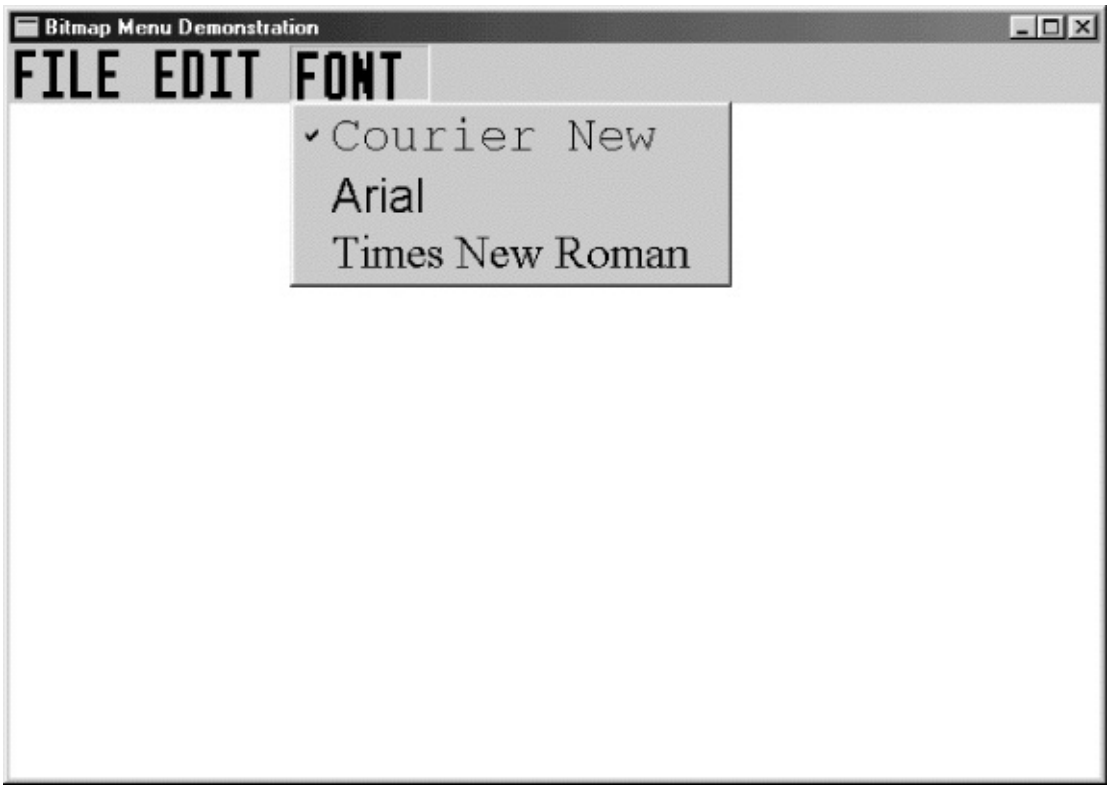
WM_MOUSEMOVESKETCH

SKETCHSKETCHWM_MOUSEMOVESKETCH

GRAFMENU14-640×16Visual
Courier New ArialTimes New RomanWindows True
14-7

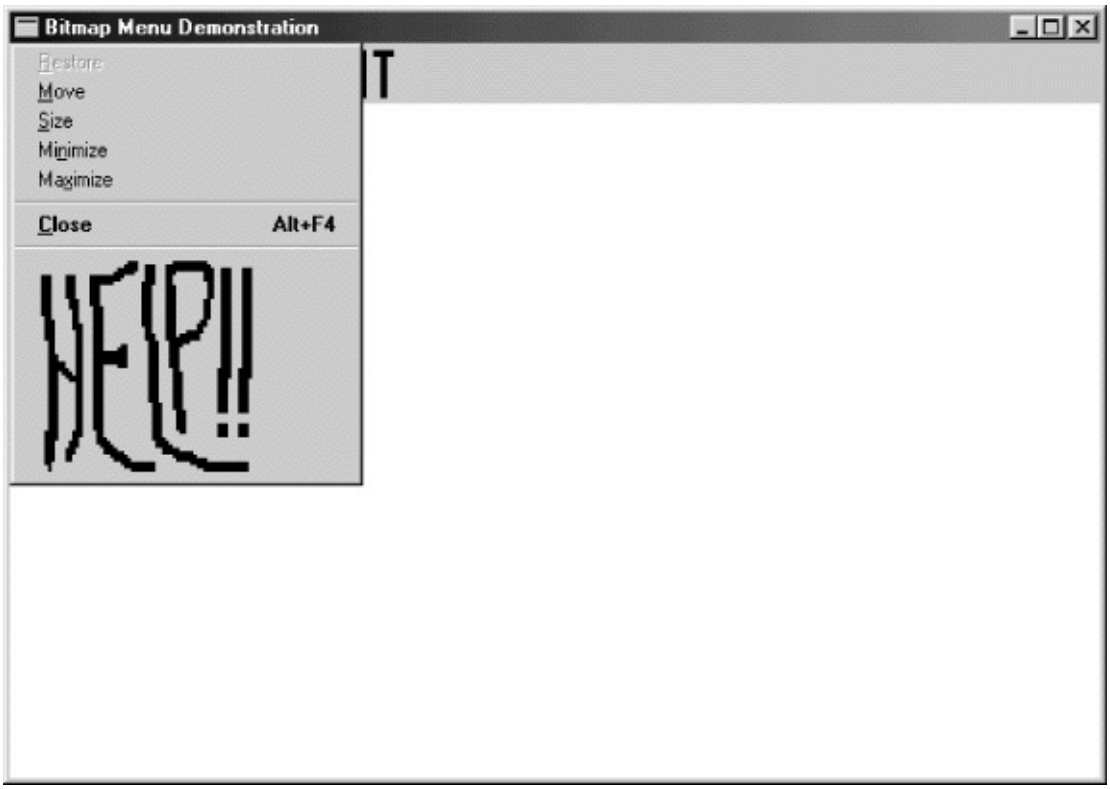


14-6 GRAFMENU



14-7　GRAFMENUFONT

HELP14-864×64Developer



14-8 GRAFMENU

GRAFMENUDeveloper Studio14-8

14-8 GRAFMENU

```
GRAFMENU.C

/*------------------------------------------------------------------------

  GRAFMENU.C --         Demonstrates Bitmap Menu Items

                                                (c) Charles Petzold, 1998

------------------------------------------------------------------------*/

#include <windows.h>
```

```
#include "resource.h"

LRESULT          CALLBACK WndProc            (HWND, UINT, WPAR
void             AddHelpToSys                (HINSTANCE, HWND) ;
HMENU            CreateMyMenu                (HINSTANCE) ;
HBITMAP          StretchBitmap           (HBITMAP) ;
HBITMAP          GetBitmapFont               (int) ;
void             DeleteAllBitmaps        (HWND) ;
TCHAR szAppName[] = TEXT ("GrafMenu") ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                PSTR szCmdLine, int iCmdShow)
{
        HWND                hwnd ;
        MSG                 msg ;
        WNDCLASS            wndclass ;

        wndclass.style                      = CS_HREDRAW | CS_V
        wndclass.lpfnWndProc                    = WndProc ;
        wndclass.cbClsExtra                     = 0 ;
```

```
     wndclass.cbWndExtra                    = 0 ;
     wndclass.hInstance                     = hInstance ;
     wndclass.hIcon                         = LoadIcon (NULL, IDI_
     wndclass.hCursor                       = LoadCursor (NULL,
     wndclass.hbrBackground                 = (HBRUSH) GetStockO
     wndclass.lpszMenuName                  = NULL ;
     wndclass.lpszClassName                 = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (  NULL, TEXT ("This program requires W
                        szAppName, MB_ICONERROR) ;
          return 0 ;
}

     hwnd = CreateWindow (szAppName,TEXT ("Bitmap Menu
                WS_OVERLAPPEDWINDOW,
               CW_USEDEFAULT, CW_USEDEFAULT,
```

```
                    CW_USEDEFAULT, CW_USEDEFAULT,
                    NULL, NULL, hInstance, NULL) ;

          ShowWindow (hwnd, iCmdShow) ;
          UpdateWindow (hwnd) ;

          while (GetMessage (&msg, NULL, 0, 0))
          {
               TranslateMessage (&msg) ;
               DispatchMessage (&msg) ;
          }
          return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT iMsg, WPARAM
{
          HMENU                         hMenu ;
          static int          iCurrentFont = IDM_FONT_COUR ;
```

```
        switch (iMsg)
        {
        case WM_CREATE:
                    AddHelpToSys (((LPCREATESTRUCT) lParam)->h
                    hMenu = CreateMyMenu (((LPCREATESTRUCT) l
                    SetMenu (hwnd, hMenu) ;
                    CheckMenuItem (hMenu, iCurrentFont, MF_CHE
                    return 0 ;

        case   WM_SYSCOMMAND:
                    switch (LOWORD (wParam))
                    {
                    case   IDM_HELP:
                                MessageBox (hwnd, TEXT ("Help r
                          szAppName, MB_OK | MB_ICONEXCLAMATI
                                return 0 ;
                    }
                    break ;
```

```
case   WM_COMMAND:
          switch (LOWORD (wParam))
          {
          case   IDM_FILE_NEW:
          case   IDM_FILE_OPEN:
          case   IDM_FILE_SAVE:
          case   IDM_FILE_SAVE_AS:
          case   IDM_EDIT_UNDO:
          case   IDM_EDIT_CUT:
          case   IDM_EDIT_COPY:
          case   IDM_EDIT_PASTE:
          case   IDM_EDIT_CLEAR:
                        MessageBeep (0) ;
                        return 0 ;

          case   IDM_FONT_COUR:
          case   IDM_FONT_ARIAL:
          case   IDM_FONT_TIMES:
```

```
                        hMenu = GetMenu (hwnd) ;
                CheckMenuItem (hMenu, iCurrentFont, MF_UNCI
                        iCurrentFont = LOWORD (wParam
                        CheckMenuItem (hMenu, iCurrentI
                        return 0 ;
                }
                break ;

            case   WM_DESTROY:
                    DeleteAllBitmaps (hwnd) ;
                    PostQuitMessage (0) ;
                    return 0 ;
        }
        return DefWindowProc (hwnd, iMsg, wParam, lParam) ;
}

/*---------------------------------------------------------------------
  AddHelpToSys: Adds bitmap Help item to system menu
---------------------------------------------------------------------*/
```

```
void AddHelpToSys (HINSTANCE hInstance, HWND hwnd)
{
        HBITMAP             hBitmap ;
        HMENU               hMenu ;

        hMenu = GetSystemMenu (hwnd, FALSE);
        hBitmap = StretchBitmap (LoadBitmap (hInstance, TEXT (
        AppendMenu (hMenu, MF_SEPARATOR, 0, NULL) ;
       AppendMenu (hMenu, MF_BITMAP, IDM_HELP, (PTSTR) (LO
}

/*-----------------------------------------------------------------------
  CreateMyMenu: Assembles menu from components
-----------------------------------------------------------------------*/

HMENU CreateMyMenu (HINSTANCE hInstance)
{
        HBITMAP             hBitmap ;
        HMENU               hMenu, hMenuPopup ;
        int                 i ;
```

```
        hMenu = CreateMenu () ;
        hMenuPopup = LoadMenu (hInstance, TEXT ("MenuFile"))
        hBitmap = StretchBitmap (LoadBitmap (hInstance, TEXT
        AppendMenu (hMenu, MF_BITMAP | MF_POPUP, (int) hMer
                                                (PTSTR) (LONG) hBitmap
        hMenuPopup = LoadMenu (hInstance, TEXT ("MenuEdit"))
        hBitmap = StretchBitmap (LoadBitmap (hInstance, TEXT
        AppendMenu (hMenu, MF_BITMAP | MF_POPUP, (int) hMer
                (PTSTR) (LONG) hBitmap) ;
        hMenuPopup = CreateMenu () ;
        for (i = 0 ; i < 3 ; i++)
{
             hBitmap = GetBitmapFont (i) ;
              AppendMenu (hMenuPopup, MF_BITMAP, IDM_FONT_
                (PTSTR) (LONG) hBitmap) ;
        }

        hBitmap = StretchBitmap (LoadBitmap (hInstance, TEXT
```

```
        AppendMenu (hMenu, MF_BITMAP | MF_POPUP, (int) hMer
                   (PTSTR) (LONG) hBitmap) ;
        return hMenu ;
}

/*------------------------------------------------------------------------
  StretchBitmap: Scales bitmap to display resolution
------------------------------------------------------------------------*/

HBITMAP StretchBitmap (HBITMAP hBitmap1)
{
        BITMAP                  bm1, bm2 ;
        HBITMAP                  hBitmap2 ;
        HDC                     hdc, hdcMem1, hdcMem2 ;
        int                     cxChar, cyChar ;

             // Get the width and height of a system font characte

        cxChar = LOWORD (GetDialogBaseUnits ()) ;
        cyChar = HIWORD (GetDialogBaseUnits ()) ;
```

```
        // Create 2 memory DCs compatible with the display
      hdc = CreateIC (TEXT ("DISPLAY"), NULL, NULL, NULL) ;
      hdcMem1 = CreateCompatibleDC (hdc) ;
      hdcMem2 = CreateCompatibleDC (hdc) ;
      DeleteDC (hdc) ;

        // Get the dimensions of the bitmap to be stretched
      GetObject (hBitmap1, sizeof (BITMAP), (PTSTR) &bm1) ;
        // Scale these dimensions based on the system font s
      bm2 = bm1 ;
      bm2.bmWidth               = (cxChar * bm2.bmWidth)  / 4
      bm2.bmHeight              = (cyChar * bm2.bmHeight) / 8
      bm2.bmWidthBytes            = ((bm2.bmWidth + 15) / 16)

        // Create a new bitmap of larger size

      hBitmap2 = CreateBitmapIndirect (&bm2) ;
        // Select the bitmaps in the memory DCs and do a St
      SelectObject (hdcMem1, hBitmap1) ;
     SelectObject (hdcMem2, hBitmap2) ;
```

```
        StretchBlt (hdcMem2, 0, 0, bm2.bmWidth, bm2.bmHeight
              hdcMem1, 0, 0, bm1.bmWidth, bm1.bmHeight, SRCC
                    // Clean up
        DeleteDC (hdcMem1) ;
        DeleteDC (hdcMem2) ;
        DeleteObject (hBitmap1) ;

        return hBitmap2 ;
}

/*---------------------------------------------------------------------
  GetBitmapFont: Creates bitmaps with font names
---------------------------------------------------------------------*/

HBITMAP GetBitmapFont (int i)
{
        static TCHAR  * szFaceName[3]= {     TEXT ("Courier New
      TEXT ("Times New Roman") } ;
        HBITMAP                         hBitmap ;
```

```
        HDC                     hdc, hdcMem ;
        HFONT                   hFont ;
        SIZE                    size ;
        TEXTMETRIC                  tm ;

        hdc = CreateIC (TEXT ("DISPLAY"), NULL, NULL, NULL) ;
        GetTextMetrics (hdc, &tm) ;

        hdcMem = CreateCompatibleDC (hdc) ;
        hFont  = CreateFont (2 * tm.tmHeight, 0, 0, 0, 0, 0, 0, 0, 0
                    szFaceName[i]) ;

        hFont = (HFONT) SelectObject (hdcMem, hFont) ;
        GetTextExtentPoint32 (hdcMem, szFaceName[i],
    lstrlen (szFaceName[i]), &size);
        hBitmap = CreateBitmap (size.cx, size.cy, 1, 1, NULL) ;
        SelectObject (hdcMem, hBitmap) ;

TextOut (hdcMem, 0, 0, szFaceName[i], lstrlen (szFaceName[i]))
```

```
        DeleteObject (SelectObject (hdcMem, hFont)) ;
        DeleteDC (hdcMem) ;
        DeleteDC (hdc) ;

        return hBitmap ;
}

/*---------------------------------------------------------------------
  DeleteAllBitmaps: Deletes all the bitmaps in the menu
---------------------------------------------------------------------*/

void DeleteAllBitmaps (HWND hwnd)
{
        HMENU                   hMenu ;
        int                     i ;
        MENUITEMINFO mii = { sizeof (MENUITEMINFO), MIIM_SU
                // Delete Help bitmap on system menu
        hMenu = GetSystemMenu (hwnd, FALSE);
        GetMenuItemInfo (hMenu, IDM_HELP, FALSE, &mii) ;
        DeleteObject ((HBITMAP) mii.dwTypeData) ;
```

```
                // Delete top-level menu bitmaps

        hMenu = GetMenu (hwnd) ;

        for (i = 0 ; i < 3 ; i++)

        {

                GetMenuItemInfo (hMenu, i, TRUE, &mii) ;

                DeleteObject ((HBITMAP) mii.dwTypeData) ;

        }


                // Delete bitmap items on Font menu

        hMenu = mii.hSubMenu ;;

        for (i = 0 ; i < 3 ; i++)

        {

             GetMenuItemInfo (hMenu, i, TRUE, &mii) ;

             DeleteObject ((HBITMAP) mii.dwTypeData) ;

        }

}
```

GRAFMENU.RC

```
//Microsoft Developer Studio generated resource script.
```

```
#include "resource.h"
#include "afxres.h"
//////////////////////////////////////////////////////////////////////////////
// Menu
MENUFILE MENU DISCARDABLE
BEGIN
        MENUITEM "&New",                IDM_FILE_NEW
        MENUITEM "&Open...",            IDM_FILE_OPEN
        MENUITEM "&Save",               IDM_FILE_SAVE
        MENUITEM "Save &As...",         IDM_FILE_SAVE_AS
END
MENUEDIT MENU DISCARDABLE
BEGIN
        MENUITEM "&Undo",               IDM_EDIT_UNDO
        MENUITEM SEPARATOR
        MENUITEM "Cu&t",                IDM_EDIT_CUT
        MENUITEM "&Copy",               IDM_EDIT_COPY
        MENUITEM "&Paste",              IDM_EDIT_PASTE
```

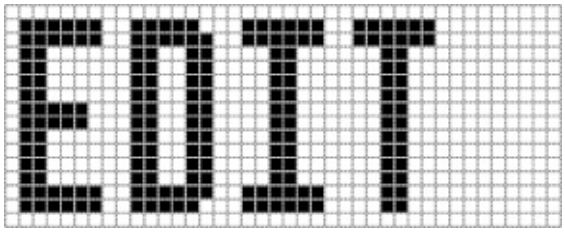
```
        MENUITEM "De&lete",          IDM_EDIT_CLEAR
END

////////////////////////////////////////////////////////////////////////////
// Bitmap
BITMAPFONT      BITMAP     DISCARDABLE     "Fontlabl.bmp"
BITMAPHELP      BITMAP     DISCARDABLE     "Bighelp.bmp"
BITMAPEDIT      BITMAP     DISCARDABLE     "Editlabl.bmp"
BITMAPFILE      BITMAP     DISCARDABLE     "Filelabl.bmp"
```

RESOURCE.H

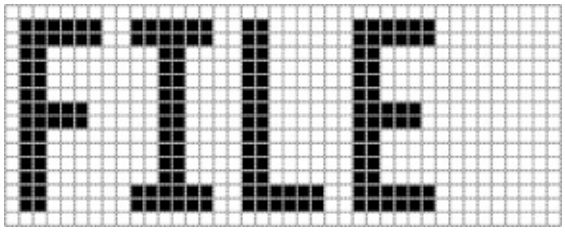
```
// Microsoft Developer Studio generated include file.
// Used by GrafMenu.rc
#define IDM_FONT_COUR    101
#define IDM_FONT_ARIAL   102
#define IDM_FONT_TIMES   103
#define IDM_HELP         104
#define IDM_EDIT_UNDO    40005
#define IDM_EDIT_CUT     40006
#define IDM_EDIT_COPY    40007
```

```
#define IDM_EDIT_PASTE   40008
#define IDM_EDIT_CLEAR   40009
#define IDM_FILE_NEW     40010
#define IDM_FILE_OPEN    40011
#define IDM_FILE_SAVE    40012
#define IDM_FILE_SAVE_AS 40013
```

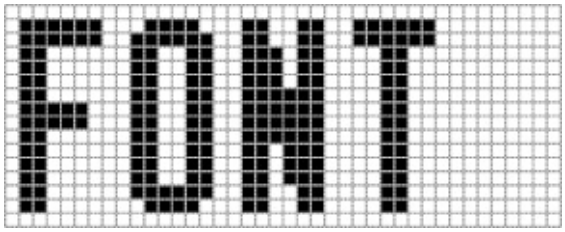
EDITLABL.BMP



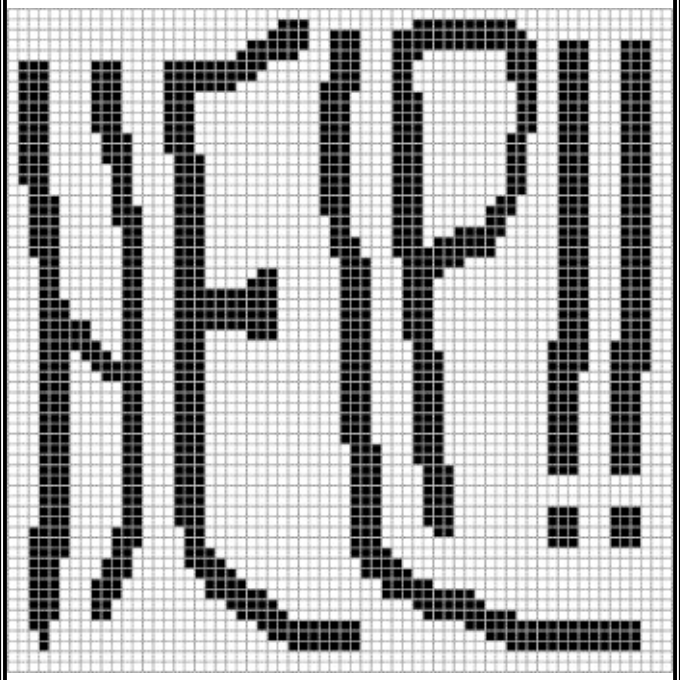
FILELABL.BMP



FONTLABL.BMP



BIGHELP.BMP



AppendMenuInsertMenuVisual
LoadBitmapAppendMenuInsertMenu

GRAFMENUGetBitmapFont012Courier
New RomanGetBitmapFontGRAFMENU.CArial
szFaceName

TEXTMETRIC

```
hdc = CreateIC (TEXT ("DISPLAY"), NULL, NULL, NULL) ;
```

```
GetTextMetrics (hdc, &tm) ;

hdcMem = CreateCompatibleDC (hdc) ;
```

CreateFontArial

```
hFont = CreateFont (2 * tm.tmHeight, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
                                    TEXT ("Arial")) ;
```

```
hFont = (HFONT) SelectObject (hdcMem, hFont) ;
```

WindowsTrueType A

GetTextExtentPoint32CreateBitmap

```
GetTextExtentPoint32 (hdcMem, TEXT ("Arial"), 5, &size) ;

hBitmap = CreateBitmap (size.cx, size.cy, 1, 1, NULL) ;

SelectObject (hdcMem, hBitmap) ;
```

```
TextOut (hdcMem, 0, 0, TEXT ("Arial"), 5) ;
```

SelectObjecthFontSelectObjectArial

```
DeleteObject (SelectObject (hdcMem, hFont)) ;
```

```
DeleteDC (hdcMem) ;

DeleteDC (hdc) ;
```

ArialArial

GRAFMENU84GRAFMENUStretchBitmap

```
hdc = CreateIC (TEXT ("DISPLAY"), NULL, NULL, NULL) ;

GetTextMetrics (hdc, &tm) ;

hdcMem1 = CreateCompatibleDC (hdc) ;

hdcMem2 = CreateCompatibleDC (hdc) ;

DeleteDC (hdc) ;
```

hBitmap1GetObject

```
GetObject (hBitmap1, sizeof (BITMAP), (PSTR) &bm1) ;
```

BITMAPbm1bm2bm1

```
bm2 = bm1 ;

bm2.bmWidth              = (tm.tmAveCharWidth * bm2.bmWidth)

bm2.bmHeight             = (tm.tmHeight      * bm2.bmHeight) / 8
```

```
bm2.bmWidthBytes          = ((bm2.bmWidth + 15) / 16) * 2 ;
```

hBitmap2

```
hBitmap2 = CreateBitmapIndirect (&bm2) ;
```

```
SelectObject (hdcMem1, hBitmap1) ;

SelectObject (hdcMem2, hBitmap2) ;
```

StretchBlt

```
StretchBlt (hdcMem2, 0, 0, bm2.bmWidth, bm2.bmHeight,

                           hdcMem1, 0, 0, bm1.bmWidth, bm1.bmHe
```

```
DeleteDC (hdcMem1) ;

DeleteDC (hdcMem2) ;

DeleteObject (hBitmap1) ;
```

GRAFMENUCreateMyMenuStretchBitmapGetBitmapFont
GRAFMENUFileEdit

```
hMenu = CreateMenu () ;
```

FileNewOpenSaveSave

```
hMenuPopup = LoadMenu (hInstance, TEXT ("MenuFile")) ;
```

FILEStretchBitmap

```
hBitmapFile = StretchBitmap (LoadBitmap (hInstance, TEXT ("Bi
```

AppendMenu

```
AppendMenu (hMenu, MF_BITMAP | MF_POPUP, hMenuPopup, (PT
hBitmapFile) ;
```

Edit

```
hMenuPopup = LoadMenu (hInstance, TEXT ("MenuEdit")) ;
hBitmapEdit = StretchBitmap (LoadBitmap (hInstance, TEXT ("Bi
AppendMenu (hMenu, MF_BITMAP | MF_POPUP, hMenuPopup, (PT
```

GetBitmapFont

```
hMenuPopup = CreateMenu () ;
for (i = 0 ; i < 3 ; i++)
{
        hBitmapPopFont [i] = GetBitmapFont (i) ;
        AppendMenu (hMenuPopup, MF_BITMAP, IDM_FONT_COUI
```

```
                (PTSTR) (LONG) hMenuPopupFont [i]) ;
}
```

```
hBitmapFont = StretchBitmap (LoadBitmap (hInstance, "BitmapF
AppendMenu (hMenu, MF_BITMAP | MF_POPUP, hMenuPopup, (PT
```

WndProcSetMenu

GRAFMENUAddHelpToSys

```
hMenu = GetSystemMenu (hwnd, FALSE) ;
```

HELP

```
hBitmapHelp = StretchBitmap (LoadBitmap (hInstance, TEXT ("B
```

```
AppendMenu (hMenu, MF_SEPARATOR, 0, NULL) ;
AppendMenu (hMenu, MF_BITMAP, IDM_HELP, (PTSTR)(LONG) hB
```

GRAFMENU

Windows()SM_CYMENUGetSystemMetrics

GRAFMENUSetMenuItemBitmaps

WindowsAltWindows

WM_MENUCHARAltWindowsWM_MENUCHARGRAFMENU
WM_MENUCHARwParamASCIIWindows2Windows

Visual C++ Developer StudioWindows

mask1()0()

BITMASK14-9

14-9 BITMASK

```
BITMASK.C

/*------------------------------------------------------------------------

  BITMASK.C -- Bitmap Masking Demonstration

                    (c) Charles Petzold, 1998

------------------------------------------------------------------------*/

#include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
```

```
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                    PSTR szCmdLine, int iCmdShow)
{
          static TCHAR        szAppName [] = TEXT ("BitMask") ;
          HWND                          hwnd ;
          MSG                           msg ;
          WNDCLASS                      wndclass ;

          wndclass.style                     = CS_HREDRAW | CS_V
          wndclass.lpfnWndProc                 = WndProc ;
          wndclass.cbClsExtra                  = 0 ;
          wndclass.cbWndExtra                  = 0 ;
          wndclass.hInstance                   = hInstance ;
          wndclass.hIcon                     = LoadIcon (NULL, IDI_
          wndclass.hCursor                   = LoadCursor (NULL,
          wndclass.hbrBackground             = (HBRUSH) GetStockO
        wndclass.lpszMenuName              = NULL ;
          wndclass.lpszClassName             = szAppName ;
```

```
     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Wi
                         szAppName, MB_ICONERROR) ;
          return 0 ;
}


     hwnd = CreateWindow (szAppName, TEXT ("Bitmap Mask
               WS_OVERLAPPEDWINDOW,
               CW_USEDEFAULT, CW_USEDEFAULT,
              CW_USEDEFAULT, CW_USEDEFAULT,
               NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
               TranslateMessage (&msg) ;
              DispatchMessage (&msg) ;
```

```
        }
        return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
        static HBITMAP          hBitmapImag, hBitmapMask ;
        static HINSTANCE         hInstance ;
        static int              cxClient, cyClient, cxBitmap, cyBitm
        BITMAP                   bitmap ;
        HDC                     hdc, hdcMemImag, hdcMemMask ;
       int                    x, y ;
        PAINTSTRUCT                ps ;

        switch (message)
   {
        case WM_CREATE:
                hInstance = ((LPCREATESTRUCT) lParam)->hIns
                        // Load the original image and get its s
```

```
hBitmapImag = LoadBitmap (hInstance, TEXT ("
GetObject (hBitmapImag, sizeof (BITMAP), &bitm
cxBitmap = bitmap.bmWidth ;
cyBitmap = bitmap.bmHeight ;

            // Select the original image into a mem
hdcMemImag  = CreateCompatibleDC (NULL) ;
SelectObject (hdcMemImag, hBitmapImag) ;
            // Create the monochrome mask bitma
hBitmapMask = CreateBitmap (cxBitmap, cyBitm
hdcMemMask = CreateCompatibleDC (NULL) ;
SelectObject (hdcMemMask, hBitmapMask) ;

            // Color the mask bitmap black with a w
SelectObject (hdcMemMask, GetStockObject (BL
Rectangle (hdcMemMask, 0, 0, cxBitmap, cyBitm
SelectObject (hdcMemMask, GetStockObject (WH
Ellipse (hdcMemMask, 0, 0, cxBitmap, cyBitmap

            // Mask the original image
```

```
                BitBlt (hdcMemImag, 0, 0, cxBitmap, cyBitmap,
                       hdcMemMask, 0, 0, SRCAND) ;
                DeleteDC (hdcMemImag) ;
                DeleteDC (hdcMemMask) ;
                return 0 ;

        case   WM_SIZE:
                cxClient = LOWORD (lParam) ;
                cyClient = HIWORD (lParam) ;
                return 0 ;

        case   WM_PAINT:
                hdc = BeginPaint (hwnd, &ps) ;

                          // Select bitmaps into memory DCs

                hdcMemImag = CreateCompatibleDC (hdc) ;
                SelectObject (hdcMemImag, hBitmapImag) ;

                hdcMemMask = CreateCompatibleDC (hdc) ;
                SelectObject (hdcMemMask, hBitmapMask) ;
```

```
                    // Center image

                    x = (cxClient - cxBitmap) / 2 ;

                    y = (cyClient - cyBitmap) / 2 ;


                    // Do the bitblts

                    BitBlt (hdc, x, y, cxBitmap, cyBitmap, hdcMemM
                    BitBlt (hdc, x, y, cxBitmap, cyBitmap, hdcMemIr

                    DeleteDC (hdcMemImag) ;
                    DeleteDC (hdcMemMask) ;
                    EndPaint (hwnd, &ps) ;
                    return 0 ;

          case   WM_DESTROY:
                    DeleteObject (hBitmapImag) ;
                    DeleteObject (hBitmapMask) ;
                    PostQuitMessage (0) ;
                    return 0 ;
```

```
   }
        return DefWindowProc (hwnd, message, wParam, lParam)
}
```

```
BITMASK.RC
// Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
//////////////////////////////////////////////////////////////////////////
// Bitmap
MATTHEW                 BITMAP     DISCARDABLE     "matthew.bmp"
```

MATTHEW.BMP2003208BITMASK

BITMASK

WM_CREATEBITMASKLoadBitmaphBitmapImagGetObject hdcMemImag

hBitmapMaskhdcMemMaskGDI

```
SelectObject (hdcMemMask, GetStockObject (BLACK_BRUSH)) ;
Rectangle (hdcMemMask, 0, 0, cxBitmap, cyBitmap) ;
SelectObject (hdcMemMask, GetStockObject (WHITE_BRUSH)) ;
```

```
Ellipse (hdcMemMask, 0, 0, cxBitmap, cyBitmap) ;
```

01

BitBlt

```
BitBlt (hdcMemImag, 0, 0, cxBitmap, cyBitmap,
                    hdcMemMask, 0, 0, SRCAND) ;
```

SRCANDAND

WM_PAINTBitBltBitBlt

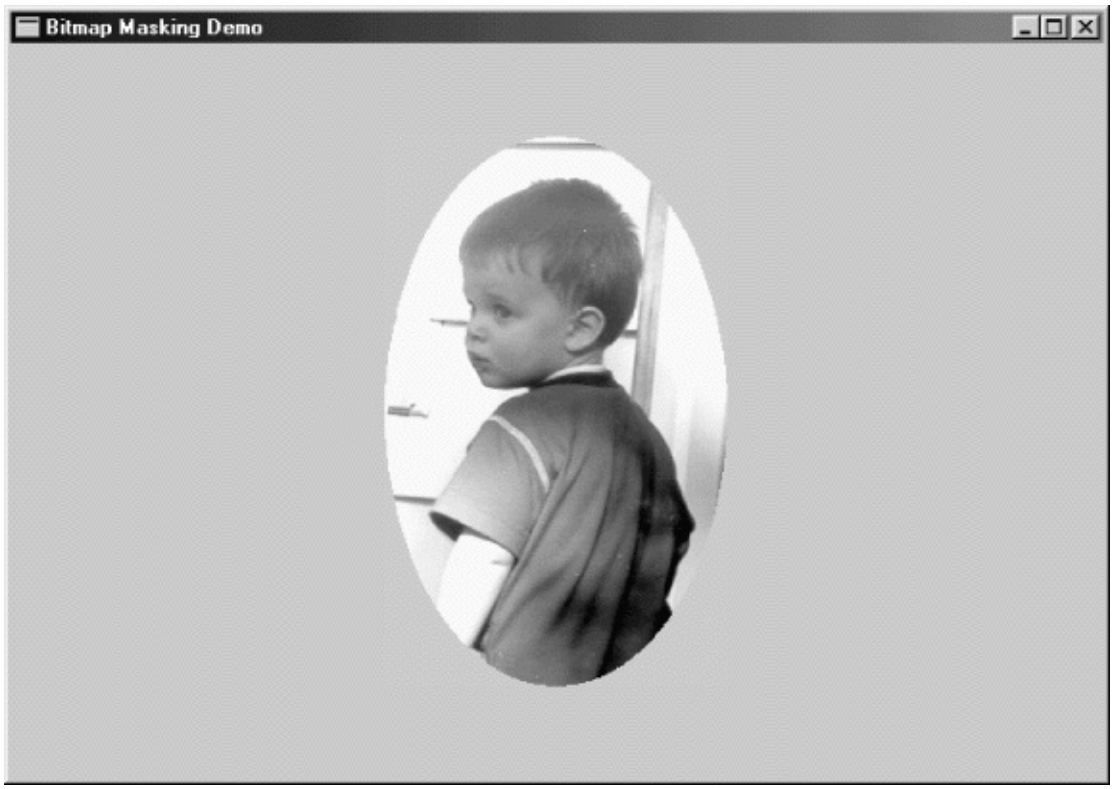
```
BitBlt (hdc, x, y, cxBitmap, cyBitmap, hdcMemMask, 0, 0, 0x220
```

D & ~S01AND1AND 0ANDBitBlt

BitBlt

```
BitBlt (hdc, x, y, cxBitmap, cyBitmap, hdcMemImag, 0, 0, SRCPA
```

OR14-9



14-9 BITMASK

Windows NTMASKBITMaskBltWindows PlgBltparallelogram blt

BITMASK1625616256

Windows

BOUNCE14-10GDIBitBlt

14-10 BOUNCE

```
BOUNCE.C

/*------------------------------------------------------------------------
  BOUNCE.C --   Bouncing Ball Program
                                                (c) Charles Petzold, 1998
------------------------------------------------------------------------*/

#include <windows.h>

#define ID_TIMER    1

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                                PSTR szCmdLine, int iCr
{
          static TCHAR szAppName[] = TEXT ("Bounce") ;
          HWND                         hwnd ;
          MSG                          msg ;
          WNDCLASS                         wndclass ;

         wndclass.style                        = CS_HREDRAW | CS_VR
          wndclass.lpfnWndProc                      = WndProc ;
          wndclass.cbClsExtra                       = 0 ;
```

```
     wndclass.cbWndExtra                    = 0 ;
     wndclass.hInstance                     = hInstance ;
     wndclass.hIcon                         = LoadIcon (NULL, IDI_
     wndclass.hCursor                       = LoadCursor (NULL,
     wndclass.hbrBackground                 = (HBRUSH) GetStockO
     wndclass.lpszMenuName                  = NULL ;
     wndclass.lpszClassName                 = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (  NULL, TEXT ("This program requires W
                                             szAppName, MB_IC
          return 0 ;
     }

     hwnd = CreateWindow ( szAppName, TEXT ("Bouncing Ba
                         WS_OVERLAPPEDWINDOW,
                         CW_USEDEFAULT, CW_USEDEFAULT,
```

```
                    CW_USEDEFAULT, CW_USEDEFAULT,
                     NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;
     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT iMsg, WPARAM
{
     static HBITMAP hBitmap ;
     static int     cxClient, cyClient, xCenter, yCenter, c
          cxRadius,           cyRadius, cxMove, cyMove, x
     HBRUSH              hBrush ;
```

```
     HDC                         hdc, hdcMem ;
     int                         iScale ;

     switch (iMsg)
     {
     case   WM_CREATE:
               hdc = GetDC (hwnd) ;
               xPixel = GetDeviceCaps (hdc, ASPECTX) ;
               yPixel = GetDeviceCaps (hdc, ASPECTY) ;
               ReleaseDC (hwnd, hdc) ;

               SetTimer (hwnd, ID_TIMER, 50, NULL) ;
               return 0 ;

     case   WM_SIZE:
               xCenter = (cxClient = LOWORD (lParam)) / 2 ;
               yCenter = (cyClient = HIWORD (lParam)) / 2 ;

               iScale = min (cxClient * xPixel, cyClient * yPixel)
```

```
     cxRadius = iScale / xPixel ;
     cyRadius = iScale / yPixel ;

     cxMove = max (1, cxRadius / 2) ;
     cyMove = max (1, cyRadius / 2) ;

     cxTotal = 2 * (cxRadius + cxMove) ;
     cyTotal = 2 * (cyRadius + cyMove) ;

     if (hBitmap)
                         DeleteObject (hBitmap) ;
     hdc = GetDC (hwnd) ;
     hdcMem = CreateCompatibleDC (hdc) ;
     hBitmap = CreateCompatibleBitmap (hdc, cxTot
    ReleaseDC (hwnd, hdc) ;

     SelectObject (hdcMem, hBitmap) ;
```

```
                Rectangle (hdcMem, -1, -1, cxTotal + 1, cyTotal

                hBrush = CreateHatchBrush (HS_DIAGCROSS, 0
                SelectObject (hdcMem, hBrush) ;
                SetBkColor (hdcMem, RGB (255, 0, 255)) ;
                Ellipse (hdcMem, cxMove, cyMove, cxTotal - cxM
                DeleteDC (hdcMem) ;
                DeleteObject (hBrush) ;
                return 0 ;

        case   WM_TIMER:
                if (!hBitmap)
                                break ;

                hdc = GetDC (hwnd) ;
                hdcMem = CreateCompatibleDC (hdc) ;
                SelectObject (hdcMem, hBitmap) ;

                BitBlt (hdc, xCenter - cxTotal / 2,
```

```
                                        yCenter - cyTotal / 2, cxTotal
                                        hdcMem, 0, 0, SRCCOPY) ;

                ReleaseDC (hwnd, hdc) ;
               DeleteDC (hdcMem) ;

                xCenter += cxMove ;
                yCenter += cyMove ;

                if ((xCenter + cxRadius >= cxClient) || (xCenter
                                cxMove = -cxMove ;

               if ((yCenter + cyRadius >= cyClient) || (yCenter
                                cyMove = -cyMove ;

                return 0 ;

      case   WM_DESTROY:
```

```
                    if (hBitmap)

                                DeleteObject (hBitmap) ;


                    KillTimer (hwnd, ID_TIMER) ;

                    PostQuitMessage (0) ;

                    return 0 ;

   }

        return DefWindowProc (hwnd, iMsg, wParam, lParam) ;

}
```

BOUNCEWM_SIZE

```
hdcMem = CreateCompatibleDC (hdc) ;
```

1.5

```
hBitmap = CreateCompatibleBitmap (hdc, cxTotal, cyTotal) ;
```

```
Rectangle (hdcMem, -1, -1, xTotal + 1, yTotal + 1) ;
```

```
Ellipse (hdcMem, xMove, yMove, xTotal - xMove, yTotal - yMove)
```

```
```

BitBltSRCCOPYROP

```
BitBlt (hdc, xCenter - cxTotal / 2, yCenter - cyTotal / 2, cxTotal, cy
                hdcMem, 0, 0, SRCCOPY) ;
```

BOUNCEROPSRCINVERTWindows
AnimatePaletteCreateDIBSectionGDIDirectX

SCRAMBLE14-11BitBlt

```
14-11  SCRAMBLE

SCRAMBLE.C

/*------------------------------------------------------------------------
  SCRAMBLE.C --   Scramble (and Unscramble) Screen
                                          (c) Charles Petzold, 1998
------------------------------------------------------------------------*/

#include <windows.h>

#define NUM 300

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
```

```
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                          PSTR szCmdLine, int iCm
{
     static int          iKeep [NUM][4] ;
     HDC                      hdcScr, hdcMem ;
     int                     cx, cy ;
     HBITMAP                 hBitmap ;
     HWND                         hwnd ;
     int                     i, j, x1, y1, x2, y2 ;

     if (LockWindowUpdate (hwnd = GetDesktopWindow ()))
     {
          hdcScr = GetDCEx (hwnd, NULL, DCX_CACHE | DCX_
          hdcMem = CreateCompatibleDC (hdcScr) ;
          cx     = GetSystemMetrics (SM_CXSCREEN) / 10 ;
          cy     = GetSystemMetrics (SM_CYSCREEN) / 10 ;
          hBitmap = CreateCompatibleBitmap (hdcScr, cx, cy)

          SelectObject (hdcMem, hBitmap) ;
```

```
srand ((int) GetCurrentTime ()) ;


 for (i = 0 ; i < 2   ; i++)
 for (j = 0 ; j < NUM ; j++)
{
          if (i == 0)
          {
          iKeep [j] [0] = x1 = cx * (rand () % 10) ;
          iKeep [j] [1] = y1 = cy * (rand () % 10) ;
          iKeep [j] [2] = x2 = cx * (rand () % 10) ;
         iKeep [j] [3] = y2 = cy * (rand () % 10) ;
          }
         else
     {
               x1 = iKeep [NUM - 1 - j] [0] ;
               y1 = iKeep [NUM - 1 - j] [1] ;
               x2 = iKeep [NUM - 1 - j] [2] ;
               y2 = iKeep [NUM - 1 - j] [3] ;
         }
```

```
                    BitBlt (hdcMem,  0,  0, cx, cy, hdcScr, x1, y
                    BitBlt (hdcScr, x1, y1, cx, cy, hdcScr, x2, y2
                   BitBlt (hdcScr, x2, y2, cx, cy, hdcMem,  0,  0

               Sleep (10) ;
          }

          DeleteDC (hdcMem) ;
          ReleaseDC (hwnd, hdcScr) ;
          DeleteObject (hBitmap) ;

          LockWindowUpdate (NULL) ;
     }
     return FALSE ;
}
```

SCRAMBLEWinMainLockWindowUpdateSCRAMBLE
DCX_LOCKWINDOWUPDATEGetDCExSCRAMBLE

SCRAMBLE10cxcy

CrandSCRAMBLEcxcyBitBltBitBlt

SCRAMBLE300SCRAMBLE!

hBitmapBITMAP

```
GetObject (hBitmap, sizeof (BITMAP), &bm) ;
```



```
hBitmap2 = CreateBitmap (  bm.bmWidth / 2, bm.bmHeight / 2,
                    bm.bmPlanes, bm.bmBitsPixel, NULL) ;
```

```
hdcMem1 = CreateCompatibleDC (hdc) ;
hdcMem2 = CreateCompatibleDC (hdc) ;
SelectObject (hdcMem1, hBitmap) ;
SelectObject (hdcMem2, hBitmap2) ;
```

```
BitBlt (    hdcMem2, 0, 0, bm.bmWidth / 2, bm.bmHeight / 2,
                    hdcMem1, 0, 0, SRCCOPY) ;
```

```
DeleteDC (hdcMem1) ;
```

```
DeleteDC (hdcMem2) ;

DeleteObject (hBitmap) ;
```

BLOWUP.C14-21BLOWUPWM_PAINT14-
12

```
14-12  BLOWUP

BLOWUP.C

/*------------------------------------------------------------------------

  BLOWUP.C --   Video Magnifier Program

                                          (c) Charles Petzold, 1998

------------------------------------------------------------------------*/

#include <windows.h>

#include <stdlib.h>                                  // for abs definition

#include "resource.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns

                                                PSTR szCmdLine, int iCr

{

          static TCHAR         szAppName [] = TEXT ("Blowup") ;
```

```
     HACCEL                  hAccel ;
     HWND                    hwnd ;
     MSG                     msg ;
     WNDCLASS                wndclass ;

     wndclass.style                  = CS_HREDRAW | CS_V
     wndclass.lpfnWndProc              = WndProc ;
     wndclass.cbClsExtra               = 0 ;
     wndclass.cbWndExtra               = 0 ;
     wndclass.hInstance                = hInstance ;
     wndclass.hIcon                    = LoadIcon (NULL, IDI_
     wndclass.hCursor                  = LoadCursor (NULL,
     wndclass.hbrBackground            = (HBRUSH) GetStockO
     wndclass.lpszMenuName             = szAppName ;
     wndclass.lpszClassName            = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (  NULL, TEXT ("This program requires W
```

```
                                                szAppName, MB_IC
     return 0 ;
}

hwnd = CreateWindow ( szAppName, TEXT ("Blow-Up Mo
                    WS_OVERLAPPEDWINDOW,
                       CW_USEDEFAULT, CW_USEDEFAULT,
                       CW_USEDEFAULT, CW_USEDEFAULT,
                       NULL, NULL, hInstance, NULL) ;

ShowWindow (hwnd, iCmdShow) ;
UpdateWindow (hwnd) ;

hAccel = LoadAccelerators (hInstance, szAppName) ;
while (GetMessage (&msg, NULL, 0, 0))
{
     if (!TranslateAccelerator (hwnd, hAccel, &msg))
     {
                       TranslateMessage (&msg) ;
```

```
                    DispatchMessage (&msg) ;
               }
          }
          return msg.wParam ;
}

void InvertBlock (HWND hwndScr, HWND hwnd, POINT ptBeg, PO
{
          HDC hdc ;
          hdc = GetDCEx (hwndScr, NULL, DCX_CACHE | DCX_LOCK
          ClientToScreen (hwnd, &ptBeg) ;
          ClientToScreen (hwnd, &ptEnd) ;
          PatBlt (hdc, ptBeg.x, ptBeg.y, ptEnd.x - ptBeg.x, ptEnd.y -
                              DSTINVERT) ;
          ReleaseDC (hwndScr, hdc) ;
}

HBITMAP CopyBitmap (HBITMAP hBitmapSrc)
{
          BITMAP              bitmap ;
```

```
        HBITMAP             hBitmapDst ;
        HDC                     hdcSrc, hdcDst ;

        GetObject (hBitmapSrc, sizeof (BITMAP), &bitmap) ;
        hBitmapDst = CreateBitmapIndirect (&bitmap) ;

        hdcSrc = CreateCompatibleDC (NULL) ;
        hdcDst = CreateCompatibleDC (NULL) ;

        SelectObject (hdcSrc, hBitmapSrc) ;
        SelectObject (hdcDst, hBitmapDst) ;

        BitBlt (hdcDst, 0, 0, bitmap.bmWidth, bitmap.bmHeight,
                            hdcSrc, 0, 0, SRCCOPY) ;
        DeleteDC (hdcSrc) ;
        DeleteDC (hdcDst) ;

        return hBitmapDst ;
}

LRESULT CALLBACK WndProc ( HWND hwnd, UINT message, WPA
```

```
{
        static BOOL          bCapturing, bBlocking ;
        static HBITMAP    hBitmap ;
        static HWND          hwndScr ;
        static POINT         ptBeg, ptEnd ;
        BITMAP               bm ;
        HBITMAP              hBitmapClip ;
        HDC                  hdc, hdcMem ;
        int                  iEnable ;
        PAINTSTRUCT          ps ;
        RECT                 rect ;

        switch (message)
        {
        case   WM_LBUTTONDOWN:
                    if (!bCapturing)
                    {
                                if (LockWindowUpdate (hwndScr = Get
                                 {
```

```
                    bCapturing = TRUE ;
                    SetCapture (hwnd) ;
                    SetCursor (LoadCursor (NULL, IDC
                }
                else
                    MessageBeep (0) ;
            }
            return 0 ;

    case   WM_RBUTTONDOWN:
            if (bCapturing)
            {
                bBlocking = TRUE ;
                ptBeg.x = LOWORD (lParam) ;
                ptBeg.y = HIWORD (lParam) ;
                ptEnd = ptBeg ;
                InvertBlock (hwndScr, hwnd, ptBeg, pt
            }
            return 0 ;
```

```
        case   WM_MOUSEMOVE:
                   if (bBlocking)
                   {
                              InvertBlock (hwndScr, hwnd, ptBeg, pt
                              ptEnd.x = LOWORD (lParam) ;
                              ptEnd.y = HIWORD (lParam) ;
                              InvertBlock (hwndScr, hwnd, ptBeg, pt
                   }
                  return 0 ;

        case   WM_LBUTTONUP:
        case   WM_RBUTTONUP:
                   if (bBlocking)
                   {
                                       InvertBlock (hwndScr, hwnd, ptBe
                                       ptEnd.x = LOWORD (lParam) ;
                                      ptEnd.y = HIWORD (lParam) ;

                                       if (hBitmap)
```

```
                    {
                        DeleteObject (hBitmap) ;
                        hBitmap = NULL ;
                    }

                    hdc = GetDC (hwnd) ;
                   hdcMem = CreateCompatibleDC (h
                    hBitmap= CreateCompatibleBitm
                abs (ptEnd.x - ptBeg.x),
                abs (ptEnd.y - ptBeg.y)) ;

            SelectObject (hdcMem, hBitmap) ;

            StretchBlt (hdcMem, 0, 0,          abs (ptEnd.x -
               abs (ptEnd.y - ptBeg.y),
        hdc, ptBeg.x, ptBeg.y, ptEnd.x - ptBeg.x,
                    ptEnd.y - ptBeg.y, SRCCOPY) ;
                    DeleteDC (hdcMem) ;
                    ReleaseDC (hwnd, hdc) ;
                    InvalidateRect (hwnd, NULL, TRUI
```

```
                }
                if (bBlocking || bCapturing)
                {
                        bBlocking = bCapturing = FALSE ;
                        SetCursor (LoadCursor (NULL, IDC_ARR
                        ReleaseCapture () ;
                        LockWindowUpdate (NULL) ;
                }
                return 0 ;

        case   WM_INITMENUPOPUP:
                iEnable =     IsClipboardFormatAavailable (CF_BITM
            MF_ENABLED : MF_GRAYED ;

                EnableMenuItem ((HMENU) wParam, IDM_EDIT_P

                iEnable = hBitmap ? MF_ENABLED : MF_GRAYED

                EnableMenuItem ((HMENU) wParam, IDM_EDIT_C
                EnableMenuItem ((HMENU) wParam, IDM_EDIT_C
                EnableMenuItem ((HMENU) wParam, IDM_EDIT_P
```

```
                return 0 ;

        case   WM_COMMAND:
                switch (LOWORD (wParam))
                {
                case IDM_EDIT_CUT:
                case IDM_EDIT_COPY:
                                if (hBitmap)
                                {
                        hBitmapClip = CopyBitmap (hBitmap) ;
                        OpenClipboard (hwnd) ;
                        EmptyClipboard () ;
                SetClipboardData (CF_BITMAP, hBitmapClip) ;
                                }
                                if (LOWORD (wParam) == IDM_ED
                                        return 0 ;
                //fall through for IDM_EDIT_CUT
                case   IDM_EDIT_DELETE:
                                if (hBitmap)
```

```
                    {
                        DeleteObject (hBitmap) ;
                        hBitmap = NULL ;
                    }
                        InvalidateRect (hwnd, NULL,
                        return 0 ;

            case   IDM_EDIT_PASTE:
                        if (hBitmap)
                        {
                    DeleteObject (hBitmap) ;
                    hBitmap = NULL ;
                }
                OpenClipboard (hwnd) ;
                hBitmapClip = GetClipboardData (CF_BITM

                if (hBitmapClip)
                        hBitmap = CopyBitmap (hBitmap

               CloseClipboard () ;
```

```
                InvalidateRect (hwnd, NULL, TRUE) ;
                return 0 ;
        }
        break ;
    case   WM_PAINT:
            hdc = BeginPaint (hwnd, &ps) ;

            if (hBitmap)
            {
                    GetClientRect (hwnd, &rect) ;

                    hdcMem = CreateCompatibleDC (hdc)
                    SelectObject (hdcMem, hBitmap) ;
                    GetObject (hBitmap, sizeof (BITMAP), (
                    SetStretchBltMode (hdc, COLORONCOL

                    StretchBlt (hdc,    0, 0, rect.right, rect.
          hdcMem, 0, 0, bm.bmWidth, bm.bmHeight, SRCCOP

                    DeleteDC (hdcMem) ;
```

```
                    }
                    EndPaint (hwnd, &ps) ;
                    return 0 ;

          case   WM_DESTROY:
                    if (hBitmap)
                                        DeleteObject (hBitmap) ;

               PostQuitMessage (0) ;
           return 0 ;
     }
          return DefWindowProc (hwnd, message, wParam, lParam)
}
```

```
BLOWUP.RC
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
//////////////////////////////////////////////////////////////////////////////
// Menu
```

```
BLOWUP MENU DISCARDABLE
BEGIN
  POPUP "&Edit"
  BEGIN
    MENUITEM "Cu&t\tCtrl+X",        IDM_EDIT_CUT
    MENUITEM "&Copy\tCtrl+C",       IDM_EDIT_COPY
    MENUITEM "&Paste\tCtrl+V",          IDM_EDIT_PASTE
    MENUITEM "De&lete\tDelete",         IDM_EDIT_DELETE
  END
END

//////////////////////////////////////////////////////////////////////////
// Accelerator
BLOWUP ACCELERATORS DISCARDABLE
BEGIN
  "C",      IDM_EDIT_COPY,     VIRTKEY, CONTROL, NOINVERT
  "V",      IDM_EDIT_PASTE,    VIRTKEY, CONTROL, NOINVERT
  VK_DELETE,      IDM_EDIT_DELETE,   VIRTKEY, NOINVERT
  "X",      IDM_EDIT_CUT,      VIRTKEY, CONTROL, NOINVERT
```

```
END
```

RESOURCE.H

```
// Microsoft Developer Studio generated include file.
// Used by Blowup.rc
#define IDM_EDIT_CUT        40001
#define IDM_EDIT_COPY       40002
#define IDM_EDIT_PASTE      40003
#define IDM_EDIT_DELETE     40004
```

14-10 BLOWUP

BLOWUP

1. BLOWUP+
2.
3. ()

BLOWUP

BLOWUPBLOWUP

BLOWUPBLOWUPWM_INITMENUPOPUPEdit
WM_COMMAND

CF_BITMAP GetCli
SetClipboardDataHBITMAPBLOWUPCopyBitmap
BITMAPCreateBitmapIndirect(SrcDst)

GetBitmapBitsSetBitmapBits

BLOWUPWindows

Windows GDIDDBWindowsDDBDDB
DDBWindows

Windows 3.0(DIB).GIF.JPEGInternetDIB.GIF.JPEG
DIBDIBDIB.GIF.JPEGWindows
DIBDIBDIBDIBDDB

DIB

DIBWindowsOS/21.1IBMMicrosoftOS/2
OS/2Presentation ManagerPMPresentation
GPI

Windows 3.01990OS/2DIBWindows
Windows 95Windows NT 4.0Windows 98Windows NT 5.0

DIB.BMP.DIBWindowsDIBDIB

DIB14packed DIBpa
WindowsDIBDIB

DIBGDIDIBDIB

DIBWindows APIDIBDIBAPIDIBGDI

DIBWindows24DIB2568DIBWindowsWindows

OS/2DIB

OS/2 1.1Windows DIB

DIB

-

-
- RGB
-

CWindowsWINGDI.Hpacked

-
- RGB
-

DIB

DIBpacked DIB14

```
typedef struct tagBITMAPFILEHEADER  // bmfh
{
        WORD        bfType ;       // signature word "BM" or 0x4D4
        DWORD        bfSize ;       // entire size of file
        WORD        bfReserved1 ;   // must be zero
        WORD        bfReserved2 ;   // must be zero
        DWORD       bfOffsetBits ;  // offset in file of DIB pixel bits
}
BITMAPFILEHEADER, * PBITMAPFILEHEADER ;
```

WINGDI.HbmfhpbmfhBITMAPFILEHEADER
PBITMAPFILEHEADER

14BMWORD0x4D42BMDWORDWORD0DIB
hot spotDWORDDIB

OS/2DIBBITMAPFILEHEADERBITMAPCOREHEADERDIB
DIBPacked DIBBITMAPCOREHEADER

```
typedef struct tagBITMAPCOREHEADER  // bmch
{
        DWORD        bcSize ;             // size of the structure = 12
        WORD         bcWidth ;            // width of image in pixels
        WORD         bcHeight ;           // height of image in pixels
        WORD         bcPlanes ;           // = 1
        WORD         bcBitCount ;         // bits per pixel (1, 4, 8, or
}
BITMAPCOREHEADER, * PBITMAPCOREHEADER ;
```

core

BITMAPCOREHEADERbcSize12

bcWidthbcHeightWORDDIB65,535

bcPlanes1Windows GDI

bcBitCountOS/2DIB14824DIB2bmch.bcBitCountC

```
1 << bmch.bcBitCount
```

bcBitCount

- 12DIB
- 416DIB
- 8256DIB
- 24full -Color DIB

8DIB8DIB

148BITMAPCOREHEADER24DIB3RGBTRIPLE

```
typedef struct tagRGBTRIPLE  // rgbt
{
        BYTE rgbtBlue ;       // blue level
        BYTE rgbtGreen ;      // green level
        BYTE rgbtRed ;        // red level
}
RGBTRIPLE ;
```

DIB

WINGDI.H

```
typedef struct tagBITMAPCOREINFO  // bmci
{
        BITMAPCOREHEADER      bmciHeader ;                // core-l
        RGBTRIPLE             bmciColors[1] ;             // color table
}
BITMAPCOREINFO, * PBITMAPCOREINFO ;
```

RGBTRIPLE1DIBRGBTRIPLE216256RGBTRIPLE8DIB
PBITMAPCOREINFO

```
pbmci = malloc (sizeof (BITMAPCOREINFO) + 255 * sizeof (RGBT
```

RGBTRIPLE

```
pbmci->bmciColors[i]
```

RGBTRIPLE3RGBTRIPLEDIBDIBRGBTRIPLEWORD

24DIB

DIBBITMAPCOREHEADERbcHeightDIB

DIBDIB

DIB

OS/2Presentation ManagerIBMPM

PMDIB

DIB

DIBDIBDIB

DIBBITMAPCOREHEADERbcHeightbcWidthbcBitCount
14824

4

```
RowLength = 4 * ((bmch.bcWidth * bmch.bcBitCount + 31) / 32)
```

C

```
RowLength = ((bmch.bcWidth * bmch.bcBitCount + 31) & ~31)
```

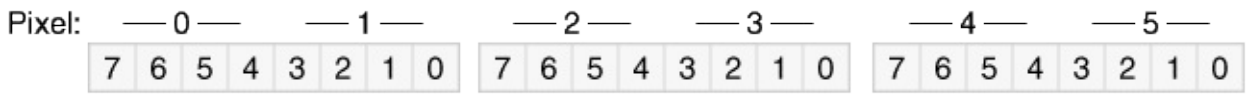
RowLengthbmch.bcHeight

700

1DIB8

010RGBTRIPLE1

4DIB4



401516

8DIB1

0255256

24DIB3RGBTRIPLE04

24DIB

Windows DIB

Windows 3.0OS/2DIBWindowsDIB

DIBBITMAPFILEHEADERBITMAPINFOHEADER
BITMAPCOREHEADER

```
typedef struct tagBITMAPINFOHEADER  // bmih

{

        DWORD biSize ;              // size of the structure = 40

       LONG  biWidth ;             // width of the image in pixels
```

```
        LONG  biHeight ;          // height of the image in pixels
        WORD  biPlanes ;           // = 1
        WORD  biBitCount ;         // bits per pixel (1, 4, 8, 16, 24,
        DWORD biCompression ;      // compression code
        DWORD biSizeImage ;        // number of bytes in image
        LONG  biXPelsPerMeter ;    // horizontal resolution
        LONG  biYPelsPerMeter ;    // vertical resolution
        DWORD biClrUsed ;          // number of colors used
        DWORD biClrImportant ;     // number of important colors
}
BITMAPINFOHEADER, * PBITMAPINFOHEADER ;
```

OS/2DIBWindows DIB1240

BITMAPINFOHEADERBITMAPCOREHEADER
BITMAPCOREHEADERbcWidthbcHeight16WORD
BITMAPINFOHEADER32LONG

BITMAPINFOHEADER148DIBRGBTRIPLE
BITMAPINFOHEADERRGBQUAD

```
typedef struct tagRGBQUAD  // rgb
{
```

```
        BYTE rgbBlue ;     // blue level
        BYTE rgbGreen ;    // green level
        BYTE rgbRed ;      // red level
        BYTE rgbReserved ; // = 0
}
RGBQUAD ;
```

0RGBTRIPLE WINGDI.H

```
typedef struct tagBITMAPINFO            // bmi
{
        BITMAPINFOHEADER bmiHeader ;     // info-header struct
        RGBQUAD                  bmiColors[1] ;  // color table array
}
BITMAPINFO, * PBITMAPINFO ;
```

BITMAPINFO32BITMAPINFOHEADER40RGBQUAD3232

BITMAPINFOHEADERWindows 3.0Windows 95Windows NT Windows 98Windows NT 5.0biHeightDIB1990DIB DIBbiHeightMicrosoft EditorDIBWord 97

biPlanes1biBitCount163214824Windows

biCompressionbiSizeImage

biXPelsPerMeterbiYPelsPerMeterpel--picture
WindowsDIBDIBDIB0722835
300 DPI11,811

biClrUsed48DIB16256DIBDIB64biClrUsed64
25664RGBQUAD0x000x3FDIB12biClrUsed0biBitCount

Windows 95biClrUsed162432DIBWindowsDIB256
DIBOS/224DIBWindows 3.0Windo
biClrUsed

- 1DIBbiClrUsed02
- 4DIBbiClrUsed016162151
- 8DIBbiClrUsed025625622251
- 162432DIBbiClrUsed00256DIB

DIB24DIB24DIB

biClrImportantbiClrUsed0biClrUsed0biClrUsedDIB
biClrImportant2568DIB

14824DIBOS/2DIB1632DIB

DIB

Windows3.0OS/2DIBDIB4DIBWindows1616

DIB88DIBDIBDIB8DIB

DIB64biClrUsed6400-00-0004-04-0408-08-080C-0C-0CRGB
F0-F0-F0F4-F4-F4F8-F8-F8FC-FC-FCRGB

```
rgb[i].rgbRed = rgb[i].rgbGreen = rgb[i].rgbBlue = i * 256 / 64 ;
```

rgbRGBQUADi063

```
rgb[i].rgbRed = rgb[i].rgbGreen = rgb[i].rgbBlue = i * 255 / 63 ;
```

FF-FF-FF

66416321531FF-FF-FF

8DIB64DIB256biClrUsed02562256biClrUsed28DIB
1DIB164DIBbiClrUsedbiClrUsed02561
biClrUsed648DIB0x000x3F

8DIBbiClrUsed640x000x2050%
0x3F

BITMAPINFOHEADERDIBDIB

8DIBbiClrUsed0256236Windows236DIB

biXPelsPerMeterbiYPelsPerMeterbiClrImportant0biClrUsed

DIB

BITMAPINFOHEADERbiCompressionbiSizeImage

biCompressionBI_RGBBI_RLE8BI_RLE4BI_BITFIELDS
WINGDI.H0348DIBrun-length1632DIBcolor
Windows 95

RLE

- 1DIBbiCompressionBI_RGB

- 4DIBbiCompressionBI_RGBBI_RLE4
- 8DIBbiCompressionBI_RGBBI_RLE8
- 24DIBbiCompressionBI_RGB

BI_RGBOS/2DIB

RLEDIBRLEDIBRLEDIB

8DIB15-1biCompressionBI_RGB8

15-1

1	**2**	**3**	**4**	
00	00			
00	01			
00	02	dx	dy	x+dxy+dy
00	n = 03FF			n
n = 01FF				n

DIBDIB12

```
0x05 0x27
```

```
0x27 0x27 0x27 0x27 0x27
```

DIB

```
0x00 0x06 0x45 0x32 0x77 0x34 0x59 0x90
```

```
0x45 0x32 0x77 0x34 0x59 0x90
```

2

```
0x00 0x05 0x45 0x32 0x77 0x34 0x59 0x00
```

```
0x45 0x32 0x77 0x34 0x59
```

DIBDIB

DIBDIBx,y0,0x1x0y

0x020x00xy0x000x00x0y0x010x00DIB

4DIB

n2n

```
0x07 0x35
```

```
0x35 0x35 0x35 0x3?
```

0x07 0x35

```
0x05 0x24
```

```
0x35 0x35 0x35 0x32 0x42 0x42
```

0x00 0x03

```
0x00 0x05 0x23 0x57 0x10 0x00
```

```
0x23 0x57 0x1?
```

biCompressionBI_RLE4BI_RLE8biSizeImageDIBbiCompression
BI_RGBbiSizeImage0biHeight

DIBDIBbiHeight

Color Masking

biCompressionWindows 951632DIBDIBbiCompressionBI_RGB BI_BITFIELDS3

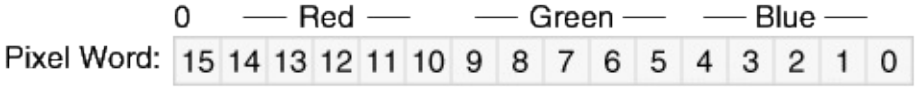
24DIBBI_RGBbiCompression

RGBTRIPLE4

biCompressionBI_RGB16DIB

5260

16



wPixel16

```
Red = ((0x7C00 & wPixel) >> 10) << 3 ;
```

```
Green          =     ((0x03E0 & wPixel)    >>          5)    << 3 ;
Blue           =     ((0x001F & wPixel)    >>          0)    << 3 ;
```

AND10500x000x1F30x000xF8

16DIB24

32DIBbiCompressionBI_RGB40RGBQUAD4

32

dwPixel32

```
Red            =     ((0x00FF0000 & dwPixel) >>         16) << 0
Green          =      ((0x0000FF00 & dwPixel) >>         8)    <<
Blue           =     ((0x000000FF & dwPixel) >>         0)    << (
```

0xFFWindows GDIRGB32CO

biCompressionBI_RGB1632DIBbiCompressionBI_BITFIELDS
DIBBITMAPINFOHEADER32CAND&1632
11

16DIBbiCompressionBI_BITFIELDSBITMAPINFOHEADER

```
0x0000F800

0x000007E0

0x0000001F
```

16DIB161dwMask[0]dwMask[1]dwMask[2]

```
int MaskToRShift (DWORD dwMask)
{
        int iShift ;
        if (   dwMask == 0)
                return 0 ;
        for    (      iShift = 0 ; !(dwMask & 1)  ; iShift++)
                        dwMask >>= 1 ;

        return iShift ;
}

int MaskToLShift (DWORD dwMask)
{
        int iShift ;
        if (   dwMask == 0)
```

```
                return 0 ;

        while (!(dwMask & 1))
                dwMask >>= 1 ;

        for (iShift = 0 ; dwMask & 1 ; iShift++)
                dwMask >>= 1 ;

        return 8 - iShift ;
}
```

MaskToRShift

```
iRShift[0] = MaskToRShift (dwMask[0]) ;
iRShift[1] = MaskToRShift (dwMask[1]) ;
iRShift[2] = MaskToRShift (dwMask[2]) ;
```

1150MaskToLShift

```
iLShift[0] = MaskToLShift (dwMask[0]) ;
iLShift[1] = MaskToLShift (dwMask[1]) ;
iLShift[2] = MaskToLShift (dwMask[2]) ;
```

```
Red          = ((dwMask[0] & wPixel) >> iRShift[0]) << iLShift[
Green         = ((dwMask[1] & wPixel) >> iRShift[1]) << iLShif
Blue         = ((dwMask[2] & wPixel) >> iRShift[2]) << iLShift[
```

0x0000FFFF16DIB32DIB

1632DIB25532DIB0320xFFFFFFFF

Windows NTWindows 95Windows 9815-2

15-2

16DIB	**16DIB**	**32DIB**
0x00007C00	0x0000F800	0x00FF0000
0x000003E0	0x000007E0	0x0000FF00
0x0000001F	0x0000001F	0x000000FF
5-5-5	5-6-5	8-8-8

biCompressionBI_RGB

4Header

Windows 95BITMAPINFOHEADERWindows
Windows 95Windows 4.0Windows NT

```
typedef struct
{
DWORD           bV4Size ;           // size of the structure = 120
LONG      bV4Width ;          // width of the image in pixels
LONG            bV4Height ;         // height of the image in pixels
WORD            bV4Planes ;         // = 1
WORD            bV4BitCount ;       // bits per pixel (1, 4, 8, 16, 2
DWORD           bV4Compression ;    // compression code
DWORD           bV4SizeImage ;      // number of bytes in imag
LONG            bV4XPelsPerMeter ;  // horizontal resolution
LONG            bV4YPelsPerMeter ;  // vertical resolution
DWORD           bV4ClrUsed ;        // number of colors used
DWORD           bV4ClrImportant ;       // number of important c
DWORD           bV4RedMask ;        // Red color mask
DWORD           bV4GreenMask ;      // Green color mask
DWORD           bV4BlueMask ;       // Blue color mask
```

```
DWORD           bV4AlphaMask ;       // Alpha mask
DWORD           bV4CSType ;          // color space type
CIEXYZTRIPLE     bV4Endpoints ;       // XYZ values
DWORD           bV4GammaRed ;        // Red gamma value
DWORD           bV4GammaGreen ;      // Green gamma value
DWORD           bV4GammaBlue ;       // Blue gamma value
}
BITMAPV4HEADER, * PBITMAPV4HEADER ;
```

11BITMAPINFOHEADER5Windows 95\
BITMAPV4HEADERBITMAPINFOHEADER
BITMAPV5HEADER

bV4CompressionBI_BITFIELDSbV4RedMaskbV4GreenMask
bV4BlueMask1632DIBBITMAPINFOHEADERDIB
bV4AlphaMask

BITMAPV5HEADERWindowsImage Color Manag

RGBRGB255,0,0RGB128,0,0
CMYcyan-magenta-yellow--CMYKcyan-magenta-yellow-
black---RGBCMYCMYK

380nm780nm1931Commission
L'Eclairage (International Commission on Illumination)CIExyz
5nmCIE Publication 15.2-1986ColorimetrySecond Edition2.1

SXYZ

$$X=\sum_{\lambda=380}^{780} S(\lambda)\,\bar{x}(\lambda)$$

$$Y=\sum_{\lambda=380}^{780} S(\lambda)\,\bar{y}(\lambda)$$

$$Z=\sum_{\lambda=380}^{780} S(\lambda)\,\bar{z}(\lambda)$$

XY **Z**y380nm780nm0YCIE

BITMAPV5HEADERbV4CSTypeLCS_CALIBRATED_RGB0

CIEXYZTRIPLE

```
typedef struct tagCIEXYZTRIPLE
{
        CIEXYZ  ciexyzRed ;
        CIEXYZ  ciexyzGreen ;
        CIEXYZ  ciexyzBlue ;
}
CIEXYZTRIPLE, * LPCIEXYZTRIPLE ;
```

CIEXYZ

```
typedef struct tagCIEXYZ
```

```
{
        FXPT2DOT30 ciexyzX ;
        FXPT2DOT30 ciexyzY ;
        FXPT2DOT30 ciexyzZ ;
}
CIEXYZ, * LPCIEXYZ ;
```

FXPT2DOT302300x400000001.00x480000001.50xFFFFFFFF
4.0

bV4EndpointsRGB255,0,00,255,00,0,255XYZDIBRGB

BITMAPV4HEADERγDIB0225
IV

$$I = (V + \varepsilon)^{\gamma}$$

ε0γγ2.5

0.452.2

YCIECIEL*

$$L^* = \begin{cases} 903.3\,\frac{Y}{Y_n} & \frac{Y}{Y_n} \le 0.008856 \\ 116\left(\frac{Y}{Y_n}\right)^{\frac{1}{3}} - 16 & 0.008856 < \frac{Y}{Y_n} \end{cases}$$

YnL*

(P)02550.01.00

$$I = V^r = \left(\frac{P}{255}\right)^r$$

γ2.5(L*)0100

$$L^* = 100\left(\frac{P}{255}\right)^{\frac{r}{3}}$$

0.851CIE

BITMAPV4HEADERDIB16160x100001.0DIB2.2

0x23333DIBCIE

5Header

Windows 98Windows NT 5.0(Windows 2000)BITMAPV5HEADERDIB

```
typedef struct
{
DWORD        bV5Size ;        // size of the structure = 120
LONG         bV5Width ;       // width of the image in pixels
LONG         bV5Height ;      // height of the image in pixels
WORD         bV5Planes ;      // = 1
WORD         bV5BitCount ;    // bits per pixel (1,4,8,16,24,
DWORD        bV5Compression ; // compression code
DWORD        bV5SizeImage ;   // number of bytes in imag
LONG         bV5XPelsPerMeter ; // horizontal resolution
LONG         bV5YPelsPerMeter ; // vertical resolution
DWORD        bV5ClrUsed ;     // number of colors used
DWORD        bV5ClrImportant ; // number of important col
DWORD        bV5RedMask ;     // Red color mask
DWORD        bV5GreenMask ;   // Green color mask
DWORD        bV5BlueMask ;    // Blue color mask
```

```
DWORD           bV5AlphaMask ;       // Alpha mask
DWORD           bV5CSType ;          // color space type
CIEXYZTRIPLE     bV5Endpoints ;       // XYZ values
DWORD           bV5GammaRed ;        // Red gamma value
DWORD           bV5GammaGreen ;      // Green gamma value
DWORD          bV5GammaBlue ;       // Blue gamma value
DWORD           bV5Intent ;          // rendering intent
DWORD           bV5ProfileData ;     // profile data or filename
DWORD           bV5ProfileSize ;     // size of embedded data or
DWORD           bV5Reserved ;
}
BITMAPV5HEADER, * PBITMAPV5HEADER ;
```

ICC Profile Format SpecificationIı
Consortium(AdobeAgfaAppleKodakMicrosoftSilicon Graphics
Sun Microsystems) http://www.icc.org
RGBCMYKCIE XYZ.ICMimag
managementDIBDIBDIB/Platform
Services/Color ManagementWindows

BITMAPV5HEADERbV5CSTypeLCS_CALIBRATED_RGB
BITMAPV4HEADERbV5Endpoints

bV5CSTypeLCS_sRGBRGBMicrosoftHewlett-Packard

Internethttp://www.color.org/contrib/sRGB.html

bV5CSTypeLCS_Windows_COLOR_SPACEWindowsAPI

bV5CSTypePROFILE_EMBEDDEDDIBICCPROFILE_LINKEDDIB ICCbV5ProfileDataBITMAPV5HEADERbV5ProfileSize bV5Endpoints

DIB

DIBDIB15-1

15-1　DIBHEADS

```
DIBHEADS.C

/*------------------------------------------------------------------------

  DIBHEADS.C -- Displays DIB Header Information

                    (c) Charles Petzold, 1998

------------------------------------------------------------------------*/

#include <windows.h>

#include "resource.h"


LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

TCHAR szAppName[] = TEXT ("DibHeads") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns

                PSTR szCmdLine, int iCmdShow)
```

```
{
        HACCEL             hAccel ;
        HWND               hwnd ;
        MSG                msg ;
  WNDCLASS           wndclass ;

        wndclass.style                   = CS_HREDRAW | CS_V
        wndclass.lpfnWndProc             = WndProc ;
        wndclass.cbClsExtra              = 0 ;
        wndclass.cbWndExtra              = 0 ;
        wndclass.hInstance               = hInstance ;
        wndclass.hIcon                   = LoadIcon (NULL, IDI_APPL
        wndclass.hCursor                 = LoadCursor (NULL, IDC_A
        wndclass.hbrBackground           = (HBRUSH) GetStockO
        wndclass.lpszMenuName            = szAppName ;
        wndclass.lpszClassName           = szAppName ;

        if (!RegisterClass (&wndclass))
        {
             MessageBox (NULL, TEXT ("This program requires Wi
```

```
                    szAppName, MB_ICONERROR) ;
     return 0 ;
}

hwnd = CreateWindow (szAppName, TEXT ("DIB Headers
                    WS_OVERLAPPEDWINDOW,
                    CW_USEDEFAULT, CW_USEDEFAULT,
                    CW_USEDEFAULT, CW_USEDEFAULT,
                    NULL, NULL, hInstance, NULL) ;

ShowWindow (hwnd, iCmdShow) ;
UpdateWindow (hwnd) ;

hAccel = LoadAccelerators (hInstance, szAppName) ;
while (GetMessage (&msg, NULL, 0, 0))
{
     if (!TranslateAccelerator (hwnd, hAccel, &msg))
     {
                    TranslateMessage (&msg) ;
```

```
                    DispatchMessage (&msg) ;
               }
          }
          return msg.wParam ;
}

void Printf (HWND hwnd, TCHAR * szFormat, ...)
{
          TCHAR               szBuffer [1024] ;
          va_list       pArgList ;

          va_start (pArgList, szFormat) ;
          wvsprintf (szBuffer, szFormat, pArgList) ;
          va_end (pArgList) ;

          SendMessage (hwnd, EM_SETSEL, (WPARAM) -1, (LPARAM
          SendMessage (hwnd, EM_REPLACESEL, FALSE, (LPARAM) 
          SendMessage (hwnd, EM_SCROLLCARET, 0, 0) ;
}
void DisplayDibHeaders (HWND hwnd, TCHAR * szFileName)
```

```
{
static TCHAR   * szInfoName []= { TEXT ("BITMAPCOREHEADER"
                          TEXT ("BITMAPINFOHEADER"),
                          TEXT ("BITMAPV4HEADER"),
                          TEXT ("BITMAPV5HEADER") } ;
Static TCHAR  * szCompression []={TEXT ("BI_RGB"),
TEXT    ("BI_RLE8"),
                          TEXT ("BI_RLE4"),
                          TEXT ("BI_BITFIELDS"),
                          TEXT ("unknown") } ;
   BITMAPCOREHEADER      *      pbmch ;
   BITMAPFILEHEADER      *      pbmfh ;
   BITMAPV5HEADER*      pbmih ;
   BOOL                        bSuccess ;
   DWORD               dwFileSize, dwHighSize, dwBytesRead ;
   HANDLE              hFile ;
  int               i ;
  PBYTE             pFile ;
```

```
TCHAR           * szV ;

                // Display the file name

      Printf (hwnd, TEXT ("File: %s\r\n\r\n"), szFileName) ;
                // Open the file
      hFile = CreateFile (szFileName, GENERIC_READ, FILE_SHA
   OPEN_EXISTING, FILE_FLAG_SEQUENTIAL_SCAN, NULL) ;
      if (hFile == INVALID_HANDLE_VALUE)
      {
                Printf (hwnd, TEXT ("Cannot open file.\r\n\r\n"))
                return ;
 }

                // Get the size of the file
      dwFileSize = GetFileSize (hFile, &dwHighSize) ;
      if (dwHighSize)
      {
           Printf (hwnd, TEXT ("Cannot deal with >4G files.\r\n\r
           CloseHandle (hFile) ;
```

```
            return ;
}
                    // Allocate memory for the file
      pFile = malloc (dwFileSize) ;
      if (!pFile)
      {
            Printf (hwnd, TEXT ("Cannot allocate memory.\r\n\r\n"
            CloseHandle (hFile) ;
            return ;
      }

                        // Read the file
      SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
      ShowCursor (TRUE) ;

      bSuccess = ReadFile (hFile, pFile, dwFileSize, &dwBytesRe
      ShowCursor (FALSE) ;
      SetCursor (LoadCursor (NULL, IDC_ARROW)) ;

      if (!bSuccess || (dwBytesRead != dwFileSize))
```

```
{
            Printf (hwnd, TEXT ("Could not read file.\r\n\r\n")
             CloseHandle (hFile) ;
       free (pFile) ;
      return ;
}

                    // Close the file
CloseHandle (hFile) ;
                    // Display file size
Printf (hwnd, TEXT ("File size = %u bytes\r\n\r\n"), dwFileS
                    // Display BITMAPFILEHEADER structure
pbmfh = (BITMAPFILEHEADER *) pFile ;
Printf (hwnd, TEXT   ("BITMAPFILEHEADER\r\n")) ;
Printf (hwnd, TEXT   ("\t.bfType = 0x%X\r\n"), pbmfh->bfT
Printf (hwnd, TEXT   ("\t.bfSize = %u\r\n"), pbmfh->bfSize
Printf (hwnd, TEXT   ("\t.bfReserved1 = %u\r\n"), pbmfh->
Printf (hwnd, TEXT   ("\t.bfReserved2 = %u\r\n"), pbmfh->
Printf (hwnd, TEXT   ("\t.bfOffBits = %u\r\n\r\n"), pbmfh->
```

```
                        // Determine which information structure w

        pbmih = (BITMAPV5HEADER *) (pFile + sizeof (BITMAPFIL
        switch (pbmih->bV5Size)
        {
       case   sizeof (BITMAPCOREHEADER):i= 0 ; break ;
        case   sizeof (BITMAPINFOHEADER): i= 1 ; szV=
TEXT ("i") ;       break ;
        case   sizeof (BITMAPV4HEADER):i= 2 ; szV=
TEXT ("V4") ;      break ;
        case   sizeof (BITMAPV5HEADER):i= 3 ; szV=
TEXT ("V5") ;      break ;
        default:
Printf (hwnd,      TEXT ("Unknown header size of %u.\r\n\r\n"),
                   pbmih->bV5Size) ;
              free (pFile) ;
              return ;
        }
```

```
        Printf (hwnd, TEXT ("%s\r\n"), szInfoName[i]) ;

                                // Display the BITMAPCOREHEADER fiel

        if (pbmih->bV5Size == sizeof (BITMAPCOREHEADER))

        {

              pbmch = (BITMAPCOREHEADER *) pbmih ;

Printf(hwnd,TEXT("\t.bcSize = %u\r\n"), pbmch->bcSize) ;

Printf(hwnd,TEXT("\t.bcWidth = %u\r\n"), pbmch->bcWidth) ;

Printf(hwnd,TEXT("\t.bcHeight = %u\r\n"), pbmch->bcHeight) ;

Printf(hwnd,TEXT("\t.bcPlanes = %u\r\n"), pbmch->bcPlanes) ;

Printf(hwnd,TEXT("\t.bcBitCount = %u\r\n\r\n"), pbmch->bcBitCo

    free (pFile) ;

    return ;

   }

                    // Display the BITMAPINFOHEADER fields

Printf(hwnd,TEXT("\t.b%sSize = %u\r\n"), szV, pbmih->bV5Size)

Printf(hwnd,TEXT("\t.b%sWidth = %i\r\n"), szV, pbmih->bV5Widt

Printf(hwnd,TEXT("\t.b%sHeight = %i\r\n"), szV, pbmih->bV5Hei

Printf(hwnd,TEXT("\t.b%sPlanes = %u\r\n"), szV, pbmih->bV5Pla
```

```
Printf(hwnd,TEXT("\t.b%sBitCount=%u\r\n"),szV, pbmih->bV5Bit
Printf(hwnd,TEXT("\t.b%sCompression = %s\r\n"), szV,
                    szCompression [min (4, pbmih->bV5Compressior
Printf(hwnd,TEXT("\t.b%sSizeImage= %u\r\n"),szV,
pbmih->bV5SizeImage) ;
Printf(hwnd,TEXT ("\t.b%sXPelsPerMeter = %i\r\n"), szV,
             pbmih->bV5XPelsPerMeter) ;
Printf(hwnd,TEXT ("\t.b%sYPelsPerMeter = %i\r\n"), szV,
                    pbmih->bV5YPelsPerMeter) ;
Printf      (hwnd, TEXT ("\t.b%sClrUsed = %i\r\n"), szV,
pbmih->bV5ClrUsed) ;
Printf      (hwnd, TEXT ("\t.b%sClrImportant = %i\r\n\r\n"), szV,
               pbmih->bV5ClrImportant) ;

         if (pbmih->bV5Size == sizeof (BITMAPINFOHEADER))
    {
                     if (pbmih->bV5Compression == BI_BITFIELDS)
        {
Printf (hwnd,TEXT("Red Mask = %08X\r\n"), pbmih->bV5RedMas
```

```
Printf (hwnd,TEXT ("Green Mask = %08X\r\n"), pbmih->bV5Gree
Printf (hwnd,TEXT ("Blue Mask  = %08X\r\n\r\n"), pbmih->bV5Bl
      }
            free (pFile) ;
            return ;
   }

                        // Display additional BITMAPV4HEADER fiel
       Printf (hwnd, TEXT ("\t.b%sRedMask   = %08X\r\n"), szV,
                                          pbmih->bV5RedMask) ;
       Printf (hwnd, TEXT ("\t.b%sGreenMask = %08X\r\n"), szV,
                                           pbmih->bV5GreenMask) ;
       Printf (hwnd, TEXT ("\t.b%sBlueMask  = %08X\r\n"), szV,
                                           pbmih->bV5BlueMask) ;
       Printf (hwnd, TEXT ("\t.b%sAlphaMask = %08X\r\n"), szV,
                                           pbmih->bV5AlphaMask) ;
       Printf (hwnd, TEXT ("\t.b%sCSType = %u\r\n"), szV,
                                           pbmih->bV5CSType) ;
  Printf (hwnd, TEXT ("\t.b%sEndpoints.ciexyzRed.ciexyzX = %08
```

```
                                        szV, pbmih->bV5Endpoints.ciexy
  Printf (hwnd, TEXT ("\t.b%sEndpoints.ciexyzRed.ciexyzY = %0
                                        szV, pbmih->bV5Endpoints.ciexy
  Printf (hwnd, TEXT ("\t.b%sEndpoints.ciexyzRed.ciexyzZ = %0
                                        szV, pbmih->bV5Endpoints.ciexy
  Printf (hwnd, TEXT ("\t.b%sEndpoints.ciexyzGreen.ciexyzX = %
                                    szV, pbmih->bV5Endpoints.ciexyzGree
  Printf (hwnd, TEXT ("\t.b%sEndpoints.ciexyzGreen.ciexyzY = %
                                        szV, pbmih->bV5Endpoints.ciexy
 Printf (hwnd, TEXT ("\t.b%sEndpoints.ciexyzGreen.ciexyzZ = %
                                        szV, pbmih->bV5Endpoints.ciexy
  Printf (hwnd, TEXT ("\t.b%sEndpoints.ciexyzBlue.ciexyzX = %0
                                    szV, pbmih->bV5Endpoints.ciexyzBlue
  Printf (hwnd, TEXT ("\t.b%sEndpoints.ciexyzBlue.ciexyzY = %0
                                    szV, pbmih->bV5Endpoints.ciexyzBlue
  Printf (hwnd, TEXT ("\t.b%sEndpoints.ciexyzBlue.ciexyzZ = %0
                                    szV, pbmih->bV5Endpoints.ciexyzBlue
  Printf (hwnd, TEXT ("\t.b%sGammaRed = %08X\r\n"), szV,
```

```
                                pbmih->bV5GammaRed) ;
Printf (hwnd, TEXT ("\t.b%sGammaGreen = %08X\r\n"), szV,
                                pbmih->bV5GammaGreen) ;
Printf (hwnd, TEXT ("\t.b%sGammaBlue  = %08X\r\n\r\n"), szV
                                pbmih->bV5GammaBlue) ;

if (pbmih->bV5Size == sizeof (BITMAPV4HEADER))
{
free (pFile) ;
return ;
}

      // Display additional BITMAPV5HEADER fields
Printf       (hwnd,       TEXT ("\t.b%sIntent = %u\r\n"), szV, pbr
Printf (hwnd, TEXT ("\t.b%sProfileData = %u\r\n"), szV,
                                    pbmih->bV5ProfileData) ;
Printf (hwnd, TEXT ("\t.b%sProfileSize = %u\r\n"), szV,
                                    pbmih->bV5ProfileSize) ;
Printf (hwnd, TEXT ("\t.b%sReserved = %u\r\n\r\n"), szV,
                                    pbmih->bV5Reserved) ;
```

```
   free (pFile) ;
  return ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
   static HWND                 hwndEdit ;
   static OPENFILENAME ofn ;
   static TCHAR                szFileName [MAX_PATH], szTitleName
   static TCHAR                szFilter[]= TEXT("Bitmap Files (*.BMP)
                               TEXT("All Files (*.*)\0*.*\0\0") ;

   switch (message)
   {
   case   WM_CREATE:
               hwndEdit = CreateWindow (TEXT ("edit"), NULL,
                           WS_CHILD | WS_VISIBLE | WS_BORDER |
                           WS_VSCROLL | WS_HSCROLL |
```

```
                    ES_MULTILINE | ES_AUTOVSCROLL | ES_REA
                    0, 0, 0, 0, hwnd, (HMENU) 1,
                    ((LPCREATESTRUCT) lParam)->hInstance,

ofn.lStructSize             = sizeof (OPENFILENAME) ;
ofn.hwndOwner         = hwnd ;
ofn.hInstance               = NULL ;
ofn.lpstrFilter             = szFilter ;
     ofn.lpstrCustomFilter = NULL ;
     ofn.nMaxCustFilter    = 0 ;
     ofn.nFilterIndex            = 0 ;
     ofn.lpstrFile               = szFileName ;
     ofn.nMaxFile                = MAX_PATH ;
    ofn.lpstrFileTitle           = szTitleName ;
     ofn.nMaxFileTitle           = MAX_PATH ;
     ofn.lpstrInitialDir         = NULL ;
     ofn.lpstrTitle              = NULL ;
     ofn.Flags                   = 0 ;
     ofn.nFileOffset             = 0 ;
```

```
            ofn.nFileExtension           = 0 ;
                ofn.lpstrDefExt       = TEXT ("bmp") ;
                ofn.lCustData         = 0 ;
                ofn.lpfnHook          = NULL ;
                ofn.lpTemplateName = NULL ;
                return 0 ;

case   WM_SIZE:
          MoveWindow (hwndEdit, 0, 0, LOWORD (lParam), HI
                return 0 ;

     case   WM_COMMAND:
                switch (LOWORD (wParam))
                {
                case   IDM_FILE_OPEN:
                                  if (GetOpenFileName (&ofn))
          DisplayDibHeaders (hwndEdit, szFileName) ;

                                  return 0 ;
                }
```

```
                break ;


        case   WM_DESTROY:
                    PostQuitMessage (0) ;
                    return 0 ;
   }
        return DefWindowProc (hwnd, message, wParam, lParam)
}
```

```
DIBHEADS.RC
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
//////////////////////////////////////////////////////////////////////////////
// Accelerator
DIBHEADS ACCELERATORS DISCARDABLE
BEGIN
  "O",          IDM_FILE_OPEN,          VIRTKEY, CONTROL, NOIN
END
```

```
//////////////////////////////////////////////////////////////////////
// Menu
DIBHEADS MENU DISCARDABLE
BEGIN
   POPUP "&File"
   BEGIN
                MENUITEM "&Open\tCtrl+O",     IDM_FILE_OPEN
   END
END
```

```
RESOURCE.H
// Microsoft Developer Studio generated include file.
// Used by DibHeads.rc
#define IDM_FILE_OPEN                40001
```

WndProcFile
DisplayDibHeadersDIB

WindowsDIB

SetDIBitsToDeviceset dee eye bits to deviceStretchDIBits

dee eye bitsDIBDIBSetDIBitsToDeviceDIB640×480DIB
VGA300dpi2.1×1.6StretchDIBitsDIB

DIB

DIBDIB

DIBpacked

DIBDIBDIBpPackedDibpacked
DIB

DIB

```
iWidth = ((PBITMAPINFOHEADER) pPackedDib)->biWidth ;
```

DIBOS/2packed DIBBITMAPCOREHEADERI
LONGDIB

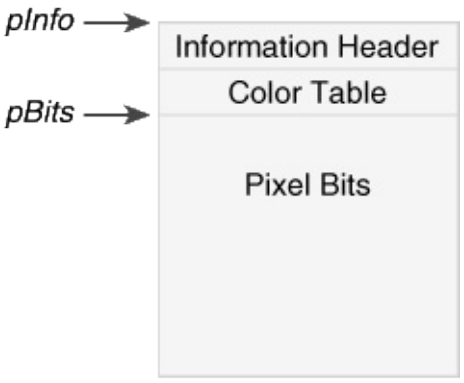
```
if (((PBITMAPCOREHEADER) pPackedDib)->bcSize == sizeof (BIT

        iWidth = ((PBITMAPCOREHEADER) pPackedDib)->bcWidth

else

        iWidth = ((PBITMAPINFOHEADER) pPackedDib)->biWidth ;
```

packed DIB5,27DIBOS/2DIB32DIB

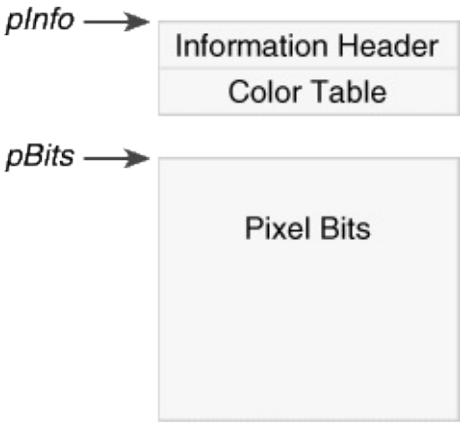
DIBpacked

SetDIBitsToDeviceStretchDIBitsDIBBITMAPINFOBITMAPINFO
BITMAPINFOHEADERpacked DIB

BITMAPFILEHEADERbfOffBitsbfOffBitsDIB
BITMAPINFOBITMAPFILEHEADERpacked
BITMAPFILEHEADER



SetDIBitsToDeviceStretchDIBitsDIB



DIBDIBpacked

SetDIBitsToDeviceStretchDIBitsDIBDIBDIB

SetDIBitsToDeviceDIBDIBDIBDIB

```
iLines =    SetDIBitsToDevice (
                        hdc,          // device context handle
                        xDst,         // x destination coordinate
```

```
                    yDst,          // y destination coordinate
                     cxSrc,         // source rectangle width
                     cySrc,         // source rectangle height
                     xSrc,          // x source coordinate
                     ySrc,          // y source coordinate
                     yScan,         // first scan line to draw
                     cyScans,       // number of scan lines to dra
                     pBits,         // pointer to DIB pixel bits
                     pInfo,         // pointer to DIB information
                    fClrUse) ;     // color use flag
```

GDISetDIBitsToDeviceDIBxDstyDstDIBDIB
Windows

DIBDIBDIBxSrcySrc0cxSrccySrcDIB
BITMAPINFOHEADERbiHeightDIBcySrcbiHeight

/Platform SDK/Graphics and Multimedia Services/GDI/Bitmaps/Bi
Reference/Bitmap Functions/SetDIBitsToDevicexSrcySrccxSrccySrc
DIBDIBcxSrccySrc

yScancyScanDIByScan0cyScanDIB

pBitsDIBpInfoDIBBITMAPINFOBITMAPINFO
BITMAPINFOHEADERSetDIBitsToDeviceBITMAPINFO148
DIBpInfoBITMAPINFOBITMAPCOREINFOBITMAPV4HEADER

BITMAPV5HEADER

DIB_RGB_COLORSDIB_PAL_COLORSWINGDI.H01
DIB_RGB_COLORSDIBDIB_PAL_COLORSDIB16
DIB_RGB_COLORS0

SetDIBitsToDevice

SetDIBitsToDeviceDIB

- **hdc**
- **xDstyDst**
- **cxDibcyDib**DIBcyDibBITMAPINFOHEADERbiHeight
- **pInfopBits**

SetDIBitsToDevice

```
SetDIBitsToDevice (
        hdc,          // device context handle
        xDst,         // x destination coordinate
  yDst,         // y destination coordinate
        cxDib,        // source rectangle width
        cyDib,        // source rectangle height
  0,            // x source coordinate
  0,            // y source coordinate
        0,            // first scan line to draw
```

```
  cyDib,        // number of scan lines to draw
  pBits,        // pointer to DIB pixel bits
  pInfo,   // pointer to DIB information
        0) ;          // color use flag
```

DIB120

15-2 SHOWDIB1SetDIBitsToDeviceDIB

15-2 SHOWDIB1

```
SHOWDIB1.C
/*------------------------------------------------------------------
  SHOWDIB1.C --         Shows a DIB in the client area
                        (c) Charles Petzold, 1998
------------------------------------------------------------------*/
#include <windows.h>
#include "dibfile.h"
#include "resource.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
TCHAR szAppName[] = TEXT ("ShowDib1") ;
```

```
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                    PSTR szCmdLine, int iCmdShow)
{
          HACCEL             hAccel ;
          HWND               hwnd ;
          MSG                     msg ;
          WNDCLASS           wndclass ;

  wndclass.style                    = CS_HREDRAW | CS_VREDR
          wndclass.lpfnWndProc              = WndProc ;
          wndclass.cbClsExtra         = 0 ;
          wndclass.cbWndExtra               = 0 ;
          wndclass.hInstance                = hInstance ;
          wndclass.hIcon                    = LoadIcon (NULL, IDI_APPLI
          wndclass.hCursor                  = LoadCursor (NULL, IDC_A
          wndclass.hbrBackground            = (HBRUSH) GetStockO
          wndclass.lpszMenuName             = szAppName ;
          wndclass.lpszClassName            = szAppName ;
```

```
     if (!RegisterClass (&wndclass))
{
          MessageBox (NULL, TEXT ("This program requires Wi
                    szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, TEXT ("Show DIB #1
               WS_OVERLAPPEDWINDOW,
               CW_USEDEFAULT, CW_USEDEFAULT,
               CW_USEDEFAULT, CW_USEDEFAULT,
               NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     hAccel = LoadAccelerators (hInstance, szAppName) ;
     while (GetMessage (&msg, NULL, 0, 0))
     {
               if (!TranslateAccelerator (hwnd, hAccel, &msg))
```

```
                {
                        TranslateMessage (&msg) ;
                        DispatchMessage (&msg) ;
            }
        }
  return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
        static BITMAPFILEHEADER      * pbmfh ;
        static BITMAPINFO            * pbmi ;
        static BYTE                  * pBits ;
        static int                   cxClient, cyClient, cxDib, cyDib ;
        static       TCHAR           szFileName [MAX_PATH], szTitleN
        BOOL                         bSuccess ;
        HDC                          hdc ;
        PAINTSTRUCT                  ps ;

        switch (message)
```

```
{
case   WM_CREATE:
           DibFileInitialize (hwnd) ;
           return 0 ;

case   WM_SIZE:
           cxClient = LOWORD (lParam) ;
           cyClient = HIWORD (lParam) ;
           return 0 ;

case   WM_INITMENUPOPUP:
           EnableMenuItem ((HMENU) wParam, IDM_FILE_S
      pbmfh ? MF_ENABLED : MF_GRAYED) ;
           return 0 ;

case   WM_COMMAND:
           switch (LOWORD (wParam))
           {
           case   IDM_FILE_OPEN:
                  // Show the File Open dialog box
```

```
                    if (!DibFileOpenDlg (hwnd, szFileN
                    return 0 ;

               // If there's an existing DIB, free the memo

                    if (pbmfh)
                    {
                    free (pbmfh) ;
                pbmfh = NULL ;
               }
                    // Load the entire DIB into memor

               SetCursor (LoadCursor (NULL, IDC_WA
               ShowCursor (TRUE) ;

               pbmfh = DibLoadImage (szFileName) ;
               ShowCursor (FALSE) ;
               SetCursor (LoadCursor (NULL, IDC_ARR

          // Invalidate the client area for later update
```

```
                    InvalidateRect (hwnd, NULL, TRUE) ;

                   if (pbmfh == NULL)
                    {
MessageBox (       hwnd, TEXT ("Cannot load DIB file"),
                                                szAppName, 0
                    return 0 ;
                   }
      // Get pointers to the info structure & the bits

                    pbmi  = (BITMAPINFO *) (pbmfh + 1) ;
                    pBits = (BYTE *) pbmfh + pbmfh->bfO

                    // Get the DIB width and height

  if (pbmi->bmiHeader.biSize == sizeof (BITMAPCOREH
                    {
     cxDib      = ((BITMAPCOREHEADER *) pbmi)->bcW
     cyDib      = ((BITMAPCOREHEADER *) pbmi)->bcH
                    }
```

```
                    else
                {
            cxDib  =      pbmi->bmiHeader.biWidth ;
             cyDib = abs(pbmi->bmiHeader.biHeight) ;
                    }
                    return 0 ;

                case   IDM_FILE_SAVE:
            // Show the File Save dialog box

                if (!DibFileSaveDlg (hwnd, szFileName, szTit
                    return 0 ;


                    // Save the DIB to memory

                    SetCursor (LoadCursor (NULL, IDC_WA
                   ShowCursor (TRUE) ;

                    bSuccess = DibSaveImage (szFileNam
                    ShowCursor (FALSE) ;
```

```
                    SetCursor (LoadCursor (NULL, IDC_ARR

                    if (!bSuccess)
          MessageBox (    hwnd, TEXT ("Cannot save DIB file"),
                    szAppName, 0) ;
                      return 0 ;
          }
          break ;

     case   WM_PAINT:
                    hdc = BeginPaint (hwnd, &ps) ;

                    if (pbmfh)
          SetDIBitsToDevice (hdc,
                      0,                  // xDst
                      0,                  // yDst
                      cxDib,                // cxSrc
                      cyDib,                // cySrc
                     0,                  // xSrc
                      0,                  // ySrc
```

```
                        0,                  // first scan line
                        cyDib,              // number of scan lines
                       pBits,
                        pbmi,
                        DIB_RGB_COLORS) ;
                    EndPaint (hwnd, &ps) ;
                    return 0 ;

        case   WM_DESTROY:
                    if (pbmfh)
                              free (pbmfh) ;

                    PostQuitMessage (0) ;
                    return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

DIBFILE.H

```
/*------------------------------------------------------------------------
```

```
  DIBFILE.H -- Header File for DIBFILE.C

-----------------------------------------------------------------------*/

void DibFileInitialize (HWND hwnd) ;

BOOL DibFileOpenDlg(HWND hwnd, PTSTR pstrFileName, PTSTR

BOOL DibFileSaveDlg(HWND hwnd, PTSTR pstrFileName, PTSTR

BITMAPFILEHEADER * DibLoadImage (PTSTR pstrFileName) ;

BOOL            DibSaveImage(PTSTR pstrFileName, BITMAPFILEHE
```

```
DIBFILE.C

/*-----------------------------------------------------------------------

DIBFILE.C -- DIB File Functions

-----------------------------------------------------------------------*/

#include <windows.h>

#include <commdlg.h>

#include "dibfile.h"


static OPENFILENAME ofn ;

void DibFileInitialize (HWND hwnd)

{
```

```
        static TCHAR szFilter[] = TEXT ("Bitmap Files (*.BMP)\0*.b
           TEXT ("All Files (*.*)\0*.*\0\0") ;
        ofn.lStructSize             = sizeof (OPENFILENAME) ;
        ofn.hwndOwner         = hwnd ;
        ofn.hInstance               = NULL ;
        ofn.lpstrFilter             = szFilter ;
        ofn.lpstrCustomFilter = NULL ;
        ofn.nMaxCustFilter    = 0 ;
        ofn.nFilterIndex      = 0 ;
        ofn.lpstrFile         = NULL ;  // Set in Open and Close funct
        ofn.nMaxFile                = MAX_PATH ;
        ofn.lpstrFileTitle    = NULL ;   // Set in Open and Close fun
        ofn.nMaxFileTitle           = MAX_PATH ;
        ofn.lpstrInitialDir         = NULL ;
        ofn.lpstrTitle              = NULL ;
        ofn.Flags                   = 0 ;       // Set in Open and Close f
        ofn.nFileOffset             = 0 ;
        ofn.nFileExtension          = 0 ;
        ofn.lpstrDefExt             = TEXT ("bmp") ;
```

```
        ofn.lCustData              = 0 ;
        ofn.lpfnHook               = NULL ;
   ofn.lpTemplateName    = NULL ;
}
BOOL DibFileOpenDlg (HWND hwnd, PTSTR pstrFileName, PTSTR
{
        ofn.hwndOwner       = hwnd ;
        ofn.lpstrFile              = pstrFileName ;
        ofn.lpstrFileTitle          = pstrTitleName ;
        ofn.Flags                  = 0 ;

        return GetOpenFileName (&ofn) ;
}

BOOL DibFileSaveDlg (HWND hwnd, PTSTR pstrFileName, PTSTR
{
        ofn.hwndOwner       = hwnd ;
        ofn.lpstrFile              = pstrFileName ;
        ofn.lpstrFileTitle          = pstrTitleName ;
```

```
        ofn.Flags                   = OFN_OVERWRITEPROMPT ;

        return GetSaveFileName (&ofn) ;
}

BITMAPFILEHEADER * DibLoadImage (PTSTR pstrFileName)
{
        BOOL                    bSuccess ;
        DWORD                     dwFileSize, dwHighSize, dwBytesR
        HANDLE                    hFile ;
        BITMAPFILEHEADER *    pbmfh ;

        hFile = CreateFile (  pstrFileName, GENERIC_READ, FILE_S
    OPEN_EXISTING, FILE_FLAG_SEQUENTIAL_SCAN,NULL) ;

        if    (     hFile == INVALID_HANDLE_VALUE)
                    return NULL ;

        dwFileSize = GetFileSize (hFile, &dwHighSize) ;

        if (dwHighSize)
```

```
        {
                    CloseHandle (hFile) ;
                    return NULL ;
        }

        pbmfh = malloc (dwFileSize) ;
        if          (!pbmfh)
        {
                    CloseHandle (hFile) ;
                    return NULL ;
  }

        bSuccess = ReadFile (hFile, pbmfh, dwFileSize, &dwBytes
        CloseHandle (hFile) ;

        if (!bSuccess || (dwBytesRead != dwFileSize)
                    || (pbmfh->bfType != * (WORD *) "BM")
                    || (pbmfh->bfSize != dwFileSize))
{
                    free (pbmfh) ;
```

```
                    return NULL ;
          }
   return pbmfh ;
}

BOOL DibSaveImage (PTSTR pstrFileName, BITMAPFILEHEADER *
{
   BOOL         bSuccess ;
   DWORD  dwBytesWritten ;
   HANDLE hFile ;

   hFile = CreateFile (  pstrFileName, GENERIC_WRITE, 0, NULL,
  CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL) ;

   if (hFile == INVALID_HANDLE_VALUE)
                    return FALSE ;
          bSuccess = WriteFile (hFile, pbmfh, pbmfh->bfSize, &dwB
          CloseHandle (hFile) ;

          if (!bSuccess || (dwBytesWritten != pbmfh->bfSize))
```

```
        {
            DeleteFile (pstrFileName) ;
            return FALSE ;
    }
        return TRUE ;
}
```

SHOWDIB1.RC

```
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
//////////////////////////////////////////////////////////////////////////////
// Menu
SHOWDIB1 MENU DISCARDABLE
BEGIN
    POPUP "&File"
    BEGIN
        MENUITEM "&Open...", IDM_FILE_OPEN
        MENUITEM "&Save...", IDM_FILE_SAVE
```

```
  END

END
```

```
RESOURCE.H

// Microsoft Developer Studio generated include file.

// Used by ShowDib1.rc

#define IDM_FILE_OPEN        40001

#define IDM_FILE_SAVE       40002
```

DIBFILE.CFile OpenFile SaveDIBBITMAPFILEHEADER

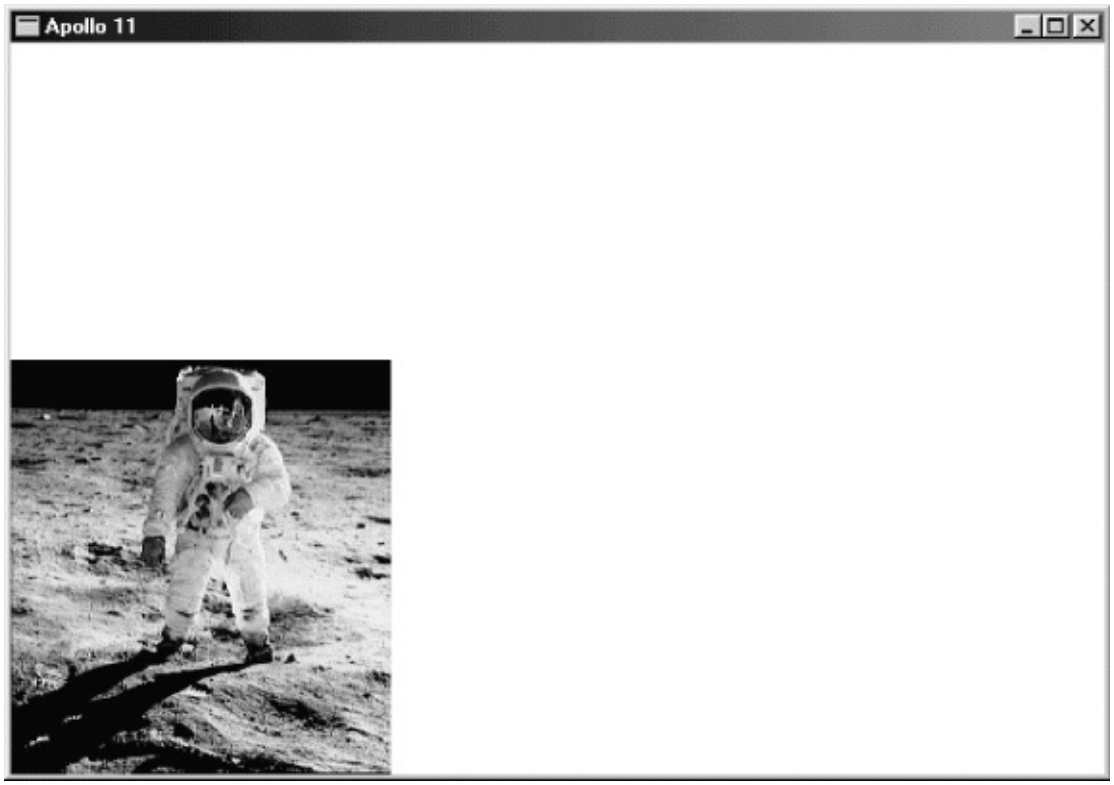
SHOWDIB1.CFile OpenDIBBITMAPINFOHEADERDIB WM_PAINTSetDIBitsToDeviceDIB

SHOWDIB1DIB

DIB

OS/2 Presentation ManagerDIBPMPM0,0

OS/2(0,0)15-1

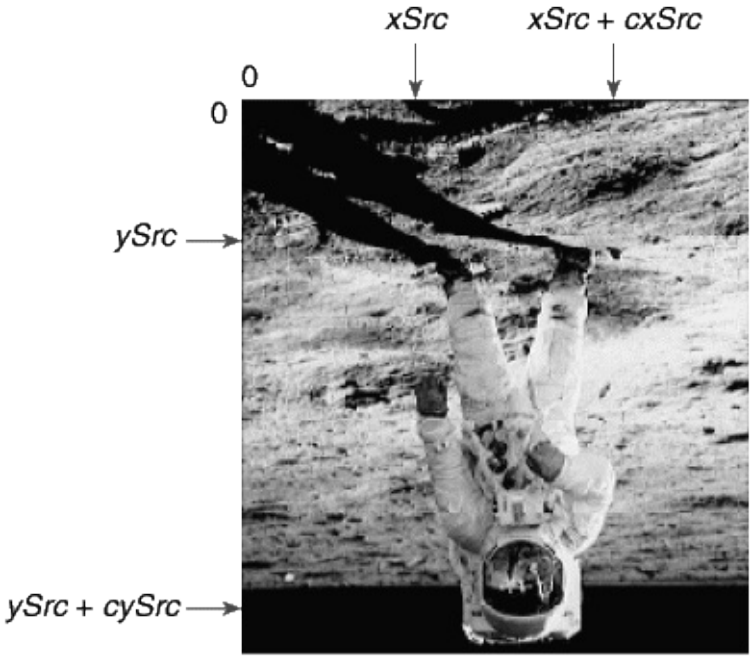


15-1　OS/200

OS/2　　(0,0)

WindowsDIBxSrcySrccxSrccySrcDIBOS/2OS/2
WindowsDIBxDst,yDst0,cyDib
xDst,yDstxSrc,　　ySrc + cySrc - 1DIB

15-2DIBDIBSetDIBitsToDevicexSrcDIBcxSrcxSrc
ySrcDIBcySrcySrc



15-2 DIB

MM_TEXT15-3

15-3

(xSrc, ySrc)	(xDst, yDst + cySrc - 1)
(xSrc + cxSrc - 1, ySrc)	(xDst + cxSrc - 1, yDst + cySrc - 1)
(xSrc, ySrc + cySrc - 1)	(xDst, yDst)

(xSrc + cxSrc - 1, ySrc + cySrc - 1)	(xDst + cxSrc - 1, yDst)

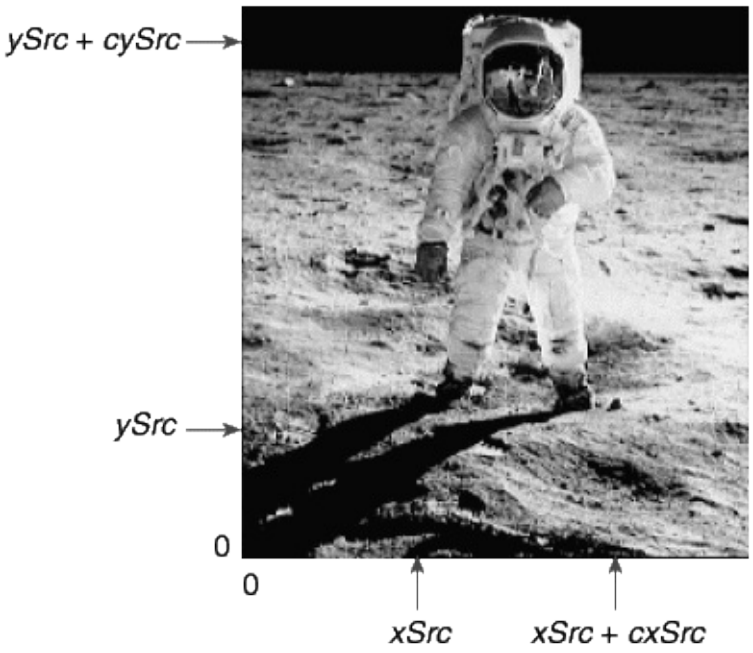
(xSrc,ySrc)(xDst,yDst)(xSrc,ySrc
MM_TEXT

BITMAPINFOHEADERbiHeightbiHeightDIB

DIBbiHeightDIBDIBDIB

DIBDIB(0,0)DIBpBitsDIB

15-3DIB



15-3
DIB

SetDIBitsToDeviceDIBDIBDIBSetDIBitsToDevice

15-3 APOLLO11

APOLLO11.C

```
/*------------------------------------------------------------------------
  APOLLO11.C -- Program for screen captures
                        (c) Charles Petzold, 1998
------------------------------------------------------------------------*/
#include <windows.h>
#include "dibfile.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
TCHAR szAppName[] = TEXT ("Apollo11") ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                    PSTR szCmdLine, int iCmdShow)
{
          HWND                hwnd ;
          MSG                      msg ;
          WNDCLAS       wndclass ;
```

```
        wndclass.style                  = CS_HREDRAW | CS_VREDR
        wndclass.lpfnWndProc              = WndProc ;
        wndclass.cbClsExtra               = 0 ;
        wndclass.cbWndExtra               = 0 ;
        wndclass.hInstance                = hInstance ;
       wndclass.hIcon                  = LoadIcon (NULL, IDI_APPLI
        wndclass.hCursor                = LoadCursor (NULL, IDC_A
        wndclass.hbrBackground            = (HBRUSH) GetStockO
        wndclass.lpszMenuName             = NULL ;
       wndclass.lpszClassName            = szAppName ;

if (!RegisterClass (&wndclass))
 {
                MessageBox (NULL, TEXT ("This program require
                        szAppName, MB_ICONERROR) ;
                return 0 ;
 }
 hwnd = CreateWindow (szAppName, TEXT ("Apollo 11"),
                WS_OVERLAPPEDWINDOW,
```

```
                    CW_USEDEFAULT, CW_USEDEFAULT,
                    CW_USEDEFAULT, CW_USEDEFAULT,
                    NULL, NULL, hInstance, NULL) ;

          ShowWindow (hwnd, iCmdShow) ;
          UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
     }
          return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
          static BITMAPFILEHEADER       * pbmfh [2] ;
          static BITMAPINFO       * pbmi  [2] ;
          static BYTE             * pBits [2] ;
```

```
static int              cxClient, cyClient, cxDib[2], cyDib[2] ;
HDC                     hdc ;
PAINTSTRUCT             ps ;

switch (message)
{
case   WM_CREATE:
            pbmfh[0] = DibLoadImage (TEXT ("Apollo11.bm
            pbmfh[1] = DibLoadImage (TEXT ("ApolloTD.bm

            if (pbmfh[0] == NULL || pbmfh[1] == NULL)
            {
MessageBox (hwnd, TEXT ("Cannot load DIB file"),
      szAppName, 0) ;
                        return 0 ;
            }
                        // Get pointers to the info structure & t

            pbmi  [0] = (BITMAPINFO *) (pbmfh[0] + 1) ;
            pbmi  [1] = (BITMAPINFO *) (pbmfh[1] + 1) ;
```

```
                pBits [0] = (BYTE *) pbmfh[0] + pbmfh[0]->bfO
                pBits [1] = (BYTE *) pbmfh[1] + pbmfh[1]->bfO

                // Get the DIB width and height (assume BITMAP
                // Note that cyDib is the absolute value of the he

                cxDib [0] =  pbmi[0]->bmiHeader.biWidth ;
                cxDib [1] =  pbmi[1]->bmiHeader.biWidth ;

                cyDib [0] = abs (pbmi[0]->bmiHeader.biHeight)
                cyDib [1] = abs (pbmi[1]->bmiHeader.biHeight)
                return 0 ;

        case   WM_SIZE:
                cxClient = LOWORD (lParam) ;
                cyClient = HIWORD (lParam) ;
                return 0 ;

        case   WM_PAINT:
                hdc = BeginPaint (hwnd, &ps) ;
```

```
            // Bottom-up DIB full size

            SetDIBitsToDevice (hdc,
        0,                              // xDst
         cyClient / 4,                    // yDst
        cxDib[0],                       // cxSrc
         cyDib[0],                      // cySrc
         0,                            // xSrc
         0,                           // ySrc
         0,                          // first scan line
         cyDib[0],                    // number of scan lines
         pBits[0],
        pbmi[0],
         DIB_RGB_COLORS) ;

                    // Bottom-up DIB partial

       SetDIBitsToDevice (hdc,
    240,                              // xDst
    cyClient / 4,                       // yDst
```

```
            80,                                 // cxSrc
             166,                                // cySrc
             80,                                 // xSrc
            60,                                 // ySrc
             0,                                  // first scan line
            cyDib[0],                               // number of scan lines
             pBits[0],
             pbmi[0],
             DIB_RGB_COLORS) ;

                        // Top-down DIB full size

               SetDIBitsToDevice     (hdc,
                    340,                // xDst
                    cyClient / 4,     // yDst
                    cxDib[0],         // cxSrc
                    cyDib[0],         // cySrc
                    0,                  // xSrc
                    0,                 // ySrc
                    0,                  // first scan line
```

```
    cyDib[0],       // number of scan lines
    pBits[0],
    pbmi[0],
    DIB_RGB_COLORS) ;

            // Top-down DIB partial

      SetDIBitsToDevice           (hdc,
580,                // xDst
cyClient / 4,       // yDst
80,                 // cxSrc
166,                // cySrc
80,                 // xSrc
60,                 // ySrc
0,                  // first scan line
cyDib[1],           // number of scan lines
pBits[1],
pbmi[1],
DIB_RGB_COLORS) ;
```

```
                EndPaint (hwnd, &ps) ;
                return 0 ;

        case   WM_DESTROY:
                if (pbmfh[0])
                        free (pbmfh[0]) ;
                if (pbmfh[1])
                        free (pbmfh[1]) ;

                PostQuitMessage (0) ;
                return 0 ;
        }
  return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

APOLLO11.BMPAPOLLOTD.BMPDIB220240DIB
absbiHeightDIBxSrcySrccxSrccySrc15-4

15-4 APOLLO11

pBitspBits

APIWindows

WindowsSetDIBitsToDeviceDIBDIBDIB
DIB

DIBDIBDIBDIB0,0DIB

DIBDIB

DIBDIBDIBWindowsDIB

DIBDIBDIB

SetDIBitsToDeviceyScancyScansSetDIBitsToDevicepBits
yScanspBitscyScanspBitsDIBBITMAPINFOHEADER

DIB235DIBpInfoDIBBITMAPINFODIB5pBits5
yScan0cyScans55yScan5yScan10153pBitsyScan
20cyScans3SetDIBitsToDevice

SetDIBitsToDeviceSetDIBitsToDevice
StretchDIBitsDIBStretchDIBitsBITMAPINFOHEADER

15-4 SEQDISP

15-4 SEQDISP

SEQDISP.C

```
/*------------------------------------------------------------------------

  SEQDISP.C --  Sequential Display of DIBs

                     (c) Charles Petzold, 1998

------------------------------------------------------------------------*/

#include <windows.h>

#include "resource.h"


LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

TCHAR szAppName[] = TEXT ("SeqDisp") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns

                PSTR szCmdLine, int iCmdShow)
```

```
{
        HACCEL             hAccel ;
        HWND               hwnd ;
        MSG                msg ;
        WNDCLASS           wndclass ;

        wndclass.style             = CS_HREDRAW | CS_VREDRAW
        wndclass.lpfnWndProc       = WndProc ;
        wndclass.cbClsExtra        = 0 ;
        wndclass.cbWndExtra   = 0 ;
        wndclass.hInstance         = hInstance ;
        wndclass.hIcon             = LoadIcon (NULL, IDI_APPLICATI
        wndclass.hCursor           = LoadCursor (NULL, IDC_ARRO
        wndclass.hbrBackground     = (HBRUSH) GetStockObjec
        wndclass.lpszMenuName      = szAppName ;
        wndclass.lpszClassName     = szAppName ;

        if (!RegisterClass (&wndclass))
        {
                MessageBox (NULL, TEXT ("This program require
```

```
                                szAppName, MB_ICONERROR) ;
                return 0 ;
}

     hwnd = CreateWindow (szAppName, TEXT ("DIB Sequent
               WS_OVERLAPPEDWINDOW,
               CW_USEDEFAULT, CW_USEDEFAULT,
              CW_USEDEFAULT, CW_USEDEFAULT,
               NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     hAccel = LoadAccelerators (hInstance, szAppName) ;
     while (GetMessage (&msg, NULL, 0, 0))
     {
               if (!TranslateAccelerator (hwnd, hAccel, &msg))
               {
                                TranslateMessage (&msg) ;
```

```
                                DispatchMessage (&msg) ;
                        }
                }
                return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
        static BITMAPINFO     *      pbmi ;
        static BYTE                *      pBits ;
        static int                      cxDib, cyDib, cBits ;
        static OPENFILENAME ofn ;
        static TCHAR                    szFileName [MAX_PATH], szT
        static TCHAR                    szFilter[]= TEXT ("Bitmap File
     TEXT ("All Files (*.*)\0*.*\0\0") ;
        BITMAPFILEHEADER          bmfh ;
        BOOL                           bSuccess, bTopDown ;
        DWORD                          dwBytesRead ;
        HANDLE                         hFile ;
```

```
HDC                         hdc ;
HMENU                       hMenu ;
int                         iInfoSize, iBitsSize, iRowLength, y
PAINTSTRUCT                 ps ;

switch (message)
{
case   WM_CREATE:
            ofn.lStructSize             = sizeof (OPENFILENAME
            ofn.hwndOwner               = hwnd ;
            ofn.hInstance               = NULL ;
            ofn.lpstrFilter             = szFilter ;
            ofn.lpstrCustomFilter       = NULL ;
            ofn.nMaxCustFilter          = 0 ;
            ofn.nFilterIndex            = 0 ;
            ofn.lpstrFile               = szFileName ;
            ofn.nMaxFile                = MAX_PATH ;
            ofn.lpstrFileTitle          = szTitleName ;
```

```
                ofn.nMaxFileTitle          = MAX_PATH ;
                ofn.lpstrInitialDir       = NULL ;
                ofn.lpstrTitle           = NULL ;
                ofn.Flags                    = 0 ;
                ofn.nFileOffset          = 0 ;
                ofn.nFileExtension        = 0 ;
                ofn.lpstrDefExt          = TEXT ("bmp") ;
                ofn.lCustData            = 0 ;
                ofn.lpfnHook             = NULL ;
                ofn.lpTemplateName      = NULL ;
               return 0 ;

        case   WM_COMMAND:
                hMenu = GetMenu (hwnd) ;

                switch (LOWORD (wParam))
                {
                case   IDM_FILE_OPEN:
                                        // Display File Open dialog
                                if (!GetOpenFileName (&ofn))
```

```
                return 0 ;

                // Get rid of old DIB

            if (pbmi)
            {
                free (pbmi) ;
                pbmi = NULL ;
            }

            if (pBits)
            {
                free (pBits) ;
                pBits = NULL ;
            }

        // Generate WM_PAINT message to erase backgr

            InvalidateRect (hwnd, NULL, TRUE
            UpdateWindow (hwnd) ;
```

```
                                        // Open the file

                                        hFile = CreateFile (szFileName, G
                    FILE_SHARE_READ, NULL, OPEN_EXISTING,
                   FILE_FLAG_SEQUENTIAL_SCAN, NULL) ;

                                        if (hFile == INVALID_HANDLE_VAL
                                        {
        MessageBox (    hwnd, TEXT ("Cannot open file."),
                                szAppName, MB_ICONWARNING | MB_
                     return 0 ;
                                        }

                                        // Read in the BITMAPFILEHEADER

                    bSuccess = ReadFile (hFile,&bmfh, sizeof (BITM
                       &dwBytesRead, NULL) ;

                   if (!bSuccess || dwBytesRead != sizeof (BITMAPF
                   {
        MessageBox (hwnd, TEXT ("Cannot read file."),
```

```
                szAppName, MB_ICONWARNING | MB_OK) ;
                 CloseHandle (hFile) ;
                return 0 ;
        }

                        // Check that it's a bitmap

            if (bmfh.bfType != * (WORD *) "BM")
            {
          MessageBox (hwnd, TEXT ("File is not a bitmap."),
                  szAppName, MB_ICONWARNING | MB_OK) ;
                  CloseHandle (hFile) ;
                  return 0 ;
            }

                  // Allocate memory for header and bits

    iInfoSize = bmfh.bfOffBits - sizeof (BITMAPFILEHEADER) ;
     iBitsSize = bmfh.bfSize - bmfh.bfOffBits ;

            pbmi  = malloc (iInfoSize) ;
```

```
                pBits = malloc (iBitsSize) ;

                if (pbmi == NULL || pBits == NULL)
                        {
          MessageBox (hwnd, TEXT ("Cannot allocate memory
                   szAppName, MB_ICONWARNING | MB_OK) ;
                    if (pbmi)
                    free (pbmi) ;
                    if (pBits)
                   free (pBits) ;
                    CloseHandle (hFile) ;
       return 0 ;
                        }

         // Read in the Information Header

bSuccess = ReadFile (hFile, pbmi, iInfoSize,   &dwBytesRe

                   if (!bSuccess || (int) dwBytesRead != iInfoSi
                   {
                MessageBox (hwnd, TEXT ("Cannot read file."),
```

```
            szAppName, MB_ICONWARNING | MB_OK) ;
               if (pbmi)
              free (pbmi) ;
               if (pBits)
              free (pBits) ;
               CloseHandle (hFile) ;
 return 0 ;
               }

// Get the DIB width and height

               bTopDown = FALSE ;

 if (pbmi->bmiHeader.biSize == sizeof (BITMAPCORE
               {
cxDib = ((BITMAPCOREHEADER *) pbmi)->bcWidth ;
cyDib = ((BITMAPCOREHEADER *) pbmi)->bcHeight
cBits = ((BITMAPCOREHEADER *) pbmi)->bcBitCoun
                        }
               else
```

```
                                {
                    if (pbmi->bmiHeader.biHeight < 0)
                                    bTopDown = TRUE ;

    cxDib =         pbmi->bmiHeader.biWidth ;
    cyDib = abs    (pbmi->bmiHeader.biHeight) ;
    cBits =         pbmi->bmiHeader.biBitCount ;

    if (pbmi->bmiHeader.biCompression != BI_RGB &&
        pbmi->bmiHeader.biCompression != BI_BITFIELDS
    {
    MessageBox (hwnd, TEXT ("File is compressed."),
                szAppName, MB_ICONWARNING | MB_OK) ;
        if (pbmi)
         free (pbmi) ;
         if (pBits)
        free (pBits) ;
         CloseHandle (hFile) ;
     return 0 ;
```

```
                }
                                        }


                                        // Get the row length

                                iRowLength = ((cxDib * cBits + 31) & ~31)

                                        // Read and display
                                        SetCursor (LoadCursor (NULL, IDC
                                        ShowCursor (TRUE) ;

                                        hdc = GetDC (hwnd) ;

                                        for (y = 0 ; y < cyDib ; y++)
                                        {
        ReadFile (hFile, pBits + y * iRowLength,iRowLength,&dwB
SetDIBitsToDevice (hdc,
                0,                      // xDst
                0,                       // yDst
                cxDib,                  // cxSrc
                cyDib,                  // cySrc
```

```
                0,                          // xSrc
                0,                          // ySrc
                bTopDown ? cyDib - y - 1 : y,
                                            // first scan line
                1,                          // number of scan lines
                pBits + y * iRowLength,
                pbmi,
                DIB_RGB_COLORS) ;
                }
                ReleaseDC (hwnd, hdc) ;
                CloseHandle (hFile) ;
                ShowCursor (FALSE) ;
                SetCursor (LoadCursor (NULL, IDC_ARROW)) ;
                return 0 ;
                }
                break ;

        case   WM_PAINT:
                hdc = BeginPaint (hwnd, &ps) ;
```

```
            if (pbmi && pBits)
                        SetDIBitsToDevice (hdc,
                0,                  // xDst
                0,                  // yDst
                cxDib,                  // cxSrc
                cyDib,                  // cySrc
                0,                  // xSrc
                0,                  // ySrc
                0,                  // first scan line
                cyDib,                  // number of scan lines
                pBits,
        DIB_RGB_COLORS) ;
            EndPaint (hwnd, &ps) ;
            return 0 ;

    case   WM_DESTROY:
            if (pbmi)
                        free (pbmi) ;
```

```
                    if (pBits)
                                        free (pBits) ;

                    PostQuitMessage (0) ;
                    return 0 ;
          }
          return DefWindowProc (hwnd, message, wParam, lParam)
}
```

SEQDISP.RC

```
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
//////////////////////////////////////////////////////////////////////////////
// Accelerator
SEQDISP ACCELERATORS DISCARDABLE
BEGIN
    "O",     IDM_FILE_OPEN,       VIRTKEY, CONTROL, NOINVERT
```

```
END

//////////////////////////////////////////////////////////////////////////////

// Menu

SEQDISP MENU DISCARDABLE

BEGIN

  POPUP "&File"

  BEGIN

    MENUITEM "&Open...\tCtrl+O",  IDM_FILE_OPEN

  END

END
```

```
RESOURCE.H

// Microsoft Developer Studio generated include file.

// Used by SeqDisp.rc

#define IDM_FILE_OPEN            40001
```

File OpenSEQDISP.CI/OWM_COMMANDSetDIBitsToDevice DIBWM_PAINT

SetDIBitsToDeviceDIBDIB

DIBStretchDIBits

```
iLines = StretchDIBits (
                        hdc,          // device context handle
                        xDst,         // x destination coordinate
                        yDst,         // y destination coordinate
                        cxDst,        // destination rectangle width
                        cyDst,        // destination rectangle height
                       xSrc,         // x source coordinate
                        ySrc,         // y source coordinate
                        cxSrc,        // source rectangle width
                        cySrc,        // source rectangle height
                        pBits,        // pointer to DIB pixel bits
                        pInfo,        // pointer to DIB information
                        fClrUse,      // color use flag
                        dwRop) ;      // raster operation
```

SetDIBitsToDevice

- (cxDst)(cyDst)
- DIB

- DIBSRCCOPY

SetDIBitsToDevicecxSrccySrcDWORD32StretchDIBitscxSrc cySrccxDstcyDstxSrcySrcint

DIB15-4

15-4

(xSrc, ySrc)	(xDst, yDst + cyDst - 1)
(xSrc + cxSrc - 1, ySrc)	(xDst + cxDst - 1, yDst + cyDst - 1)
(xSrc, ySrc + cySrc - 1)	(xDst, yDst)
(xSrc + cxSrc - 1, ySrc + cySrc - 1)	(xDst + cxDst - 1, yDst)

-1

2×2DIBStretchDIBitsxSrcySrc0cxSrccySrc2MM_TEXTxDstyDst 0cxDstcyDst42DIBx,y

```
(0,0) --> (0,2) and (1,2) and (0,3) and (1,3)

(1,0) --> (2,2) and (3,2) and (2,3) and (3,3)

(0,1) --> (0,0) and (1,0) and (0,1) and (1,1)
```

```
(1,1) --> (2,0) and (3,0) and (2,1) and (3,1)
```

(0,3)(3,3)(0,0)(3,0)

SetDIBitsToDevicexDstyDstStretchDIBitsyDIB

cyDstDIBMM_TEXTcySrccyDstDIBcxSrccxDstDIB

xMMyMMxxMM1x-1yyMM1y-1Sign
TUREFALSE

```
if (!Sign (xMM × cxSrc × cxDst))

    DIB is flipped on its vertical axis (mirror image)

if (!Sign (yMM × cySrc × cyDst))

        DIB is flipped on its horizontal axis (upside down)
```

15-4

15-5 SHOWDIBDIBDIBDIB

15-5 SHOWDIB

```
SHOWDIB2.C

/*-----------------------------------------------------------------------

  SHOWDIB2.C -- Shows a DIB in the client area

                    (c) Charles Petzold, 1998
```

```
------------------------------------------------------------------------*/

#include <windows.h>
#include "dibfile.h"
#include "resource.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
TCHAR szAppName[] = TEXT ("ShowDib2") ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                    PSTR szCmdLine, int iCmdShow)
{
          HACCEL             hAccel ;
          HWND               hwnd ;
          MSG                     msg ;
          WNDCLASS           wndclass ;
          wndclass.style                  = CS_HREDRAW | CS_VREDR
  wndclass.lpfnWndProc         = WndProc ;
          wndclass.cbClsExtra          = 0 ;
        wndclass.cbWndExtra          = 0 ;
          wndclass.hInstance           = hInstance ;
```

```
        wndclass.hIcon              = LoadIcon (NULL, IDI_APPLICAT
        wndclass.hCursor            = LoadCursor (NULL, IDC_ARRO
        wndclass.hbrBackground      = (HBRUSH) GetStockObjec
        wndclass.lpszMenuName       = szAppName ;
        wndclass.lpszClassName      = szAppName ;

        if (!RegisterClass (&wndclass))
        {
                MessageBox (NULL, TEXT ("This program require
                        szAppName, MB_ICONERROR) ;
            return 0 ;
        }

hwnd = CreateWindow (szAppName, TEXT ("Show DIB #2"),
                WS_OVERLAPPEDWINDOW,
                CW_USEDEFAULT, CW_USEDEFAULT,
                CW_USEDEFAULT, CW_USEDEFAULT,
                NULL, NULL, hInstance, NULL) ;

        ShowWindow (hwnd, iCmdShow) ;
```

```
        UpdateWindow (hwnd) ;

        hAccel = LoadAccelerators (hInstance, szAppName) ;
        while (GetMessage (&msg, NULL, 0, 0))
   {
                if (!TranslateAccelerator (hwnd, hAccel, &msg))
                 {
                                TranslateMessage (&msg) ;
                                DispatchMessage (&msg) ;
                 }
   }
   return msg.wParam ;
}

int ShowDib (HDC hdc, BITMAPINFO * pbmi, BYTE * pBits, int cxD
                            int cxClient, int cyClient, WORD wShow
{
        switch (wShow)
        {
        case   IDM_SHOW_NORMAL:
```

```
                return SetDIBitsToDevice (hdc, 0, 0, cxDib, cyDi
                0, cyDib, pBits, pbmi, DIB_RGB_COLORS) ;

        case   IDM_SHOW_CENTER:
                return SetDIBitsToDevice (hdc, (cxClient - cxDib)
                (cyClient - cyDib) / 2,
              cxDib, cyDib, 0, 0, 0, cyDib, pBits, pbmi, DIB_RGB_C

        case   IDM_SHOW_STRETCH:
                SetStretchBltMode (hdc, COLORONCOLOR) ;
return StretchDIBits(hdc,0, 0, cxClient, cyClient, 0, 0, cxDib, cyD
                  pBits, pbmi, DIB_RGB_COLORS, SRCCOPY) ;

        case   IDM_SHOW_ISOSTRETCH:
                SetStretchBltMode (hdc, COLORONCOLOR) ;
                SetMapMode (hdc, MM_ISOTROPIC) ;
                SetWindowExtEx (hdc, cxDib, cyDib, NULL) ;
                SetViewportExtEx (hdc, cxClient, cyClient, NULL
                SetWindowOrgEx (hdc, cxDib / 2, cyDib / 2, NUL
```

```
                    SetViewportOrgEx (hdc, cxClient / 2, cyClient / 2

return StretchDIBits(hdc,0, 0, cxDib, cyDib, 0, 0, cxDib, cyDib,
                            pBits, pbmi, DIB_RGB_COLORS, SRCCOPY
          }
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
          static BITMAPFILEHEADER       *      pbmfh ;
          static BITMAPINFO             *      pbmi ;
          static BYTE                   *      pBits ;
          static DOCINFO                di = {sizeof (DOCINFO),
                              TEXT ("ShowDib2: Printing") } ;
          static int                 cxClient, cyClient, cxDib, cyDib ;
          static PRINTDLG         printdlg = { sizeof (PRINTDLG) } ;
          static TCHAR            szFileName [MAX_PATH], szTitleName
          static WORD               wShow = IDM_SHOW_NORMAL ;
          BOOL                     bSuccess ;
          HDC                      hdc, hdcPrn ;
```

```
HGLOBAL                hGlobal ;
HMENU                  hMenu ;
int                    cxPage, cyPage, iEnable ;
PAINTSTRUCT            ps ;
BYTE                   * pGlobal ;

switch (message)
{
case   WM_CREATE:
           DibFileInitialize (hwnd) ;
           return 0 ;

case   WM_SIZE:
           cxClient = LOWORD (lParam) ;
           cyClient = HIWORD (lParam) ;
           return 0 ;

case   WM_INITMENUPOPUP:
           hMenu = GetMenu (hwnd) ;
```

```
                if (pbmfh)
                                iEnable = MF_ENABLED ;
                else
                                 iEnable = MF_GRAYED ;

                EnableMenuItem (hMenu, IDM_FILE_SAVE,       i
                EnableMenuItem (hMenu, IDM_FILE_PRINT,      i
                EnableMenuItem (hMenu, IDM_EDIT_CUT,        i
                EnableMenuItem (hMenu, IDM_EDIT_COPY,
                EnableMenuItem (hMenu, IDM_EDIT_DELETE, iEn
                return 0 ;

        case   WM_COMMAND:
                hMenu = GetMenu (hwnd) ;

               switch (LOWORD (wParam))
                {
                case   IDM_FILE_OPEN:
                            // Show the File Open dialog box

                      if (!DibFileOpenDlg (hwnd, szFileName, szT
```

```
                    return 0 ;

                // If there's an existing DIB, free the me
                    if (pbmfh)
                    {
                        free (pbmfh) ;
                        pbmfh = NULL ;
                    }
                // Load the entire DIB into memory
    SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
    ShowCursor (TRUE) ;
                    pbmfh = DibLoadImage (szFileName) ;
                    ShowCursor (FALSE) ;
                    SetCursor (LoadCursor (NULL, IDC_ARROW
            // Invalidate the client area for later update
                    InvalidateRect (hwnd, NULL, TRUE) ;
```

```
            if (pbmfh == NULL)
            {
MessageBox (   hwnd, TEXT ("Cannot load DIB file"),
            szAppName, MB_ICONEXCLAMATION | MB_
                  return 0 ;
            }
// Get pointers to the info structure & the bits

            pbmi   = (BITMAPINFO *) (pbmfh + 1) ;
            pBits = (BYTE *) pbmfh + pbmfh->bfOffBit

// Get the DIB width and height

      if (pbmi->bmiHeader.biSize == sizeof (BITMAPC
            {
     cxDib = ((BITMAPCOREHEADER *) pbmi)->bcWid
      cyDib = ((BITMAPCOREHEADER *) pbmi)->bcHe
            }
                  else
```

```
                    {
                cxDib =       pbmi->bmiHeader.biWidth ;
               cyDib = abs (pbmi->bmiHeader.biHeight) ;
                    }
                         return 0 ;

          case   IDM_FILE_SAVE:
                              // Show the File Save dialog box

                        if (!DibFileSaveDlg (hwnd, szFileName, szTi
                   return 0 ;
                         // Save the DIB to a disk file

                         SetCursor (LoadCursor (NULL, IDC_WA
                         ShowCursor (TRUE) ;

                         bSuccess = DibSaveImage (szFileNam

                         ShowCursor (FALSE) ;
                         SetCursor (LoadCursor (NULL, IDC_ARR

                        if (!bSuccess)
```

```
            MessageBox (    hwnd, TEXT ("Cannot save DIB file"),
                                szAppName, MB_ICONEXCLAMATI
                            return 0 ;

            case   IDM_FILE_PRINT:
                            if (!pbmfh)
                                  return 0 ;

                                  // Get printer DC

        printdlg.Flags = PD_RETURNDC | PD_NOPAGENUMS | PD_

                            if (!PrintDlg (&printdlg))
                                        return 0 ;

                            if     (NULL == (hdcPrn = printdlg.hDC
                            {
         MessageBox (hwnd, TEXT ("Cannot obtain Printer DC"),
                            szAppName, MB_ICONEXCLAMATION |
                                        return 0 ;
                            }
```

```
        // Check whether the printer can print bitmaps

    if (!(RC_BITBLT & GetDeviceCaps (hdcPrn, RASTERCAPS)
                        {
                        DeleteDC (hdcPrn) ;
    MessageBox (  hwnd, TEXT ("Printer cannot print bitmaps
              szAppName, MB_ICONEXCLAMATION | MB_OK) ;
                              return 0 ;
                        }
                               // Get size of printable area of pa

                        cxPage = GetDeviceCaps (hdcPrn, HOI
                        cyPage = GetDeviceCaps (hdcPrn, VER

                        bSuccess = FALSE ;
                               // Send the DIB to the printer

                        SetCursor (LoadCursor (NULL, IDC_WA
                    ShowCursor (TRUE) ;
```

```
if ((StartDoc (hdcPrn, &di) > 0) && (StartPage (hdcPrn) > 0))
                        {
          ShowDib (     hdcPrn, pbmi, pBits, cxDib, cyDib,
               cxPage, cyPage, wShow) ;

                              if (EndPage (hdcPrn) > 0)
                              {
                                        bSuccess = TRUE ;
                                         EndDoc (hdcPrn) ;
                              }
                        }
                        ShowCursor (FALSE) ;
                        SetCursor (LoadCursor (NULL, IDC_ARR

                        DeleteDC (hdcPrn) ;

                      if (!bSuccess)
      MessageBox (hwnd, TEXT ("Could not print bitmap"),
                        szAppName, MB_ICONEXCLAMATION |
                        return 0 ;
```

```
        case   IDM_EDIT_COPY:
        case   IDM_EDIT_CUT:
                        if (!pbmfh)
                                return 0 ;

                                // Make a copy of the packed DIB

hGlobal = GlobalAlloc (GHND | GMEM_SHARE, pbmfh->bfS
                        sizeof (BITMAPFILEHEADER)) ;

                                pGlobal = GlobalLock (hGlobal) ;

CopyMemory (   pGlobal, (BYTE *) pbmfh + sizeof (BITMAI
                pbmfh->bfSize - sizeof (BITMAPFILEHEADER)) ;

                                GlobalUnlock (hGlobal) ;

                                      // Transfer it to the clipboard

                                OpenClipboard (hwnd) ;
                                EmptyClipboard () ;
```

```
                    SetClipboardData (CF_DIB, hGlobal) ;
                    CloseClipboard () ;

                    if (LOWORD (wParam) == IDM_EDIT_C
                        return 0 ;
                // fall through if IDM_EDIT_CUT
            case   IDM_EDIT_DELETE:
                    if (pbmfh)
                    {
                    free (pbmfh) ;
                    pbmfh = NULL ;
                    InvalidateRect (hwnd, NULL, TRUE) ;
                    }
                    return 0 ;

            case   IDM_SHOW_NORMAL:
            case   IDM_SHOW_CENTER:
            case   IDM_SHOW_STRETCH:
            case   IDM_SHOW_ISOSTRETCH:
                    CheckMenuItem (hMenu, wShow, MF_U
```

```
                    wShow = LOWORD (wParam) ;
                    CheckMenuItem (hMenu, wShow, MF_C
                    InvalidateRect (hwnd, NULL, TRUE) ;
                    return 0 ;
          }
          break ;

     case   WM_PAINT:
               hdc = BeginPaint (hwnd, &ps) ;

               if (pbmfh)
          ShowDib (     hdc, pbmi, pBits, cxDib, cyDib,
            cxClient, cyClient, wShow) ;

               EndPaint (hwnd, &ps) ;
               return 0 ;

     case   WM_DESTROY:
               if (pbmfh)
```

```
                                free (pbmfh) ;

                        PostQuitMessage (0) ;
                        return 0 ;
                }
  return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

SHOWDIB2.RC

```
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
//////////////////////////////////////////////////////////////////////////////
// Menu
SHOWDIB2 MENU DISCARDABLE
BEGIN
    POPUP "&File"
   BEGIN
     MENUITEM "&Open...\tCtrl+O",IDM_FILE_OPEN
     MENUITEM "&Save...\tCtrl+S",  IDM_FILE_SAVE
```

```
    MENUITEM SEPARATOR
    MENUITEM "&Print\tCtrl+P",     IDM_FILE_PRINT
  END
  POPUP "&Edit"
   BEGIN
    MENUITEM          "Cu&t\tCtrl+X", IDM_EDIT_CUT
    MENUITEM          "&Copy\tCtrl+C", IDM_EDIT_COPY
    MENUITEM          "&Delete\tDelete", IDM_EDIT_DELETE
   END
   POPUP "&Show"
   BEGIN
    MENUITEM          "&Actual Size",       IDM_SHOW_NORMAL, C
    MENUITEM          "&Center",            IDM_SHOW_CENTER
    MENUITEM          "&Stretch to Window", IDM_SHOW_STRETC
    MENUITEM          "Stretch &Isotropically",     IDM_SHOW_ISC
  END
END

//////////////////////////////////////////////////////////////////////
```

```
// Accelerator
SHOWDIB2 ACCELERATORS DISCARDABLE
BEGIN
      "C",   IDM_EDIT_COPY,        VIRTKEY, CONTROL, NOINVERT
      "O",   IDM_FILE_OPEN,       VIRTKEY, CONTROL, NOINVERT
   "P",   IDM_FILE_PRINT,       VIRTKEY, CONTROL, NOINVERT
   "S",   IDM_FILE_SAVE,        VIRTKEY, CONTROL, NOINVERT
   VK_DELETE,      IDM_EDIT_DELETE,      VIRTKEY, NOINVERT
   "X",   IDM_EDIT_CUT,         VIRTKEY, CONTROL, NOINVERT
END
```

RESOURCE.H

```
// Microsoft Developer Studio generated include file.
// Used by ShowDib2.rc
#define IDM_FILE_OPEN         40001
#define IDM_SHOW_NORMAL       40002
#define IDM_SHOW_CENTER       40003
#define IDM_SHOW_STRETCH      40004
#define IDM_SHOW_ISOSTRETCH   40005
```

```
#define IDM_FILE_PRINT      40006
#define IDM_EDIT_COPY       40007
#define IDM_EDIT_CUT        40008
#define IDM_EDIT_DELETE      40009
#define IDM_FILE_SAVE       40010
```

ShowDibDIBSetDIBitsToDeviceDIBStretchDIBitsDIB

DIBpacked DIBCF_DIBDIBpacked

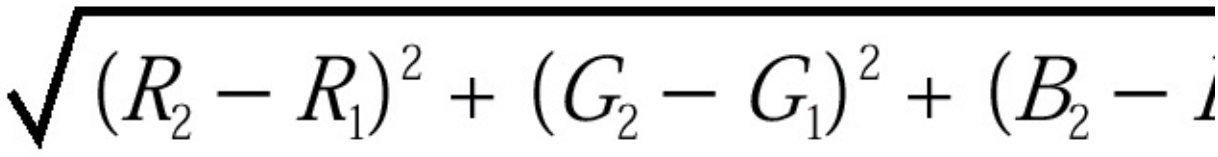
SHOWDIB2256Windows4DIB
SHOWDIB2 packed DIBDIBDIB

William GoldmanAll the President's MenDe
DIBSetDIBitsToDeviceStretchDIBits

2424DIB2416DIB2448DIBDIBDIB

48162432DIBDIBGDIGetNearestColor

RGB



R1G1B1R2G2B2RGB88DIBDIB

SetDIBitsToDeviceStretchDIBits8162432DIBDIB8DIB8DDB DIBDDBBitBltStretchBltDIB

8Windows8True-ColorDIBDIB8DIB2020

Windows 98Windows NTWindows APIDIBDDBCreateDIBSection

DIB DDB

DIBDIBSetDIBitsToDeviceStretchDIBitsDIBDIB

GDIDDBDIBSetDIBitsToDeviceStretchDIBitsBitBlt StretchBltWindows NT824DIB

DIBDDBDIBDIBDDBBitBltStretchBlt

DIBDDB

DIBGDIDIBCreateCompatibleBitmapDIBGDISetDIBitsToDevice DCDDBDIB

CreateDIBitmap

```
hBitmap = CreateDIBitmap (
            hdc,            // device context handle
                            pInfoHdr,           // pointer to DIB infor
```

```
                fInit,      // 0 or CBM_INIT
                pBits,      // pointer to DIB pixel bits
                pInfo,      // pointer to DIB information
                fClrUse) ;  // color use flag
```

pInfoHdrpInfoBITMAPINFOHEADERBITMAPINFO
BITMAPINFOBITMAPINFOHEADERDIB_RGB_
DIB_PAL_COLORSSetDIBitsToDevice

WindowsCreateDIBitmapGDICreateBitmap
CreateBitmapIndirectCreateCompatibleBitmap

CreateDIBitmap

```
hBitmap = CreateDIBitmap (NULL, pbmih, 0, NULL, NULL, 0) ;
```

BITMAPINFOHEADERGDI

```
hBitmap = CreateDIBitmap (hdc, pbmih, 0, NULL, NULL, 0) ;
```

DDBhdcCreateBitmapCreateCompatibleBitmap

CreateDIBitmapCreateDIBitmapCBM_INITWindowsDDBpInfo
BITMAPINFOpBitsBITMAPINFOSetDIBitsToDevice
CreateDIBitmap

```
HBITMAP CreateDIBitmap (   HDC hdc, CONST BITMAPINFOHEAD
                        DWORD fInit, CONST VOID * pBits,
```

```
                CONST BITMAPINFO * pbmi, UINT fUsage)
{
        HBITMAP      hBitmap ;
        HDC                hdc ;
        int                cx, cy, iBitCount ;
        if (pbmih->biSize == sizeof (BITMAPCOREHEADER))
   {
                cx   = ((PBITMAPCOREHEADER) pbmih)->bcWid
                cy   = ((PBITMAPCOREHEADER) pbmih)->bcHeig
                iBitCount     = ((PBITMAPCOREHEADER) pbmih)-
        }
        else
   {
                cx  = pbmih->biWidth ;
                cy  = pbmih->biHeight ;
               iBitCount     = pbmih->biBitCount ;
  }
        if (hdc)
```

```
                hBitmap = CreateCompatibleBitmap (hdc, cx, cy
        else
                hBitmap = CreateBitmap (cx, cy, 1, 1, NULL) ;
        if    (fInit == CBM_INIT)
        {
                hdcMem = CreateCompatibleDC (hdc) ;
                SelectObject (hdcMem, hBitmap) ;
                SetDIBitsToDevice (   hdcMem, 0, 0, cx, cy, 0, 0,
             pBits, pbmi, fUsage) ;
                DeleteDC (hdcMem) ;
   }

return hBitmap ;
}
```

DIBSetDIBitsToDeviceCreateDIBitmapBitBltStretchBltDDB
SetDIBitsToDeviceCreateDIBitmapDIBWM_PAINT

15-6　DIBCONVSetDIBitsToDeviceDIBDDB

15-6　DIBCONV

DIBCONV.C

```
/*------------------------------------------------------------------------
  DIBCONV.C --  Converts a DIB to a DDB
                        (c) Charles Petzold, 1998
------------------------------------------------------------------------*/
#include <windows.h>
#include <commdlg.h>
#include "resource.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
TCHAR szAppName[] = TEXT ("DibConv") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                PSTR szCmdLine, int iCmdShow)
{
        HWND                hwnd ;
  MSG                       msg ;
        WNDCLASS            wndclass ;

        wndclass.style                  = CS_HREDRAW | CS_VREDR
        wndclass.lpfnWndProc        = WndProc ;
```

```
     wndclass.cbClsExtra          = 0 ;
     wndclass.cbWndExtra          = 0 ;
     wndclass.hInstance           = hInstance ;
     wndclass.hIcon               = LoadIcon (NULL, IDI_APPLICAT
     wndclass.hCursor             = LoadCursor (NULL, IDC_ARRO
     wndclass.hbrBackground       = (HBRUSH) GetStockObjec
     wndclass.lpszMenuName        = szAppName ;
     wndclass.lpszClassName       = szAppName ;

     if (!RegisterClass (&wndclass))
     {
               MessageBox (NULL, TEXT ("This program require
               szAppName, MB_ICONERROR) ;
               return 0 ;
}

     hwnd = CreateWindow (szAppName, TEXT ("DIB to DDB C
               WS_OVERLAPPEDWINDOW,
               CW_USEDEFAULT, CW_USEDEFAULT,
```

```
                    CW_USEDEFAULT, CW_USEDEFAULT,
                    NULL, NULL, hInstance, NULL) ;

          ShowWindow (hwnd, iCmdShow) ;
          UpdateWindow (hwnd) ;

          while (GetMessage (&msg, NULL, 0, 0))
          {
                    TranslateMessage (&msg) ;
                    DispatchMessage (&msg) ;

          }
          return msg.wParam ;
}
HBITMAP CreateBitmapObjectFromDibFile (HDC hdc, PTSTR szFil
{
          BITMAPFILEHEADER *    pbmfh ;
          BOOL              bSuccess ;
          DWORD             dwFileSize, dwHighSize, dwBytesRead ;
          HANDLE            hFile ;
```

```
        HBITMAP         hBitmap ;

                // Open the file: read access, prohibit write acce

        hFile = CreateFile (szFileName, GENERIC_READ, FILE_SHA
    OPEN_EXISTING, FILE_FLAG_SEQUENTIAL_SCAN, NULL) ;
        if     (hFile == INVALID_HANDLE_VALUE)
                        return NULL ;

                // Read in the whole file

        dwFileSize = GetFileSize (hFile, &dwHighSize) ;

        if (dwHighSize)
        {
                CloseHandle (hFile) ;
                return NULL ;
        }

        pbmfh = malloc (dwFileSize) ;

if (!pbmfh)
```

```
        {
                CloseHandle (hFile) ;
                return NULL ;
    }

        bSuccess = ReadFile (hFile, pbmfh, dwFileSize, &dwBytes
        CloseHandle (hFile) ;

                // Verify the file
        if (!bSuccess        || (dwBytesRead != dwFileSize)
                        || (pbmfh->bfType != * (WORD *) "BM")
                        || (pbmfh->bfSize != dwFileSize))
    {
                free (pbmfh) ;
                return NULL ;
    }
                // Create the DDB
        hBitmap = CreateDIBitmap      (hdc,
                                (BITMAPINFOHEADER *) (pbmfh +
                            CBM_INIT,
```

```
                                    (BYTE *) pbmfh + pbmfh->bfOffBi
                                     (BITMAPINFO *) (pbmfh + 1),
                                    DIB_RGB_COLORS) ;
          free (pbmfh) ;
          return hBitmap ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
          static HBITMAP        hBitmap ;
          static int            cxClient, cyClient ;
          static OPENFILENAME   ofn ;
         static TCHAR  szFileName [MAX_PATH], szTitleName [MAX_
          static TCHAR  szFilter[]=TEXT("Bitmap Files(*.BMP)\0*.bm
      TEXT ("All Files (*.*)\0*.*\0\0") ;
          BITMAP                        bitmap ;
          HDC                           hdc, hdcMem ;
          PAINTSTRUCT                       ps ;
```

```
       switch (message)
     {
case   WM_CREATE:
                    ofn.lStructSize             = sizeof (OPENFILENAM
                    ofn.hwndOwner         = hwnd ;
                    ofn.hInstance               = NULL ;
                    ofn.lpstrFilter             = szFilter ;
                    ofn.lpstrCustomFilter = NULL ;
                    ofn.nMaxCustFilter    = 0 ;
                    ofn.nFilterIndex            = 0 ;
                    ofn.lpstrFile               = szFileName ;
                    ofn.nMaxFile                = MAX_PATH ;
                    ofn.lpstrFileTitle          = szTitleName ;
                    ofn.nMaxFileTitle           = MAX_PATH ;
                    ofn.lpstrInitialDir                 = NULL ;
                    ofn.lpstrTitle              = NULL ;
                    ofn.Flags                   = 0 ;
                    ofn.nFileOffset             = 0 ;
                    ofn.nFileExtension          = 0 ;
```

```
                ofn.lpstrDefExt             = TEXT ("bmp") ;
                ofn.lCustData               = 0 ;
                ofn.lpfnHook                = NULL ;
                ofn.lpTemplateName    = NULL ;

                return 0 ;

        case   WM_SIZE:
                cxClient = LOWORD (lParam) ;
                cyClient = HIWORD (lParam) ;
                return 0 ;

        case   WM_COMMAND:
                switch (LOWORD (wParam))
                {
                case   IDM_FILE_OPEN:

                     // Show the File Open dialog box

                                if (!GetOpenFileName (&ofn))
                                     return 0 ;
```

```
                // If there's an existing DIB, delete it

                    if (hBitmap)

                    {

                        DeleteObject (hBitmap) ;

                        hBitmap = NULL ;

                    }

                        // Create the DDB from the D

                    SetCursor (LoadCursor (NULL, IDC

                    ShowCursor (TRUE) ;

            hdc = GetDC (hwnd) ;

             hBitmap = CreateBitmapObjectFromDibFile (hd

             ReleaseDC (hwnd, hdc) ;

                     ShowCursor (FALSE) ;

                    SetCursor (LoadCursor (NULL, IDC

        // Invalidate the client area for later update
```

```
                    InvalidateRect (hwnd, NULL, TRUE
                    if (hBitmap == NULL)
                    {
    MessageBox (hwnd, TEXT ("Cannot load DIB file"),
          szAppName, MB_OK | MB_ICONEXCLAMATIO
                    }
                    return 0 ;
          }
          break ;

case   WM_PAINT:
          hdc = BeginPaint (hwnd, &ps) ;

           if (hBitmap)
           {
          GetObject (hBitmap, sizeof (BITMAP), &bitmap)

          hdcMem = CreateCompatibleDC (hdc) ;
          SelectObject (hdcMem, hBitmap) ;
```

```
                BitBlt (hdc,0, 0, bitmap.bmWidth, bitmap.bmHei
                     hdcMem, 0, 0, SRCCOPY) ;

                                  DeleteDC (hdcMem) ;
                     }

                EndPaint (hwnd, &ps) ;
                     return 0 ;

           case   WM_DESTROY:
                     if (hBitmap)
                          DeleteObject (hBitmap) ;

                     PostQuitMessage (0) ;
                     return 0 ;
           }
           return DefWindowProc (hwnd, message, wParam, lPa
}
```

DIBCONV.RC

```
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
/////////////////////////////////////////////////////////////////////////////
// Menu
DIBCONV MENU DISCARDABLE
BEGIN
  POPUP "&File"
  BEGIN
            MENUITEM "&Open",     IDM_FILE_OPEN
  END
END
```

```
RESOURCE.H
// Microsoft Developer Studio generated include file.
// Used by DibConv.rc
#define IDM_FILE_OPEN    40001
```

GetObject

CreateDIBitmapDDBSetDIBits

```
iLines = SetDIBits (
                    hdc,         // device context handle
                    hBitmap,    // bitmap handle
                    yScan,      // first scan line to convert
                    cyScans,   // number of scan lines to convert
                    pBits,       // pointer to pixel bits
                    pInfo,       // pointer to DIB information
                   fClrUse) ;  // color use flag
```

BITMAPINFODIB_PAL_COLORS

DDBDIB

SetDIBitsGetDIBitsDDBDIB

```
int WINAPI GetDIBits (
                hdc,                // device context handle
                hBitmap,      // bitmap handle
                yScan,        // first scan line to convert
                cyScans,      // number of scan lines to convert
```

```
            pBits,              // pointer to pixel bits (out)
            pInfo,              // pointer to DIB information (out)
            fClrUse) ;    // color use flag
```

SetDIBitsCreateDIBitmapSetDIBitsDIBDDBGetDIBitsDDBDIB
DIBWindows

GetDIBitsDIBGetDIBitsBLOWUPMicrosoft
Knowledge BaseQ80080

DIB

DIBDDBDIBDDBGDIDIBDDBDIB

32WindowsCreateDIBSection

```
hBitmap = CreateDIBSection (
             hdc,          // device context handle
             pInfo,        // pointer to DIB information
            fClrUse,     // color use flag
            ppBits,      // pointer to pointer variable
            hSection,     // file-mapping object handle
            dwOffset) ;   // offset to bits in file-mapping object
```

CreateDIBSectionWindows API

DIBDIB sectionCreateDIBSectionDIB
CreateDIBSectionDIB

GDICreateDIBSectionCreateDIBitmap
CreateDIBSection

CreateDIBSectionCreateDIBSectionCreateDIBitmap
CreateDIBitmapCreateDDBitmap

CreateDIBSectionhSectiondwOffsetNULL0fColorUse
DIB_ PAL_COLORShdcfColorUseDIB_RGB_COLORS0hdc
CreateDIBitmaphdcDDB

CreateDIBSectionBITMAPINFO

24384×256DIB24BITMAPINFOBITMAPINFOHEADER

BITMAPINFOHEADERBYTE

```
BITMAPINFOHEADER   bmih ;

BYTE                    * pBits ;

HBITMAP                 hBitmap ;
```

BITMAPINFOHEADER

```
bmih->biSize           = sizeof (BITMAPINFOHEADER) ;

bmih->biWidth           = 384 ;

bmih->biHeight          = 256 ;

bmih->biPlanes          = 1 ;

bmih->biBitCount         = 24 ;

bmih->biCompression       = BI_RGB ;

bmih->biSizeImage        = 0 ;
```

```
bmih->biXPelsPerMeter     = 0 ;
bmih->biYPelsPerMeter     = 0 ;
bmih->biClrUsed           = 0 ;
bmih->biClrImportant      = 0 ;
```

```
hBitmap = CreateDIBSection (NULL, (BITMAPINFO *)  &bmih, 0, &
```

BITMAPINFOHEADERBYIEpBits

CreateDIBSectionBITMAPINFOHEADERDIB384×256×3 pBitsCreateDIBitmap

DIBpBitsReadFile

15-7　DIBSECTCreateDIBSectionCreateDIBitmapDIBCONV

15-7　DIBSECT

```
DIBSECT.C
/*------------------------------------------------------------------------
  DIBSECT.C -- Displays a DIB Section in the client area
                    (c) Charles Petzold, 1998
------------------------------------------------------------------------*/
#include <windows.h>
```

```
#include <commdlg.h>
#include "resource.h"
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

TCHAR szAppName[] = TEXT ("DIBsect") ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                    PSTR szCmdLine, int iCmdShow)
{
     HWND         hwnd ;
     MSG          msg ;
     WNDCLASS     wndclass ;

     wndclass.style          = CS_HREDRAW | CS_VREDR
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICAT
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARRO
```

```
     wndclass.hbrBackground       = (HBRUSH) GetStockObjec
     wndclass.lpszMenuName        = szAppName ;
     wndclass.lpszClassName       = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program require
                              szAppName, MB_ICONERRO
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, TEXT ("DIB Section I
               WS_OVERLAPPEDWINDOW,
               CW_USEDEFAULT, CW_USEDEFAULT,
               CW_USEDEFAULT, CW_USEDEFAULT,
               NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
```

```
        {
                TranslateMessage (&msg) ;
                DispatchMessage (&msg) ;

        }
        return msg.wParam ;
}
HBITMAP CreateDIBsectionFromDibFile (PTSTR szFileName)
{
        BITMAPFILEHEADER                bmfh ;
        BITMAPINFO          *     pbmi ;
        BYTE                *     pBits ;
        BOOL                bSuccess ;
        DWORD               dwInfoSize, dwBytesRead ;
        HANDLE              hFile ;
        HBITMAP             hBitmap ;

             // Open the file: read access, prohibit write access

   hFile = CreateFile (szFileName, GENERIC_READ, FILE_SHARE_F
```

```
                    NULL, OPEN_EXISTING, 0, NULL) ;
     if (hFile == INVALID_HANDLE_VALUE)
                    return NULL ;
                    // Read in the BITMAPFILEHEADER
     bSuccess = ReadFile (hFile, &bmfh, sizeof (BITMAPFILEHE
                              &dwBytesRead, NULL) ;

     if (!bSuccess || (dwBytesRead != sizeof (BITMAPFILEHEAD
                    || (bmfh.bfType != * (WORD *) "BM"))
{
                    CloseHandle (hFile) ;
                    return NULL ;
}
                    // Allocate memory for the BITMAPINFO structur
     dwInfoSize = bmfh.bfOffBits - sizeof (BITMAPFILEHEADER
     pbmi = malloc (dwInfoSize) ;
     bSuccess = ReadFile (hFile, pbmi, dwInfoSize, &dwBytesR
     if (!bSuccess || (dwBytesRead != dwInfoSize))
```

```
        {
                free (pbmi) ;
                CloseHandle (hFile) ;
                return NULL ;
 }
                // Create the DIB Section
        hBitmap = CreateDIBSection (NULL, pbmi, DIB_RGB_COL
        if (hBitmap == NULL)
        {
         free (pbmi) ;
         CloseHandle (hFile) ;
         return NULL ;
}

                // Read in the bitmap bits
        ReadFile (hFile, pBits, bmfh.bfSize - bmfh.bfOffBits, &dwB
        free (pbmi) ;
        CloseHandle (hFile) ;

        return hBitmap ;
```

```
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
          static HBITMAP              hBitmap ;
          static int                  cxClient, cyClient ;
          static OPENFILENAME ofn ;
          static TCHAR        szFileName [MAX_PATH], szTitleName [M
          static TCHAR        szFilter[] =     TEXT ("Bitmap Files (*.BM
      TEXT ("All Files (*.*)\0*.*\0\0") ;
          BITMAP                    bitmap ;
          HDC                       hdc, hdcMem ;
          PAINTSTRUCT          ps ;

         switch (message)
          {
          case   WM_CREATE:
                    ofn.lStructSize             = sizeof (OPENFILENAM
                    ofn.hwndOwner        = hwnd ;
```

```
ofn.hInstance         = NULL ;
ofn.lpstrFilter       = szFilter ;
ofn.lpstrCustomFilter = NULL ;
ofn.nMaxCustFilter    = 0 ;
ofn.nFilterIndex      = 0 ;
ofn.lpstrFile         = szFileName ;
ofn.nMaxFile          = MAX_PATH ;
ofn.lpstrFileTitle    = szTitleName ;
ofn.nMaxFileTitle     = MAX_PATH ;
ofn.lpstrInitialDir   = NULL ;
ofn.lpstrTitle        = NULL ;
ofn.Flags             = 0 ;
ofn.nFileOffset       = 0 ;
ofn.nFileExtension    = 0 ;
ofn.lpstrDefExt       = TEXT ("bmp") ;
ofn.lCustData         = 0 ;
ofn.lpfnHook          = NULL ;
ofn.lpTemplateName    = NULL ;
```

```
                return 0 ;

        case   WM_SIZE:
                cxClient = LOWORD (lParam) ;
                cyClient = HIWORD (lParam) ;
                return 0 ;

case   WM_COMMAND:
                switch (LOWORD (wParam))
                {
                case   IDM_FILE_OPEN:

                // Show the File Open dialog box

                                if     (!GetOpenFileName (&ofn))
                                        return 0 ;

                // If there's an existing bitmap, delete it

                                if     (hBitmap)
                                {
```

```
                    DeleteObject (hBitmap) ;
                    hBitmap = NULL ;
                }
            // Create the DIB Section from the DIB file
                SetCursor (LoadCursor (NULL, IDC_
                ShowCursor (TRUE) ;
            hBitmap = CreateDIBsectionFromDibFile (s
                ShowCursor (FALSE) ;
                SetCursor (LoadCursor (NULL, IDC_ARR
            // Invalidate the client area for later update
                InvalidateRect (hwnd, NULL, TRUE
                if (hBitmap == NULL)
                {
        MessageBox (    hwnd, TEXT ("Cannot load DIB file"),
                szAppName, MB_OK | MB_ICONEXCLAM
                }
```

```
                    return 0 ;
               }
               break ;

     case   WM_PAINT:
               hdc = BeginPaint (hwnd, &ps) ;

               if (hBitmap)
               {
               GetObject (hBitmap, sizeof (BITMAP), &bitmap)

                    hdcMem = CreateCompatibleDC
                    SelectObject (hdcMem, hBitmap)

          BitBlt (      hdc,  0, 0, bitmap.bmWidth, bitmap.bmHe
                        hdcMem, 0, 0, SRCCOPY) ;

                    DeleteDC (hdcMem) ;
               }

               EndPaint (hwnd, &ps) ;
               return 0 ;
```

```
        case   WM_DESTROY:
                    if (hBitmap)
                                DeleteObject (hBitmap) ;

                   PostQuitMessage (0) ;
                    return 0 ;
   }
        return DefWindowProc (hwnd, message, wParam, lParam)
}
```

```
DIBSECT.RC
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
//////////////////////////////////////////////////////////////////////////////
// Menu
DIBSECT MENU DISCARDABLE
BEGIN
```

```
  POPUP "&File"
  BEGIN
        MENUITEM "&Open",     IDM_FILE_OPEN
  END
END
```

```
RESOURCE.H
// Microsoft Developer Studio generated include file.
// Used by DIBsect.rc
#define IDM_FILE_OPEN          40001
```

DIBCONVCreateBitmapObjectFromDibFileDIBSECT CreateDIbsectionFromDibFileDIBCONVDIBCreateDIBitmap DIBSECTBITMAPFILEHEADERBITMAPINFOReadFile BITMAPINFOpBitsCreateDIBSectionpBitsDIB

pBitsDeleteObjectDIBAPIWINDOWS

DIBDIB

- SetDIBitsToDeviceStretchDIBitsDIBSetDIBitsToDeviceStretchDIBits

- CreateDIBitmapSetDIBitsDIBDDBBitBltStretchBlt CBM_INITCreateDIBitmapSetDIBits

- CreateDIBSectionDIBBitBltStretchBltBitBltStretchBlt

CreateDIBSectionDIBDIBBitBltStretchBlt

GDI pBits DIBWindows

DIBSECTpBitsCreateDIBSectionCreateDIBSection

DIB

CreateDIBitmaphdcBITMAPGetObject

CreateDIBSectionBITMAPGetObjectBITMAPINFOHEADER
DDBDIB

DIB4GDIGetObjectBITMAPbmWidthBytes2242
BITMAPINFOHEADERGetObjectbmWidthBytes86

CreateDIBSectionDIBSECTIONGetObject

```
GetObject (hBitmap, sizeof (DIBSECTION), &dibsection) ;
```

DIBSECTION

```
typedef struct tagDIBSECTION  // ds
{
        BITMAP                  dsBm ;              // BITMAP structur
        BITMAPINFOHEADER     dsBmih ;               // DIB inform
        DWORD                   dsBitfields [3] ;   // color masks
       HANDLE                   dshSection ;        // file-mapping c
        DWORD                    dsOffset ;         // offset to bitma
}
```

```
DIBSECTION, * PDIBSECTION ;
```

BITMAPBITMAPINFOHEADERCreateDIBSection

DIBSECTIONDIBGetDIBColorTable

```
hdcMem = CreateCompatibleDC (NULL) ;
SelectObject (hdcMem, hBitmap) ;
GetDIBColorTable (hdcMem, uFirstIndex, uNumEntries, &rgb) ;
DeleteDC (hdcMem) ;
```

SetDIBColorTable

CreateDIBSection

DIBDIBDIB

DIBSECTDIBCreateDIBSection

```
HBITMAP CreateDIBsectionMappingFromFile (PTSTR szFileName)
{
        BITMAPFILEHEADER            bmfh ;
        BITMAPINFO                *     pbmi ;
        BYTE                      *     pBits ;
        BOOL                       bSuccess ;
```

```
DWORD                   dwInfoSize, dwBytesRead ;
HANDLE                  hFile, hFileMap ;
HBITMAP                 hBitmap ;
hFile = CreateFile (szFileName, GENERIC_READ | GENERIC
                    0,          // No sharing!
                    NULL, OPEN_EXISTING, 0, NULL) ;

if (hFile == INVALID_HANDLE_VALUE)
               return NULL ;
bSuccess = ReadFile ( hFile, &bmfh, sizeof (BITMAPFILEHE
                    &dwBytesRead, NULL) ;

if (!bSuccess || (dwBytesRead != sizeof (BITMAPFILEHEAD
               || (bmfh.bfType != * (WORD *) "BM"))
{
               CloseHandle (hFile) ;
               return NULL ;
}
dwInfoSize = bmfh.bfOffBits - sizeof (BITMAPFILEHEADER
```

```
        pbmi = malloc (dwInfoSize) ;
        bSuccess = ReadFile (hFile, pbmi, dwInfoSize, &dwBytesR

        if (!bSuccess || (dwBytesRead != dwInfoSize))
        {
                free (pbmi) ;
                CloseHandle (hFile) ;
                return NULL ;
        }
        hFileMap = CreateFileMapping (hFile, NULL, PAGE_READW
        hBitmap = CreateDIBSection (  NULL, pbmi, DIB_RGB_COL
   free (pbmi) ;
   return hBitmap ;
}
```

CreateDIBSectiondwOffset []DWORD4414 bmfh.bfOffBits4

DIBSetDIBitsToDeviceStretchDIBitsDIB8Windows

CreateDIBitmapSetDIBitsDIBDDBBitBltStretchBlt

CreateDIBSectionWindows NTBitBltStretchBltSetDIBits

StretchDIBitsDDBDIB

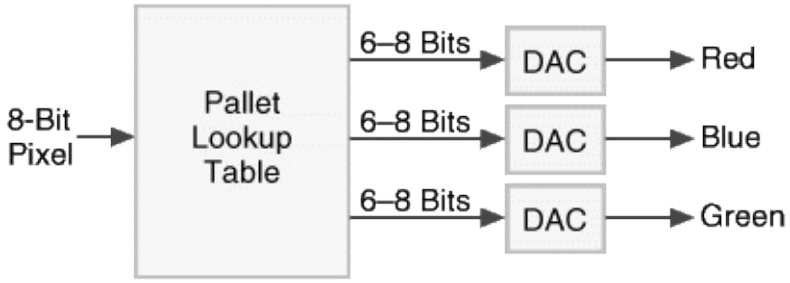
Windows

24true color168256

25616256256256

Windows88

256



88256RGBRGB24186

Microsoft \

WindowsWindows 3.0Windows25620236256254
Windows202016-1

16-1 25620

	RGB		RGB
00000000	00 00 00	11111111	FF FF FF
00000001	80 00 00	11111110	00 FF FF
00000010	00 80 00	11111101	FF 00 FF
00000011	80 80 00	11111100	00 00 FF
00000100	00 00 80	11111011	FF FF 00
00000101	80 00 80	11111010	00 FF 00
00000110	00 80 80	11111001	FF 00 00
00000111	C0 C0 C0	11111000	80 80 80
00001000	C0 DC C0	11110111	A0 A0 A4
00001001	A6 CA F0	11110110	FF FB F0

256Windows16-1logical

256

16-1GRAYS1Windows65

16-1　GRAYS1

```
GRAYS1.C
/*------------------------------------------------------------------------
  GRAYS1.C --   Gray Shades
                                             (c) Charles Petzold, 1998
------------------------------------------------------------------------*/
#include <windows.h>
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                    PSTR szCmdLine, int iCmdShow)
{
          static TCHAR szAppName[] = TEXT ("Grays1") ;
          HWND                 hwnd ;
          MSG                  msg ;
          WNDCLASS                 wndclass ;

          wndclass.style                       = CS_HREDRAW | CS_V
```

```
     wndclass.lpfnWndProc                   = WndProc ;
     wndclass.cbClsExtra                    = 0 ;
     wndclass.cbWndExtra                    = 0 ;
     wndclass.hInstance                     =      hInstance ;
     wndclass.hIcon                         =      LoadIcon (NULL, I
     wndclass.hCursor                       =      LoadCursor (NU
     wndclass.hbrBackground                 =      (HBRUSH) GetSto
    wndclass.lpszMenuName                   =      NULL ;
     wndclass.lpszClassName                 =      szAppName ;

     if (!RegisterClass (&wndclass))
     {
               MessageBox (  NULL, TEXT ("This program requi
                             szAppName, MB_ICONERROR) ;
              return 0 ;
     }

     hwnd = CreateWindow (szAppName, TEXT ("Shades of Gr
                    WS_OVERLAPPEDWINDOW,
```

```
                    CW_USEDEFAULT, CW_USEDEFAULT,
                    CW_USEDEFAULT, CW_USEDEFAULT,
                   NULL, NULL, hInstance, NULL) ;

          ShowWindow (hwnd, iCmdShow) ;
          UpdateWindow (hwnd) ;

          while (GetMessage (&msg, NULL, 0, 0))
          {
                    TranslateMessage (&msg) ;
                    DispatchMessage (&msg) ;
        }
          return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
          static int          cxClient, cyClient ;
          HBRUSH              hBrush ;
```

```
HDC                 hdc ;
int                 i ;
PAINTSTRUCT         ps ;
RECT                rect ;

switch (message)
{
case   WM_SIZE:
            cxClient = LOWORD (lParam) ;
            cyClient = HIWORD (lParam) ;
            return 0 ;

case   WM_PAINT:
            hdc = BeginPaint (hwnd, &ps) ;

                        // Draw the fountain of grays

            for (i = 0 ; i < 65 ; i++)
            {
```

```
                                rect.left            = i * cxClient / 65
                                rect.top             = 0 ;
                                rect.right           = (i + 1) * cxClie
                                rect.bottom            = cyClient ;

                        hBrush = CreateSolidBrush (RGB(min (255,
                                        min (255, 4 * i),
                                        min (255, 4 * i))) ;
                                FillRect (hdc, &rect, hBrush) ;
                                DeleteObject (hBrush) ;
                        }
                        EndPaint (hwnd, &ps) ;
                        return 0 ;

        case   WM_DESTROY:
                        PostQuitMessage (0) ;
                        return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam)
}
```

WM_PAINT65FillRectRGB0,0,04,4,48,8,8
255,255,255CreateSolidBrushmin

25665128,128,128192,192,19265
Windows20Windows

16-2GRAYS2

16-2　GRAYS2

```
GRAYS2.C

/*------------------------------------------------------------------------

  GRAYS2.C --   Gray Shades Using Palette Manager

                                          (c) Charles Petzold, 1998

------------------------------------------------------------------------*/

#include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns

                                          PSTR szCmdLine, int iCr

{

          static TCHAR szAppName[] = TEXT ("Grays2") ;

          HWND                    hwnd ;
```

```
MSG                     msg ;
WNDCLASS                wndclass ;

wndclass.style                  = CS_HREDRAW | CS_V
wndclass.lpfnWndProc                = WndProc ;
wndclass.cbClsExtra                 = 0 ;
wndclass.cbWndExtra                 = 0 ;
wndclass.hInstance                  = hInstance ;
wndclass.hIcon                  = LoadIcon (NULL, IDI_
wndclass.hCursor                = LoadCursor (NULL,
wndclass.hbrBackground          = (HBRUSH) GetStockO
wndclass.lpszMenuName           = NULL ;
wndclass.lpszClassName          = szAppName ;

if    (!    RegisterClass (&wndclass))
{
          MessageBox (  NULL, TEXT ("This program requi
                    szAppName, MB_ICONERROR) ;
     return 0 ;
}
```

```
        hwnd = CreateWindow ( szAppName, TEXT ("Shades of G
                        WS_OVERLAPPEDWINDOW,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         NULL, NULL, hInstance, NULL) ;

        ShowWindow (hwnd, iCmdShow) ;
        UpdateWindow (hwnd) ;
        while (GetMessage (&msg, NULL, 0, 0))
        {
                TranslateMessage (&msg) ;
                DispatchMessage (&msg) ;
        }
        return msg.wParam ;
}

LRESULT CALLBACK WndProc ( HWND hwnd, UINT message, WPA
{
```

```
static HPALETTE      hPalette ;
static int           cxClient, cyClient ;
HBRUSH               hBrush ;
HDC                  hdc ;
int                  i ;
LOGPALETTE           *     plp ;
PAINTSTRUCT          ps ;
RECT                 rect ;

switch (message)
{
case   WM_CREATE:
                     // Set up a LOGPALETTE structure and

          plp = malloc (sizeof (LOGPALETTE) + 64 * sizeof

          plp->palVersion      = 0x0300 ;
          plp->palNumEntries   = 65 ;

          for (i = 0 ; i < 65 ; i++)
```

```
{
            plp->palPalEntry[i].peRed     = (BYTE) min (255,
            plp->palPalEntry[i].peGreen   = (BYTE) min (255
            plp->palPalEntry[i].peBlue    = (BYTE) min (255,
            plp->palPalEntry[i].peFlags   = 0 ;
            }
            hPalette = CreatePalette (plp) ;
            free (plp) ;
            return 0 ;

 case   WM_SIZE:
            cxClient = LOWORD (lParam) ;
            cyClient = HIWORD (lParam) ;
            return 0 ;
 case   WM_PAINT:
            hdc = BeginPaint (hwnd, &ps) ;

                        // Select and realize the palette in the

            SelectPalette (hdc, hPalette, FALSE) ;
```

```
                RealizePalette (hdc) ;

                        // Draw the fountain of grays

                for (i = 0 ; i < 65 ; i++)
                {
                        rect.left            = i * cxClient / 64 ;
                        rect.top             = 0 ;
                        rect.right           = (i + 1) * cxClient / 6
                        rect.bottom            = cyClient ;

                    hBrush = CreateSolidBrush (PALETTERGB(
                                        min   (255, 4 * i),
                                        min   (255, 4 * i))) ;
                        FillRect (hdc, &rect, hBrush) ;
                        DeleteObject (hBrush) ;
                }
                EndPaint (hwnd, &ps) ;
                return 0 ;
```

```
        case   WM_QUERYNEWPALETTE:
                    if (!hPalette)
                                return FALSE ;

                    hdc = GetDC (hwnd) ;
                    SelectPalette (hdc, hPalette, FALSE) ;
                    RealizePalette (hdc) ;
                    InvalidateRect (hwnd, NULL, TRUE) ;

                    ReleaseDC (hwnd, hdc) ;
                    return TRUE ;

        case   WM_PALETTECHANGED:
                    if (!hPalette || (HWND) wParam == hwnd)
                                break ;

                    hdc = GetDC (hwnd) ;
                    SelectPalette (hdc, hPalette, FALSE) ;
                    RealizePalette (hdc) ;
                    UpdateColors (hdc) ;
```

```
                    ReleaseDC (hwnd, hdc) ;
                    break ;

          case   WM_DESTROY:
                    DeleteObject (hPalette) ;
                    PostQuitMessage (0) ;
                    return 0 ;
     }
          return DefWindowProc (hwnd, message, wParam, lParam)
}
```

CreatePalette236GRAYS1WM_CREATE LOGPALETTElogical paletteCreatePaletteCreatePalette hPalette

LOGPALETTE

```
typedef struct
{
          WORD        palVersion ;
          WORD        palNumEntries ;
          PALETTEENTRY palPalEntry[1] ;
}
```

```
LOGPALETTE, * PLOGPALETTE ;
```

0x0300Windows 3.0LOGPALETTEPALETTEENTRY
PALETTEENTRY

```
typedef struct
{
        BYTE peRed ;
        BYTE peGreen ;
        BYTE peBlue ;
       BYTE peFlags ;
}
PALETTEENTRY, * PPALETTEENTRY ;
```

PALETTEENTRYRGB

LOGPALETTEPALETTEENTRYLOGPALETTEPALETTEENTRY
GRAYS265LOGPALETTE64PALETTEENTRYGRAYS2
palNumEntries650644255peRedpeGreenpeBluepeFlags
0CreatePalette

GDIWndProcWM_DESTROYDeleteObject

WM_PAINTSelectPaletteSelectObjectFALSE
SelectPaletteTRUE

RealizePaletteWindowsWindows

GRAYS1RGBRGB32COLORREF03

WindowsRGBRGBPALETTERGBCOLORREF20
RGBRGB

- RGBRGBRGB
- RGB
- RGB

GRAYS2WM_PAINTRGB

GRAYS2256GRAYS2GRAYS2GRASY1

20WindowsPALETTEENTRYpeFlags
PC_NOCOLLAPSEPC_EXPLICITPC_RESERVED

Windows

QM_QUERYNEWPALETTEGRAYS2
RealizePaletteWM_PAINTTRUEFALSE

WM_QUERYNEWPALETTEWindowsWM_PALETTECHANGED
wParamwParam

WM_PALETTECHANGEDSelectPaletteRealizePaletteRealizePalette
WindowsRGBRGBWindows20

WM_PALETTECHANGEDGRAYS2
WM_QUERYNEWPALETTERealizePalette
WM_QUERYNEWPALETTEInvalidateRectGRAYS2UpdateColors

WM_QUERYNEWPALETTEWM_PALETTECHANGEDGRAYS2

16-3GRAYS3GRAYS2WM_PAINTPALETTEINDEXPALETTERGB

16-3　GRAYS3

```
GRAYS3.C
/*------------------------------------------------------------------------
  GRAYS3.C --   Gray Shades Using Palette Manager
                                             (c) Charles Petzold, 1998
------------------------------------------------------------------------*/
#include <windows.h>
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                                PSTR szCmdLine, int iCr
{
          static TCHAR szAppName[] = TEXT ("Grays3") ;
    HWND                                hwnd ;
          MSG                           msg ;
          WNDCLASS                          wndclass ;
```

```
     wndclass.style                  = CS_HREDRAW | CS_V
     wndclass.lpfnWndProc               = WndProc ;
     wndclass.cbClsExtra                = 0 ;
     wndclass.cbWndExtra                = 0 ;
     wndclass.hInstance                 = hInstance ;
     wndclass.hIcon                  = LoadIcon (NULL, IDI_
     wndclass.hCursor                   = LoadCursor (NULL,
     wndclass.hbrBackground          = (HBRUSH) GetStockO
     wndclass.lpszMenuName           = NULL ;
     wndclass.lpszClassName          = szAppName ;

     if (!RegisterClass (&wndclass))
{
          MessageBox (  NULL, TEXT ("This program require
                              szAppName, MB_ICONERROR
           return 0 ;
     }

     hwnd = CreateWindow ( szAppName, TEXT ("Shades of G
```

```
                        WS_OVERLAPPEDWINDOW,
                       CW_USEDEFAULT, CW_USEDEFAULT,
                        CW_USEDEFAULT, CW_USEDEFAULT,
                        NULL, NULL, hInstance, NULL) ;

        ShowWindow (hwnd, iCmdShow) ;
        UpdateWindow (hwnd) ;

        while (GetMessage (&msg, NULL, 0, 0))
        {
                TranslateMessage (&msg) ;
                DispatchMessage (&msg) ;
        }
        return msg.wParam ;
}

LRESULT CALLBACK WndProc ( HWND hwnd, UINT message, WPA
{
        static HPALETTE             hPalette ;
```

```
        static int                  cxClient, cyClient ;
        HBRUSH                      hBrush ;
HDC                         hdc ;
        int                         i ;
       LOGPALETTE                   *       plp ;
        PAINTSTRUCT                 ps ;
        RECT                        rect ;
        switch (message)
 {
        case   WM_CREATE:
                        //     Set up a LOGPALETTE structure and cr

                plp = malloc (sizeof (LOGPALETTE) + 64 * sizeof

                plp->palVersion             = 0x0300 ;
                plp->palNumEntries    = 65 ;

                for (i = 0 ; i < 65 ; i++)
                {
                plp->palPalEntry[i].peRed     = (BYTE) min (255,
```

```
                plp->palPalEntry[i].peGreen   = (BYTE) min (255
                 plp->palPalEntry[i].peBlue    = (BYTE) min (255,
                plp->palPalEntry[i].peFlags   = 0 ;
                 }
          hPalette = CreatePalette (plp) ;
                free (plp) ;
                 return 0 ;

     case   WM_SIZE:
                 cxClient = LOWORD (lParam) ;
                 cyClient = HIWORD (lParam) ;
                return 0 ;


     case   WM_PAINT:
                 hdc = BeginPaint (hwnd, &ps) ;

                // Select and realize the palette in the device cor

                 SelectPalette (hdc, hPalette, FALSE) ;
                 RealizePalette (hdc) ;
```

```
                    // Draw the fountain of grays

          for (i = 0 ; i < 65 ; i++)
          {
                    rect.left     = i * cxClient / 64 ;
                    rect.top      = 0 ;
                    rect.right    = (i + 1) * cxClient / 64 ;
                    rect.bottom   = cyClient ;

                    hBrush = CreateSolidBrush (PALETTEIN

                    FillRect (hdc, &rect, hBrush) ;
                    DeleteObject (hBrush) ;
               }

               EndPaint (hwnd, &ps) ;
               return 0 ;

     case   WM_QUERYNEWPALETTE:
               if (!hPalette)
```

```
                    return FALSE ;

               hdc = GetDC (hwnd) ;
               SelectPalette (hdc, hPalette, FALSE) ;
               RealizePalette (hdc) ;
               InvalidateRect (hwnd, NULL, FALSE) ;

               ReleaseDC (hwnd, hdc) ;
               return TRUE ;

          case   WM_PALETTECHANGED:
               if (!hPalette || (HWND) wParam == hwnd)
                    break ;

               hdc    = GetDC (hwnd) ;
               SelectPalette (hdc, hPalette, FALSE) ;
               RealizePalette (hdc) ;
               UpdateColors (hdc) ;

              ReleaseDC (hwnd, hdc) ;
               break ;
```

```
        case   WM_DESTROY:
                    PostQuitMessage (0) ;
                    return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam)
}
```

RGB1GRAYS365064

```
PALETTEINDEX (0)
```

```
PALETTEINDEX (32)
```

```
PALETTEINDEX (64)
```

WindowsRGB

Windows1624GRAYS2GRAYS3Windows

GetDeviceCapsPASTERCAPSRC_PALETTE

```
RC_PALETTE & GetDeviceCaps (hdc, RASTERCAPS)
```

GetDeviceCaps

```
GetDeviceCaps (hdc, SIZEPALETTE)
```

8256

```
GetDeviceCaps (hdc, NUMRESERVED)
```

20Windows256236

```
GetDeviceCaps (hdc, COLORRES)
```

RGB6ADC188ADC24

Windows1812864128

WindowsRGB

```
GetSystemPaletteEntries (hdc, uStart, uNum, &pe) ;
```

PALETTEENTRY

PALETTEENTRY

```
PALETTEENTRY pe ;
```

GetSystemPaletteEntries

```
GetSystemPaletteEntries (hdc, i, 1, &pe) ;
```

i0GetDeviceCapsSIZEPALETTE255PALETTEENTRY
PALETTEENTRY

GetSystemPaletteEntries

WindowsRGBCreatePalette

```
GetPaletteEntries (hPalette, uStart, uNum, &pe) ;
```

RGBGetSystemPaletteEntries

```
SetPaletteEntries (hPalette, uStart, uNum, &pe) ;
```

ResizePalette

RGBRGB

```
iIndex = GetNearestPaletteIndex (hPalette, cr) ;
```

COLORREFGetPaletteEntriesRGB

8236GetSystemPaletteUse254

GDISetROP22binaryBitBlt

GRAYS2GRAYS3WindowsGRAYS2GRAYS3

16-120

01RGB

WindowsWindows

SYSPAL116-4GetSystemPaletteEntries

16-4　　SYSPAL1

```
SYSPAL1.C

/*------------------------------------------------------------------------
  SYSPAL1.C --    Displays system palette
                                        (c) Charles Petzold, 1998
------------------------------------------------------------------------*/

#include <windows.h>
```

```
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

TCHAR szAppName [] = TEXT ("SysPal1") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                          PSTR szCmdLine, int iCr
{
     HWND                  hwnd ;
     MSG                   msg ;
     WNDCLASS              wndclass ;
     wndclass.style                = CS_HREDRAW | CS_VR
     wndclass.lpfnWndProc             = WndProc ;
     wndclass.cbClsExtra               = 0 ;
     wndclass.cbWndExtra               = 0 ;
     wndclass.hInstance                = hInstance ;
     wndclass.hIcon                 = LoadIcon (NULL, IDI_
     wndclass.hCursor                  = LoadCursor (NULL,
     wndclass.hbrBackground            = (HBRUSH) GetStock
     wndclass.lpszMenuName                = NULL ;
```

```
    wndclass.lpszClassName                  = szAppName ;

  if (!RegisterClass (&wndclass))
    {
          MessageBox (  NULL, TEXT ("This program requires W
                                szAppName, MB_ICONERROR) ;
           return 0 ;
    }

    hwnd = CreateWindow ( szAppName, TEXT ("System Pale
           WS_OVERLAPPEDWINDOW,
           CW_USEDEFAULT, CW_USEDEFAULT,
           CW_USEDEFAULT, CW_USEDEFAULT,
           NULL, NULL, hInstance, NULL) ;

    if (!hwnd)
           return 0 ;
  ShowWindow (hwnd, iCmdShow) ;
    UpdateWindow (hwnd) ;
```

```
        while (GetMessage (&msg, NULL, 0, 0))
        {
             TranslateMessage (&msg) ;
             DispatchMessage (&msg) ;
        }
        return msg.wParam ;
}

BOOL CheckDisplay (HWND hwnd)
{
        HDC    hdc ;
        int    iPalSize ;

        hdc = GetDC (hwnd) ;
        iPalSize = GetDeviceCaps (hdc, SIZEPALETTE) ;
        ReleaseDC (hwnd, hdc) ;

        if (iPalSize != 256)
  {
         MessageBox (hwnd,TEXT ("This program requires that th
```

```
                    TEXT ("display mode have a 256-color palette."),
                          szAppName, MB_ICONERROR) ;
               return FALSE ;
          }
          return TRUE ;
}

LRESULT CALLBACK WndProc ( HWND hwnd, UINT message, WPA
{
          static int          cxClient, cyClient ;
          static SIZE          sizeChar ;
          HDC                 hdc ;
          HPALETTE            hPalette ;
          int                 i, x, y ;
          PAINTSTRUCT         ps ;
          PALETTEENTRY        pe [256] ;
          TCHAR                              szBuffer [16] ;

          switch (message)
```

```
{
  case WM_CREATE:
            if (!CheckDisplay (hwnd))
                        return -1 ;
            hdc = GetDC (hwnd) ;
            SelectObject (hdc, GetStockObject (SYSTEM_FIX
            GetTextExtentPoint32 (hdc, TEXT ("FF-FF-FF"), 1
            ReleaseDC (hwnd, hdc) ;
            return 0 ;

  case   WM_DISPLAYCHANGE:
            if (!CheckDisplay (hwnd))
                        DestroyWindow (hwnd) ;

            return 0 ;

  case   WM_SIZE:
           cxClient = LOWORD (lParam) ;
            cyClient = HIWORD (lParam) ;
            return 0 ;
```

```
case   WM_PAINT:
            hdc = BeginPaint (hwnd, &ps) ;
            SelectObject (hdc, GetStockObject (SYSTEM_FIX

            GetSystemPaletteEntries (hdc, 0, 256, pe) ;

            for (i = 0, x = 0, y = 0 ; i < 256 ; i++)
            {
            wsprintf (    szBuffer, TEXT ("%02X-%02X-%02X"
       pe[i].peRed, pe[i].peGreen, pe[i].peBlue) ;

           TextOut (hdc, x, y, szBuffer, lstrlen (szBuffer)) ;

            if ((  x += sizeChar.cx) + sizeChar.cx > cxClient
                              {
                              x = 0 ;

                              if ((  y += sizeChar.cy) > cyClient
                                        break ;
                              }
```

```
                }
                EndPaint (hwnd, &ps) ;
                return 0 ;

        case   WM_PALETTECHANGED:
                InvalidateRect (hwnd, NULL, FALSE) ;
                return 0 ;

        case   WM_DESTROY:
               PostQuitMessage (0) ;
                return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

SYSPALSIZEPALETTEGetDeviceCaps256SYSPAL1

SYSPAL1WM_PALETTECHANGEDWM_PAINTSYSPAL1
GetSystemPaletteEntries256PALETTEENTRYRGB20RGB1010
16-1

SYSPAL1256SYSPAL216-5

16-5 SYSPAL2

```
SYSPAL2.C

/*------------------------------------------------------------------------
  SYSPAL2.C --    Displays system palette
                                        (c) Charles Petzold, 1998
------------------------------------------------------------------------*/

#include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

TCHAR szAppName [] = TEXT ("SysPal2") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                    PSTR szCmdLine, int iCmdShow)
{
          HWND                  hwnd ;
          MSG                   msg ;
          WNDCLASS              wndclass ;

          wndclass.style                    = CS_HREDRAW | CS_V
          wndclass.lpfnWndProc                   = WndProc ;
```

```
        wndclass.cbClsExtra                    = 0 ;
        wndclass.cbWndExtra                    = 0 ;
        wndclass.hInstance                     = hInstance ;
wndclass.hIcon                                 = LoadIcon (NULL, IDI
        wndclass.hCursor                       = LoadCursor (NULL, I
        wndclass.hbrBackground                 = (HBRUSH) GetStockO
        wndclass.lpszMenuName                  = NULL ;
        wndclass.lpszClassName                 = szAppName ;

        if (!RegisterClass (&wndclass))
        {
                MessageBox (  NULL, TEXT ("This program requi
                                                szAppName, M
                return 0 ;
}

        hwnd = CreateWindow ( szAppName, TEXT ("System Pale
                        WS_OVERLAPPEDWINDOW,
                       CW_USEDEFAULT, CW_USEDEFAULT,
```

```
                        CW_USEDEFAULT, CW_USEDEFAULT,
                        NULL, NULL, hInstance, NULL) ;
     if (!hwnd)
               return 0 ;
     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
               TranslateMessage (&msg) ;
               DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}

BOOL CheckDisplay (HWND hwnd)
{
     HDC hdc ;
     int iPalSize ;
```

```
        hdc = GetDC (hwnd) ;
        iPalSize = GetDeviceCaps (hdc, SIZEPALETTE) ;
        ReleaseDC (hwnd, hdc) ;

   if (iPalSize != 256)
        {
  MessageBox (hwnd, TEXT("This program requires that the vide
                TEXT ("display mode have a 256-color palette."),
                        szAppName, MB_ICONERROR) ;
             return FALSE ;
        }
        return TRUE ;
}

LRESULT CALLBACK WndProc ( HWND hwnd, UINT message, WPA
{
        static HPALETTE             hPalette ;
        static int                  cxClient, cyClient ;
        HBRUSH                      hBrush ;
```

```
     HDC                    hdc ;
     int                    i, x, y ;
     LOGPALETTE             *     plp ;
     PAINTSTRUCT            ps ;
     RECT                   rect ;

     switch (message)
{
     case   WM_CREATE:
          if (!CheckDisplay (hwnd))
               return -1 ;

        plp = malloc (sizeof (LOGPALETTE) + 255 * sizeof (PA

        plp->palVersion          = 0x0300 ;
        plp->palNumEntries       = 256 ;

       for (i = 0 ; i < 256 ; i++)
        {
            plp->palPalEntry[i].peRed       = i ;
           plp->palPalEntry[i].peGreen      = 0 ;
```

```
                                plp->palPalEntry[i].peBlue    = 0
                                plp->palPalEntry[i].peFlags   = PC
                    }

                hPalette = CreatePalette (plp) ;
                free (plp) ;
                return 0 ;

    case   WM_DISPLAYCHANGE:
                    if (!CheckDisplay (hwnd))
                                DestroyWindow (hwnd) ;

                    return 0 ;

          case   WM_SIZE:
                    cxClient = LOWORD (lParam) ;
                    cyClient = HIWORD (lParam) ;
                    return 0 ;

          case   WM_PAINT:
```

```
            hdc = BeginPaint (hwnd, &ps) ;

            SelectPalette (hdc, hPalette, FALSE) ;
            RealizePalette (hdc) ;

            for (y = 0 ; y < 16 ; y++)
            for (x = 0 ; x < 16 ; x++)
            {
                        hBrush = CreateSolidBrush (PALETTEIN
                        SetRect (&rect, x     * cxClient /16, y
             (x + 1)`     *      cxClient      /      16,(y+1)      *
                        FillRect (hdc, &rect, hBrush) ;
                        DeleteObject (hBrush) ;
            }
            EndPaint (hwnd, &ps) ;
            return 0 ;

      case   WM_PALETTECHANGED:
            if ((HWND) wParam != hwnd)
                              InvalidateRect (hwnd, NULL, FALS
```

```
                return 0 ;

        case   WM_DESTROY:
                DeleteObject (hPalette) ;
                PostQuitMessage (0) ;
                return 0 ;
   }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

SYSPAL2WM_CREATE2560255peFlagsPC_EXPLICIT

WM_PAINTSYSPAL2PALETTEINDEXSYSPAL2256
101020 16-1

SYSPAL2RGB WHATCLR

SYSPALWindows

GDIRGBGDI162448

GDIBitBltStretchBlt

16-6SYSPAL3StretchBlt

16-6 SYSPAL3

```
SYSPAL3.C

/*------------------------------------------------------------------------
  SYSPAL3.C -- Displays system palette
                                                (c) Charles Petzold, 1998
------------------------------------------------------------------------*/

#include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

TCHAR szAppName [] = TEXT ("SysPal3") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                                PSTR szCmdLine, int iCr
{
          HWND                  hwnd ;
          MSG                   msg ;
          WNDCLASS              wndclass ;

          wndclass.style                    = CS_HREDRAW | CS_V
          wndclass.lpfnWndProc                = WndProc ;
          wndclass.cbClsExtra                 = 0 ;
          wndclass.cbWndExtra                 = 0 ;
```

```
        wndclass.hInstance                      = hInstance ;
        wndclass.hIcon                          = LoadIcon (NULL, IDI_
        wndclass.hCursor                        = LoadCursor (NULL,
        wndclass.hbrBackground                  = (HBRUSH) GetStockO
        wndclass.lpszMenuName                   = NULL ;
        wndclass.lpszClassName                  = szAppName ;
        if (!RegisterClass (&wndclass))
        {
                    MessageBox (  NULL, TEXT ("This program requi
                                                        szAppName, M
             return 0 ;
        }

hwnd = CreateWindow ( szAppName, TEXT ("System Palette #
                    WS_OVERLAPPEDWINDOW,
                    CW_USEDEFAULT, CW_USEDEFAULT,
                    CW_USEDEFAULT, CW_USEDEFAULT,
                   NULL, NULL, hInstance, NULL) ;
```

```
    if (!hwnd)
                    return 0 ;
    ShowWindow (hwnd, iCmdShow) ;
         UpdateWindow (hwnd) ;

          while (GetMessage (&msg, NULL, 0, 0))
          {
                    TranslateMessage (&msg) ;
                    DispatchMessage (&msg) ;
          }
          return msg.wParam ;
}

BOOL CheckDisplay (HWND hwnd)
{
          HDC hdc ;
          int iPalSize ;

          hdc = GetDC (hwnd) ;
          iPalSize = GetDeviceCaps (hdc, SIZEPALETTE) ;
```

```
        ReleaseDC (hwnd, hdc) ;

        if (iPalSize != 256)
        {
     MessageBox (hwnd,TEXT("This program requires that the vid
                    TEXT("display mode have a 256-color palette."),
                           szAppName, MB_ICONERROR) ;
                  return FALSE ;
        }
   return TRUE ;
}

LRESULT CALLBACK WndProc ( HWND hwnd, UINT message, WPA
{
        static HBITMAP hBitmap ;
        static int                cxClient, cyClient ;
        BYTE                      bits [256] ;
        HDC                       hdc, hdcMem ;
        int                      i ;
        PAINTSTRUCT                ps ;
```

```
switch (message)
{
case   WM_CREATE:
            if (!  CheckDisplay (hwnd))
                                    return -1 ;

            for (  i = 0 ; i < 256 ; i++)
                                    bits [i] = i ;

           hBitmap = CreateBitmap (16, 16, 1, 8, &bits) ;
            return 0 ;

case   WM_DISPLAYCHANGE:
            if (!CheckDisplay (hwnd))
                                DestroyWindow (hwnd) ;

           return 0 ;

case   WM_SIZE:
```

```
                cxClient = LOWORD (lParam) ;
                cyClient = HIWORD (lParam) ;
                return 0 ;

        case   WM_PAINT:
                hdc = BeginPaint (hwnd, &ps) ;

                hdcMem = CreateCompatibleDC (hdc) ;
                SelectObject (hdcMem, hBitmap) ;

                StretchBlt (hdc,    0, 0, cxClient, cyClient,
                                hdcMem, 0, 0, 16, 16, SRCCOPY) ;

                DeleteDC (hdcMem) ;
                EndPaint (hwnd, &ps) ;
                return 0 ;
        case   WM_DESTROY:
                DeleteObject (hBitmap) ;
                PostQuitMessage (0) ;
                return 0 ;
        }
```

```
        return DefWindowProc (hwnd, message, wParam, lParam)
}
```

WM_CREATESYSPAL3CreateBitmap16×1680255256256
WM_PAINTStretchBltWindows256
WM_PALETTECHANGEDSYSPAL3

Windows

Windows

RGBPALETTEENTRYpeFlagsPC_RESERVED

peFlags0GDIWindowsRGB10-10-10Windows10-10-10GDIGDIPC_RESERVEDpeFlags

WM_PAINTSelectPaletteRealizePalettePALETTEINDEX

WM_TIMERRGBPALETTEENTRYAnimatePalette

16-7BOUNCEPALANIM.C

16-7 BOUNCE

```
PALANIM.C

/*------------------------------------------------------------------------
  PALANIM.C -- Palette Animation Shell Program
                                        s(c) Charles Petzold, 1998
```

```
------------------------------------------------------------------------*/
#include <windows.h>
extern HPALETTE    CreateRoutine        (HWND) ;
extern void              PaintRoutine   (HDC, int, int) ;
extern void              TimerRoutine   (HDC, HPALETTE) ;
extern void              DestroyRoutine (HWND, HPALETTE) ;

LRESULT     CALLBA CK     WndProc  (HWND, UINT, WPARAM, LPA

extern TCHAR szAppName [] ;
extern TCHAR szTitle [] ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                          PSTR szCmdLine, int iCr
{
          HWND                        hwnd ;
          MSG                        msg ;
          WNDCLASS                  wndclass ;
          wndclass.style                    = CS_HREDRAW | CS_VRE
          wndclass.lpfnWndProc                    = WndProc ;
```

```
     wndclass.cbClsExtra                    = 0 ;
     wndclass.cbWndExtra                    = 0 ;
     wndclass.hInstance                     = hInstance ;
     wndclass.hIcon                         = LoadIcon (NULL, IDI_
     wndclass.hCursor                       = LoadCursor (NULL,
     wndclass.hbrBackground                 = (HBRUSH) GetStockO
     wndclass.lpszMenuName                  = NULL ;
     wndclass.lpszClassName                 = szAppName ;

     if (!RegisterClass (&wndclass))
     {
               MessageBox (  NULL, TEXT ("This program requi
                                                  szAppName, M
               return 0 ;
}

     hwnd = CreateWindow (szAppName, szTitle,
               WS_OVERLAPPEDWINDOW,
```

```
                    CW_USEDEFAULT, CW_USEDEFAULT,
                    CW_USEDEFAULT, CW_USEDEFAULT,
                    NULL, NULL, hInstance, NULL) ;

          if (!hwnd)
                     return 0 ;
          ShowWindow (hwnd, iCmdShow) ;
          UpdateWindow (hwnd) ;

          while (GetMessage (&msg, NULL, 0, 0))
     {
                    TranslateMessage (&msg) ;
                    DispatchMessage (&msg) ;
          }
          return msg.wParam ;
}

BOOL CheckDisplay (HWND hwnd)
{
          HDC hdc ;
```

```
        int iPalSize ;

        hdc = GetDC (hwnd) ;
        iPalSize = GetDeviceCaps (hdc, SIZEPALETTE) ;
        ReleaseDC (hwnd, hdc) ;

   if (iPalSize != 256)
        {
  MessageBox (hwnd,    TEXT ("This program requires that the v
                TEXT ("display mode have a 256-color palette."),
                        szAppName, MB_ICONERROR) ;
             return FALSE ;
        }
        return TRUE ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
        static HPALETTE      hPalette ;
        static int                    cxClient, cyClient ;
```

```
        HDC                                     hdc ;
        PAINTSTRUCT                     ps ;

        switch (message)
        {
       case   WM_CREATE:
                    if (!CheckDisplay (hwnd))
                                return -1 ;

                    hPalette = CreateRoutine (hwnd) ;
                    return 0 ;

case   WM_DISPLAYCHANGE:
                    if (!CheckDisplay (hwnd))
                                DestroyWindow (hwnd) ;

                    return 0 ;

case   WM_SIZE:
                    cxClient = LOWORD (lParam) ;
                    cyClient = HIWORD (lParam) ;
```

```
                return 0 ;

case   WM_PAINT:
                hdc = BeginPaint (hwnd, &ps) ;

                SelectPalette (hdc, hPalette, FALSE) ;
                RealizePalette (hdc) ;

                PaintRoutine (hdc, cxClient, cyClient) ;

                EndPaint (hwnd, &ps) ;
                return 0 ;
case   WM_TIMER:
                hdc = GetDC (hwnd) ;

                SelectPalette (hdc, hPalette, FALSE) ;

                TimerRoutine (hdc, hPalette) ;

                ReleaseDC (hwnd, hdc) ;
                return 0 ;
```

```
case   WM_QUERYNEWPALETTE:
            if (!hPalette)
                        return FALSE ;

            hdc = GetDC (hwnd) ;
            SelectPalette (hdc, hPalette, FALSE) ;
            RealizePalette (hdc) ;
            InvalidateRect (hwnd, NULL, TRUE) ;

            ReleaseDC (hwnd, hdc) ;
            return TRUE ;

       case   WM_PALETTECHANGED:
                  if (!hPalette || (HWND) wParam == hwnd)
                              break ;

                  hdc = GetDC (hwnd) ;
                  SelectPalette (hdc, hPalette, FALSE) ;
                  RealizePalette (hdc) ;
                  UpdateColors (hdc) ;
```

```
                    ReleaseDC (hwnd, hdc) ;
                    break ;

          case   WM_DESTROY:
                    DestroyRoutine (hwnd, hPalette) ;
                    PostQuitMessage (0) ;
                    return 0 ;
          }
          return DefWindowProc (hwnd, message, wParam, lParam)
}
```

BOUNCE.C

```
/*------------------------------------------------------------------------
  BOUNCE.C --   Palette Animation Demo
                                          (c) Charles Petzold, 1998
------------------------------------------------------------------------*/
#include <windows.h>
#define ID_TIMER 1
TCHAR szAppName    [] = TEXT ("Bounce") ;
TCHAR szTitle      [] = TEXT ("Bounce: Palette Animation Demo")
```

```
static LOGPALETTE * plp ;
HPALETTE CreateRoutine (HWND hwnd)
{
        HPALETTE            hPalette ;
        int                 i ;

        plp = malloc (sizeof (LOGPALETTE) + 33 * sizeof (PALETTE
        plp->palVersion             = 0x0300 ;
        plp->palNumEntries    = 34 ;

        for (i = 0 ; i < 34 ; i++)
        {
                    plp->palPalEntry[i].peRed     = 255 ;
                    plp->palPalEntry[i].peGreen   = (i == 0 ? 0 : 25
                    plp->palPalEntry[i].peBlue    = (i == 0 ? 0 : 255
                    plp->palPalEntry[i].peFlags   = (i == 33 ? 0 : PC
   }
   hPalette = CreatePalette (plp) ;
   SetTimer (hwnd, ID_TIMER, 50, NULL) ;
```

```
    return hPalette ;
}

void PaintRoutine (HDC hdc, int cxClient, int cyClient)
{
          HBRUSH hBrush ;
          int          i, x1, x2, y1, y2 ;
          RECT         rect ;
                         // Draw window background using palette i

          SetRect (&rect, 0, 0, cxClient, cyClient) ;
          hBrush = CreateSolidBrush (PALETTEINDEX (33)) ;
    FillRect (hdc, &rect, hBrush) ;
    DeleteObject (hBrush) ;

                    // Draw the 33 balls

          SelectObject (hdc, GetStockObject (NULL_PEN)) ;
          for (i = 0 ; i < 33 ; i++)
          {
                    x1 =  i * cxClient / 33 ;
```

```
        x2 = (i + 1)* cxClient / 33 ;

        if (i < 9)
        {
                y1  = i * cyClient / 9 ;
                y2 = (i + 1) * cyClient / 9 ;
            }
    else if (i < 17)
            {
                y1 = (16 - i) * cyClient / 9 ;
                y2 = (17 - i) * cyClient / 9 ;
            }
    else if (i < 25)
    {
                y1 = (i - 16) * cyClient / 9 ;
                y2 = (i - 15) * cyClient / 9 ;
    }
    else
    {
```

```
                        y1 = (32 - i) * cyClient / 9 ;
                        y2 = (33 - i) * cyClient / 9 ;
                }

                hBrush = CreateSolidBrush (PALETTEINDEX (i)) ;
                SelectObject (hdc, hBrush) ;
                Ellipse (hdc, x1, y1, x2, y2) ;
                DeleteObject (SelectObject (hdc, GetStockObject (WH
        }
        return ;
}

void TimerRoutine (HDC hdc, HPALETTE hPalette)
{
  static BOOL bLeftToRight = TRUE ;
        static int  iBall ;

                // Set old ball to white
        plp->palPalEntry[iBall].peGreen = 255 ;
        plp->palPalEntry[iBall].peBlue  = 255 ;
```

```
        iBall += (bLeftToRight ? 1 : -1) ;
        if (   iBall == (bLeftToRight ? 33 : -1))
        {
                iBall = (bLeftToRight ? 31 : 1) ;
                bLeftToRight ^= TRUE ;
        }

             // Set new ball to red

        plp->palPalEntry[iBall].peGreen = 0 ;
        plp->palPalEntry[iBall].peBlue  = 0 ;

             // Animate the palette

        AnimatePalette (hPalette, 0, 33, plp->palPalEntry) ;
        return ;
}

void DestroyRoutine (HWND hwnd, HPALETTE hPalette)
{
```

```
        KillTimer (hwnd, ID_TIMER) ;

        DeleteObject (hPalette) ;

        free (plp) ;

        return ;

}
```

WindowsPALANIM.CCheckDisplaySYSPAL
WM_CREATE

PALANIM.CBOUNCE.CWM_CREATECreateRoutineBOUNCE
WM_PAINTPaintRoutineWM_TIMERTimerRoutineWM_DESTROY
DestroyRoutineBOUNCEPaintRoutineTimerRoutinePALANIM.C
PaintRoutinePALANIM.CTimerRoutineAnimatePalette
AnimatePaletteRealizePalette

BOUNCEW333433CreateRoutine
BOUNCEPALETTEENTRYpeFlags
PC_RESERVEDBOUNCEWindows50CreateRoutine

BOUNCEPaintRoutine3333032BOUNCE0

WndProcWM_TIMERTimerRoutineTimerRoutineAnimatePalette

```
AnimatePalette (hPalette, uStart, uNum, &pe) ;
```

PALETTEENTRYuStartuNumuStartPALETTEENTRY
uStartPALETTEENTRY

BOUNCEPALETTEENTRYLOGPALETTE032iBall
TimerRoutineBOUNCEPALETTEENTRY

```
AnimatePalette (hPalette, 0, 33, plp->palPalEntry) ;
```

GDI33

BOUNCESYSPAL2SYSPAL3

AnimatePaletteBOUNCEiBalliBallOld
iBallMiniBalliBallOldAnimatePalette

```
iBallMin = min (iBall, iBallOld) ;

AnimatePalette (hPal, iBallMin, 2, plp->palPalEntry + iBallMin) ;
```

PALETTEENTRY

```
PALETTEENTRY pe ;
```

TimerRoutinePALETTEENTRYAnimatePaletteiBall

```
pe.peRed         = 255 ;

        pe.peGreen          = 255 ;

        pe.peBlue           = 255 ;

        pe.peFlags          = PC_RESERVED ;

        AnimatePalette (hPalette, iBall, 1, &pe) ;
```

BOUNCEiBallPALETTEENTRYAnimatePalette

```
pe.peRed         = 255 ;
```

```
        pe.peGreen           = 0 ;
        pe.peBlue            = 0 ;
        pe.peFlags           = PC_RESERVED ;
        AnimatePalette (hPalette, iBall, 1, &pe) ;
```

16-8FADERPALANIM.C

16-8 FADER

```
FADER.C
/*------------------------------------------------------------------------
  FADER.C --    Palette Animation Demo
                                        (c) Charles Petzold, 1998
------------------------------------------------------------------------*/
#include <windows.h>
#define ID_TIMER 1
TCHAR szAppName    []     = TEXT ("Fader") ;
TCHAR szTitle      []     = TEXT ("Fader: Palette Animation Demo") ;
```

```
static LOGPALETTE lp ;

HPALETTE CreateRoutine (HWND hwnd)

{

          HPALETTE hPalette ;

          lp.palVersion                    = 0x0300 ;

         lp.palNumEntries                   = 1 ;

          lp.palPalEntry[0].peRed            = 255 ;

          lp.palPalEntry[0].peGreen           = 255 ;

          lp.palPalEntry[0].peBlue           = 255 ;

          lp.palPalEntry[0].peFlags          = PC_RESERVED ;


          hPalette = CreatePalette (&lp) ;

          SetTimer (hwnd, ID_TIMER, 50, NULL) ;

          return hPalette ;

}

void       PaintRoutine (HDC hdc, int cxClient, int cyClient)

{

  static TCHAR         szText [] = TEXT (" Fade In and Out ") ;
```

```
  int                                   x, y ;
        SIZE                                    sizeText ;

        SetTextColor (hdc, PALETTEINDEX (0)) ;
        GetTextExtentPoint32 (hdc, szText, lstrlen (szText), &sizeT

        for (x = 0 ; x < cxClient ; x += sizeText.cx)
        for (y = 0 ; y < cyClient ; y += sizeText.cy)
        {
              TextOut (hdc, x, y, szText, lstrlen (szText)) ;
        }

        return ;
}

void TimerRoutine (HDC hdc, HPALETTE hPalette)
{
        static BOOL bFadeIn = TRUE ;
        if (bFadeIn)
        {
                    lp.palPalEntry[0].peRed   -= 4 ;
```

```
                lp.palPalEntry[0].peGreen -= 4 ;

                if (   lp.palPalEntry[0].peRed == 3)
                                bFadeIn = FALSE ;
        }
        else
        {
                lp.palPalEntry[0].peRed   += 4 ;
                lp.palPalEntry[0].peGreen += 4 ;

                if     (lp.palPalEntry[0].peRed == 255)
                                bFadeIn = TRUE ;
        }

        AnimatePalette (hPalette, 0, 1, lp.palPalEntry) ;
        return ;
}

void DestroyRoutine (HWND hwnd, HPALETTE hPalette)
{
```

```
        KillTimer (hwnd, ID_TIMER) ;

        DeleteObject (hPalette) ;

        return ;

}
```

FADERFade In And OutFADER

FADERCreateRoutine255PaintRoutinePALANIM
FADERSetTextColorPALETTEINDEX(0)FADERFade
Out

TimerRoutineFADERPALETTEENTRYAnimatePalette
WM_TIMER434255

16-9ALLCOLOR18262144554

16-9 ALLCOLOR

```
ALLCOLOR.C

/*------------------------------------------------------------------------

  ALLCOLOR.C --         Palette Animation Demo

                                                (c) Charles Petzold, 1998

------------------------------------------------------------------------*/

#include <windows.h>

#define ID_TIMER    1
```

```
TCHAR szAppName    [] = TEXT ("AllColor") ;
TCHAR szTitle      [] = TEXT ("AllColor: Palette Animation Demo")

static     int                          iIncr ;
static     PALETTEENTRY pe ;

HPALETTE CreateRoutine (HWND hwnd)
{
          HDC                hdc ;
    HPALETTE                    hPalette ;
          LOGPALETTE           lp ;

                    // Determine the color resolution and set iIncr
          hdc = GetDC (hwnd) ;
          iIncr = 1 << (8 - GetDeviceCaps (hdc, COLORRES) / 3) ;
          ReleaseDC (hwnd, hdc) ;

                    // Create the logical palette
          lp.palVersion                = 0x0300 ;
          lp.palNumEntries                  = 1 ;
          lp.palPalEntry[0].peRed             = 0 ;
```

```
        lp.palPalEntry[0].peGreen           = 0 ;
        lp.palPalEntry[0].peBlue            = 0 ;
        lp.palPalEntry[0].peFlags           = PC_RESERVED ;

        hPalette = CreatePalette (&lp) ;
                    // Save global for less typing
        pe = lp.palPalEntry[0] ;
        SetTimer (hwnd, ID_TIMER, 10, NULL) ;
        return hPalette ;
}

void DisplayRGB (HDC hdc, PALETTEENTRY * ppe)
{
        TCHAR szBuffer [16] ;
        wsprintf (szBuffer, TEXT (" %02X-%02X-%02X "),
                                    ppe->peRed, ppe->peGreen, ppe
        TextOut (hdc, 0, 0, szBuffer, lstrlen (szBuffer)) ;
}
```

```
void PaintRoutine (HDC hdc, int cxClient, int cyClient)
{
        HBRUSH              hBrush ;
        RECT                    rect ;

                    // Draw Palette Index 0 on entire window

   hBrush = CreateSolidBrush (PALETTEINDEX (0)) ;
        SetRect (&rect, 0, 0, cxClient, cyClient) ;
        FillRect (hdc, &rect, hBrush) ;
        DeleteObject (SelectObject (hdc, GetStockObject (WHITE_
                    // Display the RGB value
        DisplayRGB (hdc, &pe) ;
        return ;
}

void TimerRoutine (HDC hdc, HPALETTE hPalette)
{
        static BOOL  bRedUp = TRUE, bGreenUp = TRUE, bBlueU
                                // Define new color value
```

```
        pe.peBlue += (bBlueUp ? iIncr : -iIncr) ;
        if (   pe.peBlue == (BYTE) (bBlueUp ? 0 : 256 - iIncr))
        {
                    pe.peBlue = (bBlueUp ? 256 - iIncr : 0) ;
                    bBlueUp ^= TRUE ;
                   pe.peGreen += (bGreenUp ? iIncr : -iIncr) ;
              if (   pe.peGreen == (BYTE) (bGreenUp ? 0 : 256 - iInc
        {
                          pe.peGreen = (bGreenUp ? 256 - iIncr : 0)
                                    bGreenUp ^= TRUE ;
                          pe.peRed += (bRedUp ? iIncr : -iIncr) ;
                                     if (   pe.peRed == (BYTE) (bRedU
                                     {
                                     pe.peRed = (bRedUp ? 256 - iIncr
                                     bRedUp ^= TRUE ;
                                     }
                    }
  }
```

```
                                // Animate the palette

        AnimatePalette (hPalette, 0, 1, &pe) ;

        DisplayRGB (hdc, &pe) ;

        return ;

}

void DestroyRoutine (HWND hwnd, HPALETTE hPalette)

{

        KillTimer (hwnd, ID_TIMER) ;

        DeleteObject (hPalette) ;

        return ;

}
```

ALLCOLORFADERCreateRoutineALLCOLOR PALETTEENTRYredgreenblue0PaintRoutineALLCOLOR PALETTEINDEX(0)FillRect

TimerRoutineALLCOLORPALETTEENTRYAnimatePalette ALLCOLORiIncrCreateRoutine COLORRESGetDeviceCapsGetDeviceCaps18iIncr4

ALLCOLORRGB

16-10PIPES

16-10 PIPES

PIPES.C

```
/*------------------------------------------------------------------------
  PIPES.C --           Palette Animation Demo
                                                (c) Charles Petzold, 1998
------------------------------------------------------------------------*/
#include <windows.h>
#define ID_TIMER 1
TCHAR szAppName [] = TEXT ("Pipes") ;
TCHAR szTitle   [] = TEXT ("Pipes: Palette Animation Demo") ;

static LOGPALETTE * plp ;

HPALETTE CreateRoutine (HWND hwnd)
{
          HPALETTE             hPalette ;
         int                 i ;
```

```
      plp = malloc (sizeof (LOGPALETTE) + 32 * sizeof (PALETTE
                        // Initialize the fields of the LOGPALETTE str
       plp->palVersion       = 0x300 ;
       plp->palNumEntries    = 16 ;

       for (i = 0 ; i <= 8 ; i++)
  {
                  plp->palPalEntry[i].peRed     = (BYTE) min (255,
                  plp->palPalEntry[i].peGreen   = 0 ;
                  plp->palPalEntry[i].peBlue    = (BYTE) min (255,
                  plp->palPalEntry[i].peFlags   = PC_RESERVED ;

                  plp->palPalEntry[16 - i]      = plp->palPalEntry[i]
                  plp->palPalEntry[16 + i]      = plp->palPalEntry[
                  plp->palPalEntry[32 - i]      = plp->palPalEntry[i]
       }

       hPalette = CreatePalette (plp) ;
       SetTimer (hwnd, ID_TIMER, 100, NULL) ;
       return hPalette ;
```

```
}

void PaintRoutine (HDC hdc, int cxClient, int cyClient)
{
        HBRUSH hBrush ;
        int         i ;
        RECT        rect ;

                        // Draw window background

        SetRect (&rect, 0, 0, cxClient, cyClient) ;
        hBrush = SelectObject (hdc, GetStockObject (WHITE_BRU
        FillRect (hdc, &rect, hBrush) ;

                        // Draw the interiors of the pipes

        for (i = 0 ; i < 128 ; i++)
        {
                hBrush = CreateSolidBrush (PALETTEINDEX (i %
                SelectObject (hdc, hBrush) ;

                rect.left           = (127 - i) * cxClient / 128 ;
```

```
                zrect.right          = (128 - i) * cxClient / 128 ;

                rect.top             = 4 * cyClient / 14 ;

                rect.bottom          = 5 * cyClient / 14 ;

                FillRect (hdc, &rect, hBrush) ;

                rect.left            =      i * cxClient / 128 ;

                rect.right           = (    i + 1) * cxClient / 128 ;

                rect.top             =      9      * cyClient / 14 ;

                rect.bottom          =      10     * cyClient / 14 ;

                FillRect (hdc, &rect, hBrush) ;

          DeleteObject (SelectObject (hdc, GetStockObject (WH
}

                         // Draw the edges of the pipes

     MoveToEx            (hdc, 0,  4 * cyClient / 14, NULL) ;

     LineTo              (hdc, cxClient,  4    * cyClient / 14) ;

     MoveToEx            (hdc, 0,  5 * cyClient / 14, NULL) ;

     LineTo              (hdc, cxClient,5      * cyClient / 14) ;
```

```
        MoveToEx            (hdc, 0,  9   * cyClient / 14, NULL) ;
        LineTo              (hdc, cxClient, 9 * cyClient / 14) ;

        MoveToEx            (hdc, 0,  10 * cyClient / 14, NULL) ;
        LineTo   (hdc, cxClient,            10 * cyClient / 14) ;
        return ;
}

void TimerRoutine (HDC hdc, HPALETTE hPalette)
{
        static int iIndex ;
        AnimatePalette (hPalette, 0, 16, plp->palPalEntry + iIndex
        iIndex = (iIndex + 1) % 16 ;

       return ;
}

void DestroyRoutine (HWND hwnd, HPALETTE hPalette)
{
        KillTimer (hwnd, ID_TIMER) ;
```

```
          DeleteObject (hPalette) ;
          free (plp) ;
          return ;
}
```

PIPES16

16-11TUNNEL128

16-11 TUNNEL

```
TUNNEL.C
/*------------------------------------------------------------------------
  TUNNEL.C --   Palette Animation Demo
                                          (c) Charles Petzold, 1998
------------------------------------------------------------------------*/
#include <windows.h>
#define ID_TIMER 1
TCHAR szAppName    [] = TEXT ("Tunnel") ;
TCHAR szTitle      [] = TEXT ("Tunnel: Palette Animation Demo") ;

static LOGPALETTE * plp ;
```

```
HPALETTE CreateRoutine (HWND hwnd)
{
        BYTE                byGrayLevel ;
        HPALETTE            hPalette ;
        int                 i ;

        plp = malloc (sizeof (LOGPALETTE) + 255 * sizeof (PALET
                    // Initialize the fields of the LOGPALETTE structu
        plp->palVersion      = 0x0300 ;
        plp->palNumEntries    = 128 ;

        for (i = 0 ; i < 128 ; i++)
        {
                    if (i < 64)
                                byGrayLevel = (BYTE) (4 * i) ;
                    else
                                byGrayLevel = (BYTE) min (255, 4
                   plp->palPalEntry[i].peRed     = byGrayLevel ;
                    plp->palPalEntry[i].peGreen   = byGrayLevel ;
```

```
                plp->palPalEntry[i].peBlue    = byGrayLevel ;
                plp->palPalEntry[i].peFlags   = PC_RESERVED ;

                plp->palPalEntry[i + 128].peRed      = byGrayLe
                plp->palPalEntry[i + 128].peGreen = byGrayLev
                plp->palPalEntry[i + 128].peBlue  = byGrayLeve
                plp->palPalEntry[i + 128].peFlags = PC_RESERV
        }

        hPalette = CreatePalette (plp) ;
        SetTimer (hwnd, ID_TIMER, 50, NULL) ;
        return hPalette ;
}

void PaintRoutine (HDC hdc, int cxClient, int cyClient)
{
        HBRUSH hBrush ;
        int         i ;
        RECT        rect ;
```

```
        for (i = 0 ; i < 127 ; i++)
   {
                                // Use a RECT structure for each of 128
                    rect.left   =   i * cxClient / 255 ;
                    rect.top    =   i * cyClient / 255 ;
                    rect.right  = cxClient - i * cxClient / 255 ;
                    rect.bottom = cyClient - i * cyClient / 255 ;

                    hBrush = CreateSolidBrush (PALETTEINDEX (i)) ;
                                // Fill the rectangle and delete the brus

                    FillRect (hdc, &rect, hBrush) ;
                    DeleteObject (hBrush) ;
        }
        return ;
}

void TimerRoutine (HDC hdc, HPALETTE hPalette)
```

```
{
        static int iLevel ;
        iLevel = (iLevel + 1) % 128 ;
        AnimatePalette (hPalette, 0, 128, plp->palPalEntry + iLev
        return ;
}

void DestroyRoutine (HWND hwnd, HPALETTE hPalette)
{
        KillTimer (hwnd, ID_TIMER) ;
        DeleteObject (hPalette) ;
       free (plp) ;
        return ;
}
```

TUNNEL12864

8packed

Packed DIB

packed DIB16-12PACKEDIB

PACKEDIB.H

```
/*--------------------------------------------------------------------
  PACKEDIB.H --   Header file for PACKEDIB.C
                                          (c) Charles Petzold, 1998
--------------------------------------------------------------------*/
#include <windows.h>

BITMAPINFO * PackedDibLoad (PTSTR szFileName) ;
int PackedDibGetWidth (BITMAPINFO * pPackedDib) ;
int PackedDibGetHeight (BITMAPINFO * pPackedDib) ;
int PackedDibGetBitCount (BITMAPINFO * pPackedDib) ;
int PackedDibGetRowLength (BITMAPINFO * pPackedDib) ;
int PackedDibGetInfoHeaderSize (BITMAPINFO * pPackedDib) ;
int PackedDibGetColorsUsed (BITMAPINFO * pPackedDib) ;
int PackedDibGetNumColors (BITMAPINFO * pPackedDib) ;
int PackedDibGetColorTableSize (BITMAPINFO * pPackedDib) ;
RGBQUAD * PackedDibGetColorTablePtr (BITMAPINFO * pPackedD
RGBQUAD * PackedDibGetColorTableEntry (BITMAPINFO * pPacke
```

```
BYTE * PackedDibGetBitsPtr (BITMAPINFO * pPackedDib) ;
int PackedDibGetBitsSize (BITMAPINFO * pPackedDib) ;
HPALETTE PackedDibCreatePalette (BITMAPINFO * pPackedDib) ;
```

PACKEDIB.C

```
/*----------------------------------------------------------------------
  PACKEDIB.C --         Routines for using packed DIBs
                                                (c) Charles Petzold, 1998
----------------------------------------------------------------------*/
#include <windows.h>

/*----------------------------------------------------------------------
  PackedDibLoad: Load DIB File as Packed-Dib Memory Block
----------------------------------------------------------------------*/

BITMAPINFO * PackedDibLoad (PTSTR szFileName)
{
          BITMAPFILEHEADER             bmfh ;
          BITMAPINFO                   *      pbmi ;
         BOOL                         bSuccess ;
```

```
    DWORD                   dwPackedDibSize, dwBytesRead ;
    HANDLE                  hFile ;
          // Open the file: read access, prohibit write access

    hFile = CreateFile (szFileName, GENERIC_READ, FILE_SHA
OPEN_EXISTING, FILE_FLAG_SEQUENTIAL_SCAN, NULL) ;
    if (hFile == INVALID_HANDLE_VALUE)
               return NULL ;
         // Read in the BITMAPFILEHEADER
    bSuccess = ReadFile (hFile, &bmfh, sizeof (BITMAPFILEHE
              &dwBytesRead, NULL) ;

    if (!bSuccess || (dwBytesRead != sizeof (BITMAPFILEHEAD
               || (bmfh.bfType != * (WORD *) "BM"))
    {
               CloseHandle (hFile) ;
               return NULL ;
    }

              // Allocate memory for the packed DIB & read it i
```

```
        dwPackedDibSize = bmfh.bfSize - sizeof (BITMAPFILEHEA
        pbmi = malloc (dwPackedDibSize) ;
  bSuccess = ReadFile (hFile, pbmi, dwPackedDibSize, &dwBytes
        CloseHandle (hFile) ;

        if (!bSuccess || (dwBytesRead != dwPackedDibSize))
        {
                free (pbmi) ;
                 return NULL ;
        }

        return pbmi ;
}

/*-------------------------------------------------------------------------
  Functions to get information from packed DIB
-------------------------------------------------------------------------*/

int PackedDibGetWidth (BITMAPINFO * pPackedDib)
{
```

```
        if (pPackedDib->bmiHeader.biSize == sizeof (BITMAPCOR
                return ((PBITMAPCOREINFO)pPackedDib)->bmci
        else
                return pPackedDib->bmiHeader.biWidth ;
}

int PackedDibGetHeight (BITMAPINFO * pPackedDib)
{
  if (pPackedDib->bmiHeader.biSize == sizeof (BITMAPCOREHEA
                return ((PBITMAPCOREINFO)pPackedDib)->bmci
        else
                return abs (pPackedDib->bmiHeader.biHeight) ;
}

int PackedDibGetBitCount (BITMAPINFO * pPackedDib)
{
        if (pPackedDib->bmiHeader.biSize == sizeof (BITMAPCOR
               return ((PBITMAPCOREINFO)pPackedDib)->bmciH
        else
                return pPackedDib->bmiHeader.biBitCount ;
```

```
}

int PackedDibGetRowLength (BITMAPINFO * pPackedDib)

{

        return ((     PackedDibGetWidth (pPackedDib) *

                                        PackedDibGetBitCount (pPackedD

}

/*-------------------------------------------------------------------------

  PackedDibGetInfoHeaderSize includes possible color masks!

-------------------------------------------------------------------------*/

int PackedDibGetInfoHeaderSize (BITMAPINFO * pPackedDib)

{

        if (   pPackedDib->bmiHeader.biSize == sizeof (BITMAPC

                    return ((PBITMAPCOREINFO)pPackedDib)->bmciH

        else   if (pPackedDib->bmiHeader.biSize == sizeof (BITMA

                     return pPackedDib->bmiHeader.biSize +

                (pPackedDib->bmiHeader.biCompression ==
```

```
                BI_BITFIELDS ? 12 : 0) ;
        else return pPackedDib->bmiHeader.biSize ;
}

/*-------------------------------------------------------------------
  PackedDibGetColorsUsed returns value in information header;
                could be 0 to indicate non-truncated color table!
-------------------------------------------------------------------*/

int PackedDibGetColorsUsed (BITMAPINFO * pPackedDib)
{
        if (pPackedDib->bmiHeader.biSize == sizeof (BITMAPCOR
                return 0 ;
        else
                 return pPackedDib->bmiHeader.biClrUsed ;
}

/*-------------------------------------------------------------------
  PackedDibGetNumColors is actual number of entries in color ta
-------------------------------------------------------------------*/
```

```
int PackedDibGetNumColors (BITMAPINFO * pPackedDib)
{
        int iNumColors ;
        iNumColors = PackedDibGetColorsUsed (pPackedDib) ;
        if (   iNumColors == 0 && PackedDibGetBitCount (pPacke
                iNumColors =1 << PackedDibGetBitCount (pPac

       return iNumColors ;
}

int PackedDibGetColorTableSize (BITMAPINFO * pPackedDib)
{
        if (pPackedDib->bmiHeader.biSize == sizeof (BITMAPCOR
                return PackedDibGetNumColors (pPackedDib) * s
        else
                 return PackedDibGetNumColors (pPackedDib) *
}

RGBQUAD * PackedDibGetColorTablePtr (BITMAPINFO * pPackedD
{
```

```
   if (PackedDibGetNumColors (pPackedDib) == 0)
                    return 0 ;
         return (RGBQUAD *) (((BYTE *)pPackedDib) +
                    PackedDibGetInfoHeaderSize (pPackedDib)) ;
}

RGBQUAD * PackedDibGetColorTableEntry (BITMAPINFO * pPacke
{
         if     (      PackedDibGetNumColors (pPackedDib) == 0)
                    return 0 ;

         if     (      pPackedDib->bmiHeader.biSize == sizeof (BITM
                    return (RGBQUAD *)
                              (((RGBTRIPLE *) PackedDibGetColorTab
         else
               return PackedDibGetColorTablePtr (pPackedDib) + i ;
}

/*---------------------------------------------------------------------
  PackedDibGetBitsPtr finally!
```

```
-------------------------------------------------------------------------*/

BYTE * PackedDibGetBitsPtr (BITMAPINFO * pPackedDib)
{
        return ((BYTE *) pPackedDib)+PackedDibGetInfoHeaderSi
                        PackedDibGetColorTableSize (pPackedD
}

/*-------------------------------------------------------------------------
  PackedDibGetBitsSize can be calculated from the height and ro
            if it's not explicitly in the biSizeImage field
-------------------------------------------------------------------------*/

int PackedDibGetBitsSize (BITMAPINFO * pPackedDib)
{
            if ((pPackedDib->bmiHeader.biSize != sizeof (BITMAP
                    (pPackedDib->bmiHeader.biSizeImage != 0
                    return pPackedDib->bmiHeader.biSizeImag

            return PackedDibGetHeight (pPackedDib) *
```

```
                                PackedDibGetRowLength (pPackedDib)
}

/*--------------------------------------------------------------------------
  PackedDibCreatePalette creates logical palette from PackedDib
--------------------------------------------------------------------------*/

HPALETTE PackedDibCreatePalette (BITMAPINFO * pPackedDib)
{
          HPALETTE   hPalette ;
          int        i, iNumColors ;
          LOGPALETTE    *      plp ;
          RGBQUAD    * prgb ;

          if (0 == ( iNumColors = PackedDibGetNumColors (pPacke
               return NULL ;
          plp = malloc (sizeof (LOGPALETTE) *
       (iNumColors - 1) * sizeof (PALETTEENTRY)) ;

          plp->palVersion    = 0x0300 ;
          plp->palNumEntries = iNumColors ;
```

```
        for (i = 0 ; i < iNumColors ; i++)
        {
                prgb = PackedDibGetColorTableEntry (pPackedD
                plp->palPalEntry[i].peRed    = prgb->rgbRed ;
                plp->palPalEntry[i].peGreen   = prgb->rgbGreer
                plp->palPalEntry[i].peBlue    = prgb->rgbBlue ;
                plp->palPalEntry[i].peFlags   = 0 ;
        }

        hPalette = CreatePalette (plp) ;
        free (plp) ;

        return hPalette ;
}
```

PackedDibLoadpacked DI

packed DIBpacked

```
dwPixel = PackedDibGetPixel (pPackedDib, x, y) ;
```

OS/2DIBBITMAPINFOBITMAPCOREHEADER

PackedDibCreatePaletteDIBDIBDIB162432DIBDIB

PACKEDIBSHOWDIB316-13

16-13　SHOWDIB3

```
SHOWDIB3.C
/*------------------------------------------------------------------------
SHOWDIB3.C --        Displays DIB with native palette
                                                (c) Charles Petzold, 199
------------------------------------------------------------------------*/
#include <windows.h>
#include "PackeDib.h"
#include "resource.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
TCHAR szAppName[] = TEXT ("ShowDib3") ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                PSTR szCmdLine, int iCmdShow)
{
        HWND         hwnd ;
        MSG          msg ;
```

```
     WNDCLASS     wndclass ;

     wndclass.style                    = CS_HREDRAW | CS_V
    wndclass.lpfnWndProc                   = WndProc ;
     wndclass.cbClsExtra                   = 0 ;
     wndclass.cbWndExtra                    = 0 ;
     wndclass.hInstance                    = hInstance ;
     wndclass.hIcon                     = LoadIcon (NULL, IDI_
     wndclass.hCursor                    = LoadCursor (NULL, I
     wndclass.hbrBackground            = (HBRUSH) GetStockO
     wndclass.lpszMenuName              = szAppName ;
     wndclass.lpszClassName            = szAppName ;
     if (!RegisterClass (&wndclass))
     {
               MessageBox (  NULL, TEXT ("This program requir
                             szAppName, MB_ICONERROR) ;
              return 0 ;
}
```

```
        hwnd = CreateWindow (szAppName, TEXT ("Show DIB #3
                    WS_OVERLAPPEDWINDOW,
                     CW_USEDEFAULT, CW_USEDEFAULT,
                     CW_USEDEFAULT, CW_USEDEFAULT,
                     NULL, NULL, hInstance, NULL) ;

        ShowWindow (hwnd, iCmdShow) ;
        UpdateWindow (hwnd) ;

  while (GetMessage (&msg, NULL, 0, 0))
        {
              TranslateMessage (&msg) ;
                   DispatchMessage (&msg) ;
        }
  return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
        static BITMAPINFO *   pPackedDib ;
```

```
    static HPALETTE                 hPalette ;
    static int                      cxClient, cyClient ;
    static OPENFILENAME   ofn ;
    static TCHAR         szFileName [MAX_PATH], szTitleName
    static TCHAR        szFilter[] = TEXT ("Bitmap Files (*.BMP)
TEXT ("All Files (*.*)\0*.*\0\0") ;
    HDC                                 hdc ;
    PAINTSTRUCT                             ps ;

    switch (message)
    {
    case   WM_CREATE:
                ofn.lStructSize             = sizeof (OPENFILENAM
                ofn.hwndOwner               = hwnd ;
                ofn.hInstance               = NULL ;
                ofn.lpstrFilter             = szFilter ;
                ofn.lpstrCustomFilter       = NULL ;
                ofn.nMaxCustFilter          = 0 ;
                ofn.nFilterIndex            = 0 ;
```

```
                ofn.lpstrFile             = szFileName ;
                ofn.nMaxFile               = MAX_PATH ;
                ofn.lpstrFileTitle         = szTitleName ;
                ofn.nMaxFileTitle          = MAX_PATH ;
                ofn.lpstrInitialDir        = NULL ;
                ofn.lpstrTitle             = NULL ;
                ofn.Flags                  = 0 ;
                ofn.nFileOffset            = 0 ;
                ofn.nFileExtension           = 0 ;
                ofn.lpstrDefExt            = TEXT ("bmp") ;
                ofn.lCustData              = 0 ;
                ofn.lpfnHook               = NULL ;
                ofn.lpTemplateName           = NULL ;

                return 0 ;

        case   WM_SIZE:
                cxClient = LOWORD (lParam) ;
                cyClient = HIWORD (lParam) ;
```

```
                return 0 ;

case   WM_COMMAND:
                switch (LOWORD (wParam))
                {
                case IDM_FILE_OPEN:

                // Show the File Open dialog box

                                if (!GetOpenFileName (&ofn))
                                        return 0 ;


                // If there's an existing packed DIB, free the mer

                                if (pPackedDib)
                                {
                                        free (pPackedDib) ;
                                        pPackedDib = NULL ;
                                }


                // If there's an existing logical palette, delete it
```

```
                    if (hPalette)
                    {
                              DeleteObject (hPalette) ;
                               hPalette = NULL ;
               }
                              // Load the packed DIB in

                     SetCursor (LoadCursor (NUL
                     ShowCursor (TRUE) ;

                    pPackedDib = PackedDibLoad

                     ShowCursor (FALSE) ;
                     SetCursor (LoadCursor (NUL

                     if (pPackedDib)
                     {
          // Create the palette from the DIB color tab

          hPalette = PackedDibCreatePalette (pPacke
```

```
                                        }
                                        else
                                        {
          MessageBox (   hwnd, TEXT ("Cannot load DIB file"),
                         szAppName, 0) ;
                                        }
                                        InvalidateRect (hwnd, NULL,
                                        return 0 ;
               }
               break ;

     case   WM_PAINT:
               hdc = BeginPaint (hwnd, &ps) ;

               if (hPalette)
               {
                                 SelectPalette (hdc, hPalette, FALS
                                RealizePalette (hdc) ;
               }
```

```
            if (pPackedDib)
            SetDIBitsToDevice (hdc, 0,0,PackedDibGetWidth
                                    PackedDibGetHeight (pPacke
                        0,0,0,PackedDibGetHeight (pPackedDi
                            PackedDibGetBitsPtr (pPackedDib),
                                pPackedDib,
        DIB_RGB_COLORS) ;
            EndPaint (hwnd, &ps) ;
            return 0 ;

    case   WM_QUERYNEWPALETTE:
            if (!hPalette)
                            return FALSE ;

            hdc = GetDC (hwnd) ;
            SelectPalette (hdc, hPalette, FALSE) ;
            RealizePalette (hdc) ;
            InvalidateRect (hwnd, NULL, TRUE) ;

            ReleaseDC (hwnd, hdc) ;
```

```
                return TRUE ;

case   WM_PALETTECHANGED:
                if (!hPalette || (HWND) wParam == hwnd)
                                break ;

                hdc = GetDC (hwnd) ;
                SelectPalette (hdc, hPalette, FALSE) ;
                RealizePalette (hdc) ;
                UpdateColors (hdc) ;

                ReleaseDC (hwnd, hdc) ;
                break ;


     case   WM_DESTROY:
                if (pPackedDib)
                                free (pPackedDib) ;

                if (hPalette)
                                DeleteObject (hPalette) ;
```

```
                    PostQuitMessage (0) ;
                    return 0 ;
          }
          return DefWindowProc (hwnd, message, wParam, lParam)
}
```

SHOWDIB3.RC

```
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
//////////////////////////////////////////////////////////////////////////////
// Menu
SHOWDIB3 MENU DISCARDABLE
BEGIN
   POPUP "&File"
   BEGIN
      MENUITEM "&Open",                IDM_FILE_OPEN
   END
```

```
END
```

```
RESOURCE.H

// Microsoft Developer Studio generated include file.

// Used by ShowDib3.rc

#define IDM_FILE_OPEN        40001
```

SHOWDIB3packed DIBFile SHOWDIB3PackedDibCreatePaletteDIBSHOWDIB3DIBDIBDIB WM_DESTROYDIB

WM_PAINTSHOWDIB3SetDIBitsToDeviceDIBDIB) PACKEDIB

SHOWDIB3DIBDIB162432DIB8DIB20

DIB

16-14SHOWDIB4DIBSHOWDIB4SHOWDIB3

16-14 SHOWDIB4

```
SHOWDIB4.C

/*------------------------------------------------------------------------

  SHOWDIB4.C -- Displays DIB with "all-purpose" palette

                                        (c) Charles Petzold, 1998
```

```
------------------------------------------------------------------------*/

#include <windows.h>
#include "..\\ShowDib3\\PackeDib.h"
#include "resource.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
TCHAR szAppName[] = TEXT ("ShowDib4") ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                    PSTR szCmdLine, int iCmdShow)
{
     HWND         hwnd ;
     MSG          msg ;
     WNDCLASS     wndclass ;

     wndclass.style          = CS_HREDRAW | CS_V
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_
```

```
    wndclass.hCursor                        = LoadCursor (NULL,
    wndclass.hbrBackground                  = (HBRUSH) GetStockO
    wndclass.lpszMenuName                   = szAppName ;
    wndclass.lpszClassName                  = szAppName ;
    if (!RegisterClass (&wndclass))
    {
                MessageBox (  NULL, TEXT ("This program requi
                                                    szAppName, M
                return 0 ;
    }

    hwnd = CreateWindow (szAppName, TEXT ("Show DIB #4
                WS_OVERLAPPEDWINDOW,
                CW_USEDEFAULT, CW_USEDEFAULT,
                 CW_USEDEFAULT, CW_USEDEFAULT,
                 NULL, NULL, hInstance, NULL) ;

    ShowWindow (hwnd, iCmdShow) ;
   UpdateWindow (hwnd) ;
```

```
        while (GetMessage (&msg, NULL, 0, 0))
        {
                TranslateMessage (&msg) ;
                DispatchMessage (&msg) ;
        }
        return msg.wParam ;
}

/*------------------------------------------------------------------------
  CreateAllPurposePalette: Creates a palette suitable for a wide v
        of images; the palette has 247 entries, but 15 of ther
        duplicates or match the standard 20 colors.
------------------------------------------------------------------------*/

HPALETTE CreateAllPurposePalette (void)
{
        HPALETTE hPalette ;
        int                     i, incr, R, G, B ;
        LOGPALETTE * plp ;
```

```
plp = malloc (sizeof (LOGPALETTE) + 246 * sizeof (PALET
plp->palVersion       = 0x0300 ;
plp->palNumEntries    = 247 ;

      // The following loop calculates 31 gray shades, but 3
      //     will match the standard 20 colors

for (i = 0, G = 0, incr = 8 ; G <= 0xFF ; i++, G += incr)
{
            plp->palPalEntry[i].peRed     = (BYTE) G ;
            plp->palPalEntry[i].peGreen   = (BYTE) G ;
            plp->palPalEntry[i].peBlue    = (BYTE) G ;
            plp->palPalEntry[i].peFlags   = 0 ;

            incr = (incr == 9 ? 8 : 9) ;
}

      // The following loop is responsible for 216 entries, bu
      // them will match the standard 20 colors, and anothe
      // 4 of them will match the gray shades above.
```

```
        for (R = 0 ; R <= 0xFF ; R += 0x33)
       for (G = 0 ; G <= 0xFF ; G += 0x33)
        for (B = 0 ; B <= 0xFF ; B += 0x33)
   {
                plp->palPalEntry     [i].peRed     = (BYTE) R ;
                plp->palPalEntry     [i].peGreen   = (BYTE) G ;
                plp->palPalEntry     [i].peBlue    = (BYTE) B ;
                plp->palPalEntry     [i].peFlags   = 0 ;

                i++ ;
        }
        hPalette = CreatePalette (plp) ;
        free (plp) ;
        return hPalette ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
```

```
        static BITMAPINFO * pPackedDib ;
        static HPALETTE                 hPalette ;
        static int                      cxClient, cyClient ;
        static OPENFILENAME ofn ;
        static TCHAR       szFileName [MAX_PATH], szTitleName [M
        static TCHAR       szFilter[] =     TEXT ("Bitmap Files (*.BMP
    TEXT ("All Files (*.*)\0*.*\0\0") ;
        HDC                                         hdc ;
        PAINTSTRUCT                             ps ;
        switch (message)
{
        case   WM_CREATE:
                    ofn.lStructSize             = sizeof (OPENFILENAME
                    ofn.hwndOwner               = hwnd ;
                    ofn.hInstance               = NULL ;
                   ofn.lpstrFilter              = szFilter ;
                    ofn.lpstrCustomFilter       = NULL ;
                    ofn.nMaxCustFilter          = 0 ;
                    ofn.nFilterIndex            = 0 ;
```

```
                ofn.lpstrFile             = szFileName ;
                ofn.nMaxFile               = MAX_PATH ;
                ofn.lpstrFileTitle          = szTitleName ;
                ofn.nMaxFileTitle           = MAX_PATH ;
                ofn.lpstrInitialDir         = NULL ;
                ofn.lpstrTitle             = NULL ;
                ofn.Flags                  = 0 ;
                ofn.nFileOffset             = 0 ;
                ofn.nFileExtension           = 0 ;
                ofn.lpstrDefExt             = TEXT ("bmp") ;
                ofn.lCustData               = 0 ;
                ofn.lpfnHook               = NULL ;
                ofn.lpTemplateName           = NULL ;

                                // Create the All-Purpose Palette

                hPalette = CreateAllPurposePalette () ;
                return 0 ;

        case   WM_SIZE:
```

```
                cxClient = LOWORD (lParam) ;
                cyClient = HIWORD (lParam) ;
                return 0 ;

        case   WM_COMMAND:
                switch (LOWORD (wParam))
                {
                case   IDM_FILE_OPEN:

                                        // Show the File Open dialog

                                if (!GetOpenFileName (&ofn))
                                        return 0 ;

        // If there's an existing packed DIB, free the memory
                                if (pPackedDib)
                                {
                                        free (pPackedDib) ;
                                        pPackedDib = NULL ;
                                }
```

```
                    // Load the packed DIB into m

                SetCursor (LoadCursor (NULL, IDC
                ShowCursor (TRUE) ;

                pPackedDib = PackedDibLoad (szF

                ShowCursor (FALSE) ;
                SetCursor (LoadCursor (NULL, IDC

                if (!pPackedDib)
                {
MessageBox (   hwnd, TEXT ("Cannot load DIB file"),
                szAppName, 0) ;
                }
                InvalidateRect (hwnd, NULL, TRUE
                return 0 ;
        }
        break ;
```

```
        case   WM_PAINT:
                    hdc = BeginPaint (hwnd, &ps) ;

                    if (pPackedDib)
                    {
                                        SelectPalette (hdc, hPalette, FALS
                                        RealizePalette (hdc) ;

                          SetDIBitsToDevice (hdc,0,0,PackedDibGetW
                                                        PackedDibGetHeigh
                                0,0,0,PackedDibGetHeight (pPackedDi
                                        PackedDibGetBitsPtr (pPackedDib)
                                        pPackedDib,
                DIB_RGB_COLORS) ;
                    }
                    EndPaint (hwnd, &ps) ;
                    return 0 ;

        case   WM_QUERYNEWPALETTE:
```

```
                hdc = GetDC (hwnd) ;
                SelectPalette (hdc, hPalette, FALSE) ;
                RealizePalette (hdc) ;
                InvalidateRect (hwnd, NULL, TRUE) ;

                ReleaseDC (hwnd, hdc) ;
                return TRUE ;

        case   WM_PALETTECHANGED:
                if ((HWND) wParam != hwnd)

                hdc = GetDC (hwnd) ;
                SelectPalette (hdc, hPalette, FALSE) ;
                RealizePalette (hdc) ;
                UpdateColors (hdc) ;

                ReleaseDC (hwnd, hdc) ;
                break ;

        case   WM_DESTROY:
```

```
                    if (pPackedDib)
                                        free (pPackedDib) ;

                    DeleteObject (hPalette) ;

                    PostQuitMessage (0) ;
                    return 0 ;
          }
          return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

```
SHOWDIB4.RC
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
//////////////////////////////////////////////////////////////////////////////
// Menu
SHOWDIB4 MENU DISCARDABLE
BEGIN
   POPUP "&Open"
```

```
  BEGIN
                    MENUITEM "&File",                   IDM_FILE_OPEN
  END
END
```

```
RESOURCE.H

// Microsoft Developer Studio generated include file.

// Used by ShowDib4.rc

#define IDM_FILE_OPEN           40001
```

WM_CREATESHOWDIB4CreateAllPurposePaletteWM_DESTROY
WM_PAINTWM_QUERYNEWPALETTE
WM_PALETTECHANGED

CreateAllPurposePalette2472361520

CreateAllPurposePalette310x000x090x110x1A0x220x2B0x33
0x3C0x440x4D0x550x5E0x660x6F0x770x800x880x91
0x990xA20xAA0xB30xBB0xC40xCC0xD50xDD0xE6
0xEE0xF90xFF200x000x330x660x990xCC
0xFF2168204PALETTEENTRYpeFlags0Windows

162432DIBDIB8DIBSHOWDIB4SHOWDIB4
SHOWDIB38DIBDIBSHOWDIB4

SHOWDIB4CreateAllPurposePaletteWindows20

Windows APICreateHalftonePaletteSHOWDIB4

CreateAllPurposePaletteHALFTONESetStretchBltMode16-15 SHOWDIB5

```
16-15  SHOWDIB5

SHOWDIB5.C

/*------------------------------------------------------------------------

  SHOWDIB5.C -- Displays DIB with halftone palette

                                                (c) Charles Petzold, 1998

------------------------------------------------------------------------*/

#include <windows.h>

#include "..\\ShowDib3\\PackeDib.h"

#include "resource.h"


LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

TCHAR szAppName[] = TEXT ("ShowDib5") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns

                                                PSTR szCmdLine, int iCr

{

        HWND                hwnd ;

        MSG                 msg ;

        WNDCLASS            wndclass ;
```

```
     wndclass.style                  = CS_HREDRAW | CS_V
     wndclass.lpfnWndProc              = WndProc ;
     wndclass.cbClsExtra               = 0 ;
     wndclass.cbWndExtra               = 0 ;
     wndclass.hInstance                = hInstance ;
     wndclass.hIcon                  = LoadIcon (NULL, IDI_
     wndclass.hCursor                = LoadCursor (NULL, I
     wndclass.hbrBackground          = (HBRUSH) GetStockO
     wndclass.lpszMenuName           = szAppName ;
     wndclass.lpszClassName          = szAppName ;

     if (!RegisterClass (&wndclass))
     {
               MessageBox (  NULL, TEXT ("This program requir
                                                szAppName, M
                return 0 ;
}

     hwnd = CreateWindow (szAppName, TEXT ("Show DIB #5
```

```
                        WS_OVERLAPPEDWINDOW,
                        CW_USEDEFAULT, CW_USEDEFAULT,
                        CW_USEDEFAULT, CW_USEDEFAULT,
                        NULL, NULL, hInstance, NULL) ;

        ShowWindow (hwnd, iCmdShow) ;
        UpdateWindow (hwnd) ;

       while (GetMessage (&msg, NULL, 0, 0))
   {
                TranslateMessage (&msg) ;
                 DispatchMessage (&msg) ;
   }
        return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
        static BITMAPINFO * pPackedDib ;
        static HPALETTE     hPalette ;
```

```
     static int         cxClient, cyClient ;
     static OPENFILENAME ofn ;
     static TCHAR       szFileName [MAX_PATH], szTitleName [M
     static TCHAR       szFilter[] =  TEXT ("Bitmap Files (*.BMP)
  TEXT ("All Files (*.*)\0*.*\0\0") ;
     HDC                                hdc ;
     PAINTSTRUCT                              ps ;

     switch (message)
{
     case   WM_CREATE:
               ofn.lStructSize             = sizeof (OPENFILENAM
               ofn.hwndOwner                  = hwnd ;
               ofn.hInstance               = NULL ;
               ofn.lpstrFilter             = szFilter ;
               ofn.lpstrCustomFilter          = NULL ;
               ofn.nMaxCustFilter          = 0 ;
               ofn.nFilterIndex            = 0 ;
               ofn.lpstrFile               = szFileName ;
```

```
     ofn.nMaxFile               = MAX_PATH ;
     ofn.lpstrFileTitle         = szTitleName ;
     ofn.nMaxFileTitle          = MAX_PATH ;
     ofn.lpstrInitialDir        = NULL ;
     ofn.lpstrTitle             = NULL ;
     ofn.Flags                  = 0 ;
     ofn.nFileOffset            = 0 ;
     ofn.nFileExtension         = 0 ;
     ofn.lpstrDefExt            = TEXT ("bmp") ;
     ofn.lCustData              = 0 ;
     ofn.lpfnHook               = NULL ;
     ofn.lpTemplateName         = NULL ;

                    // Create the All-Purpose Palette

     hdc = GetDC (hwnd) ;
     hPalette = CreateHalftonePalette (hdc) ;
     ReleaseDC (hwnd, hdc) ;
     return 0 ;
```

```
        case   WM_SIZE:
                    cxClient = LOWORD (lParam) ;
                    cyClient = HIWORD (lParam) ;
                    return 0 ;

        case   WM_COMMAND:
                    switch (LOWORD (wParam))
                    {
                   case   IDM_FILE_OPEN:

                                        // Show the File Open dialog

                                 if (!GetOpenFileName (&ofn))
                                          return 0 ;

             // If there's an existing packed DIB, free the memory

                                 if (pPackedDib)
                                 {
                                       free (pPackedDib) ;
```

```
                    pPackedDib = NULL ;
                }

                    // Load the packed DIB into m

                SetCursor (LoadCursor (NULL, IDC
                ShowCursor (TRUE) ;

                pPackedDib = PackedDibLoad (szF

                ShowCursor (FALSE) ;
                SetCursor (LoadCursor (NULL, IDC

                if (!pPackedDib)
                {
MessageBox (    hwnd, TEXT ("Cannot load DIB file"),
            szAppName, 0) ;
                }
                InvalidateRect (hwnd, NULL, TRUE
                return 0 ;
            }
```

```
            break ;

        case   WM_PAINT:
                hdc = BeginPaint (hwnd, &ps) ;

                if (pPackedDib)
                {
                                // Set halftone stretch mode

                        SetStretchBltMode (hdc, HALFTONE) ;
                        SetBrushOrgEx (hdc, 0, 0, NULL) ;

                                // Select and realize halftone pale

                        SelectPalette (hdc, hPalette, FALSE) ;
                        RealizePalette (hdc) ;

              // StretchDIBits rather than SetDIBitsToDevice
                        StretchDIBits (hdc,0,0,PackedDibGetWidth (
                                        PackedDibGetHeight (pPacke
                                        0,0,PackedDibGetWidth (pPa
```

```
                                        PackedDibGetHeight (pPacke
                                       PackedDibGetBitsPtr (pPacke
                                        pPackedDib,
          DIB_RGB_COLORS,
          SRCCOPY) ;
               }
               EndPaint (hwnd, &ps) ;
               return 0 ;

     case   WM_QUERYNEWPALETTE:
               hdc = GetDC (hwnd) ;
               SelectPalette (hdc, hPalette, FALSE) ;
               RealizePalette (hdc) ;
               InvalidateRect (hwnd, NULL, TRUE) ;

               ReleaseDC (hwnd, hdc) ;
               return TRUE ;

     case   WM_PALETTECHANGED:
               if ((HWND) wParam != hwnd)
```

```
                hdc = GetDC (hwnd) ;
                SelectPalette (hdc, hPalette, FALSE) ;
                RealizePalette (hdc) ;
                UpdateColors (hdc) ;

                ReleaseDC (hwnd, hdc) ;
                break ;

        case   WM_DESTROY:
                if (pPackedDib)
                                free (pPackedDib) ;

                DeleteObject (hPalette) ;

                PostQuitMessage (0) ;
                return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

SHOWDIB5.RC

```
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
//////////////////////////////////////////////////////////////////////////////
// Menu
SHOWDIB5 MENU DISCARDABLE
BEGIN
   POPUP "&Open"
 BEGIN
                        MENUITEM "&File",            IDM_FILE_OPEN
   END
END
```

RESOURCE.H

```
// Microsoft Developer Studio generated include file.
// Used by ShowDib5.rc
#define IDM_FILE_OPEN           40001
```

SHOWDIB5SHOWDIB4SHOWDIB4DIBSHOWDIB5Windows

CreateHalftonePalette

SHOWDIB4CreateAllPurposePalette

```
SetStretchBltMode (hdc, HALFTONE) ;

                        SetBrushOrgEx (hdc, x, y, NULL) ;
```

xyDIBStretchDIBitsSetDIBitsToDeviceDIB
CreateAllPurposePaletteCreateHalftonePaletteWindows8

SetDIBitsToDeviceStretchDIBitsCreateDIBitmapSetDIBitsGetDIBits
CreateDIBSectionfClrUseDIB_RGB_COLORS0
DIB_PAL_COLORSBITMAPINFORGB16
CreateDIBSectionNULLDIB_PAL_COLORS

DIB_PAL_COLORS8SetDIBitsToDevice8DIBWindowsDIB
DIB256DIB

16-16SHOWDIB6SHOWDIB3

16-16 SHOWDIB6

```
SHOWDIB6.C

/*------------------------------------------------------------------------

  SHOWDIB6.C -- Display DIB with palette indices

                                        (c) Charles Petzold, 1998

------------------------------------------------------------------------*/

#include <windows.h>
```

```
#include "..\\ShowDib3\\PackeDib.h"
#include "resource.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
TCHAR szAppName[] = TEXT ("ShowDib6") ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                          PSTR szCmdLine, int iCr
{
          HWND                  hwnd ;
          MSG                   msg ;
          WNDCLASS              wndclass ;

          wndclass.style                    = CS_HREDRAW | CS_V
          wndclass.lpfnWndProc                = WndProc ;
          wndclass.cbClsExtra                 = 0 ;
          wndclass.cbWndExtra                   = 0 ;
          wndclass.hInstance                  = hInstance ;
          wndclass.hIcon                    = LoadIcon (NULL, IDI_
          wndclass.hCursor                    = LoadCursor (NULL,
```

```
        wndclass.hbrBackground          = (HBRUSH) GetStockO
        wndclass.lpszMenuName           = szAppName ;
        wndclass.lpszClassName          = szAppName ;

        if (!RegisterClass (&wndclass))
        {
                MessageBox (  NULL, TEXT ("This program requir
                                                szAppName, M
                 return 0 ;
}

        hwnd = CreateWindow (szAppName, TEXT ("Show DIB #6
                WS_OVERLAPPEDWINDOW,
                CW_USEDEFAULT, CW_USEDEFAULT,
                 CW_USEDEFAULT, CW_USEDEFAULT,
                 NULL, NULL, hInstance, NULL) ;

        ShowWindow (hwnd, iCmdShow) ;
        UpdateWindow (hwnd) ;

        while (GetMessage (&msg, NULL, 0, 0))
```

```
   {
            TranslateMessage (&msg) ;
                DispatchMessage (&msg) ;
        }
        return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
        static BITMAPINFO *  pPackedDib ;
        static HPALETTE     hPalette ;
        static int          cxClient, cyClient ;
        static OPENFILENAME ofn ;
        static TCHAR        szFileName [MAX_PATH], szTitleName [M
        static TCHAR       szFilter[] = TEXT ("Bitmap Files (*.BMP)\0
     TEXT ("All Files (*.*)\0*.*\0\0") ;
        HDC                 hdc ;
        int                 i, iNumColors ;
        PAINTSTRUCT                 ps ;
```

```
WORD                    *          pwIndex ;

switch (message)
{
case   WM_CREATE:
            ofn.lStructSize             = sizeof (OPENFILENAM
            ofn.hwndOwner                 = hwnd ;
            ofn.hInstance               = NULL ;
            ofn.lpstrFilter             = szFilter ;
            ofn.lpstrCustomFilter         = NULL ;
            ofn.nMaxCustFilter            = 0 ;
            ofn.nFilterIndex            = 0 ;
            ofn.lpstrFile               = szFileName ;
            ofn.nMaxFile                = MAX_PATH ;
            ofn.lpstrFileTitle          = szTitleName ;
            ofn.nMaxFileTitle             = MAX_PATH ;
            ofn.lpstrInitialDir         = NULL ;
            ofn.lpstrTitle              = NULL ;
            ofn.Flags                   = 0 ;
```

```
                ofn.nFileOffset             = 0 ;
                ofn.nFileExtension          = 0 ;
                ofn.lpstrDefExt             = TEXT ("bmp") ;
                ofn.lCustData               = 0 ;
                ofn.lpfnHook                = NULL ;
                ofn.lpTemplateName          = NULL ;

                return 0 ;

        case   WM_SIZE:
                cxClient = LOWORD (lParam) ;
                cyClient = HIWORD (lParam) ;
                return 0 ;

        case   WM_COMMAND:
                switch (LOWORD (wParam))
                {
                case   IDM_FILE_OPEN:

                                        // Show the File Open dialog l
```

```
                    if (!GetOpenFileName (&ofn))
                              return 0 ;

          // If there's an existing packed DIB, free the mem

               if (pPackedDib)
               {
                         free (pPackedDib) ;
                         pPackedDib = NULL ;
          }

     // If there's an existing logical palette, delete it

               if (hPalette)
              {
                         DeleteObject (hPalette) ;
                        hPalette = NULL ;
               }

                    // Load the packed DIB into memory
```

```
                SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
                ShowCursor (TRUE) ;

                pPackedDib = PackedDibLoad (szFileName)

                ShowCursor (FALSE) ;
                SetCursor (LoadCursor (NULL, IDC_ARROW

                if (pPackedDib)
               {
          // Create the palette from the DIB color table

     hPalette = PackedDibCreatePalette (pPackedDib) ;

          // Replace DIB color table with indices

                     if (hPalette)
                     {
          iNumColors = PackedDibGetNumColors (pPacke
                                   pwIndex = (WORD *)
                              PackedDibGetColorTablePtr (pPac
```

```
                    for (i = 0 ; i < iNumColors ; i++)
                                                        pwIndex[i] = (
                         }
                    }
                    else
                    {
MessageBox (    hwnd, TEXT ("Cannot load DIB file"),
                   szAppName, 0) ;
                    }
                    InvalidateRect (hwnd, NULL, TRUE) ;
                    return 0 ;
               }
               break ;

 case   WM_PAINT:
               hdc = BeginPaint (hwnd, &ps) ;

              if (hPalette)
               {
```

```
                    SelectPalette (hdc, hPalette, FALS
                    RealizePalette (hdc) ;
               }

               if (pPackedDib)
                    SetDIBitsToDevice (hdc,0,0,PackedDibGetW
                         PackedDibGetHeight (pPackedDib),
                        0,0,0,PackedDibGetHeight (pPackedDi
                         PackedDibGetBitsPtr (pPackedDib),
                          pPackedDib,
          DIB_PAL_COLORS) ;
               EndPaint (hwnd, &ps) ;
               return 0 ;

     case   WM_QUERYNEWPALETTE:
               if (!hPalette)
                    return FALSE ;

               hdc = GetDC (hwnd) ;
```

```
                SelectPalette (hdc, hPalette, FALSE) ;
                RealizePalette (hdc) ;
                InvalidateRect (hwnd, NULL, TRUE) ;

                ReleaseDC (hwnd, hdc) ;
                return TRUE ;

        case   WM_PALETTECHANGED:
                if (!hPalette || (HWND) wParam == hwnd)
                                break ;

                hdc = GetDC (hwnd) ;
                SelectPalette (hdc, hPalette, FALSE) ;
                RealizePalette (hdc) ;
                UpdateColors (hdc) ;

                ReleaseDC (hwnd, hdc) ;
                break ;


        case   WM_DESTROY:
```

```
                    if (pPackedDib)
                                        free (pPackedDib) ;

                    if (hPalette)
                                        DeleteObject (hPalette) ;

                    PostQuitMessage (0) ;
                    return 0 ;
   }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

```
SHOWDIB6.RC
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
//////////////////////////////////////////////////////////////////////////////
// Menu
SHOWDIB6 MENU DISCARDABLE
BEGIN
```

```
  POPUP "&File"
  BEGIN
                    MENUITEM "&Open",                    IDM_FILE_
  END
END
```

```
RESOURCE.H
// Microsoft Developer Studio generated include file.
// Used by ShowDib6.rc
//
#define IDM_FILE_OPEN          40001
```

SHOWDIB6DIBSHOWDIB60WORDDIBPackedDibGetNumColors PackedDibGetColorTablePtrDIB

DIBDIB

DIBDIBDIBDIBRGB

16-17SHOWDIB7DIBDIBCreateDIBitmapGDI

16-17 SHOWDIB7

```
SHOWDIB7.C
```

```
/*------------------------------------------------------------------------
  SHOWDIB7.C --      Shows DIB converted to DDB
                                                (c) Charles Petzold, 1998
------------------------------------------------------------------------*/
#include <windows.h>
#include "..\\ShowDib3\\PackeDib.h"
#include "resource.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
TCHAR szAppName[] = TEXT ("ShowDib7") ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                                PSTR szCmdLine, int iCm
{
  HWND                  hwnd ;
  MSG                   msg ;
  WNDCLASS              wndclass ;

        wndclass.style                  = CS_HREDRAW | CS_VR
        wndclass.lpfnWndProc                = WndProc ;
        wndclass.cbClsExtra                 = 0 ;
```

```
     wndclass.cbWndExtra                    = 0 ;
     wndclass.hInstance                     = hInstance ;
     wndclass.hIcon                         = LoadIcon (NULL, IDI_
     wndclass.hCursor                       = LoadCursor (NULL,
     wndclass.hbrBackground                 = (HBRUSH) GetStockO
     wndclass.lpszMenuName                  = szAppName ;
     wndclass.lpszClassName                 = szAppName ;

     if (!RegisterClass (&wndclass))
     {
               MessageBox (  NULL, TEXT ("This program requi
                                                  szAppName, M
               return 0 ;
}

     hwnd = CreateWindow (szAppName, TEXT ("Show DIB #7
               WS_OVERLAPPEDWINDOW,
              CW_USEDEFAULT, CW_USEDEFAULT,
               CW_USEDEFAULT, CW_USEDEFAULT,
```

```
                    NULL, NULL, hInstance, NULL) ;

          ShowWindow (hwnd, iCmdShow) ;
          UpdateWindow (hwnd) ;

          while (GetMessage (&msg, NULL, 0, 0))
          {
               TranslateMessage (&msg) ;
               DispatchMessage (&msg) ;
     }
          return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
          static HBITMAP                    hBitmap ;
          static HPALETTE              hPalette ;
          static int                   cxClient, cyClient ;
          static OPENFILENAME      ofn ;
          static TCHAR                 szFileName [MAX_PATH], szTitleN
```

```
     static TCHAR               szFilter[] = TEXT ("Bitmap Files (*
  TEXT ("All Files (*.*)\0*.*\0\0") ;
     BITMAP                     bitmap ;
     BITMAPINFO                 *   pPackedDib ;
     HDC                        hdc, hdcMem ;
     PAINTSTRUCT                ps ;

switch (message)
{
     case   WM_CREATE:
                ofn.lStructSize       = sizeof (OPENFILENAME) ;
                ofn.hwndOwner         = hwnd ;
                ofn.hInstance         = NULL ;
                ofn.lpstrFilter       = szFilter ;
                ofn.lpstrCustomFilter = NULL ;
               ofn.nMaxCustFilter    = 0 ;
                ofn.nFilterIndex      = 0 ;
                ofn.lpstrFile         = szFileName ;
                ofn.nMaxFile          = MAX_PATH ;
```

```
                ofn.lpstrFileTitle     = szTitleName ;
                ofn.nMaxFileTitle      = MAX_PATH ;
                ofn.lpstrInitialDir    = NULL ;
                ofn.lpstrTitle         = NULL ;
                ofn.Flags              = 0 ;
                ofn.nFileOffset        = 0 ;
                ofn.nFileExtension     = 0 ;
                ofn.lpstrDefExt        = TEXT ("bmp") ;
                ofn.lCustData          = 0 ;
                ofn.lpfnHook           = NULL ;
                ofn.lpTemplateName     = NULL ;

                return 0 ;

        case   WM_SIZE:
                cxClient = LOWORD (lParam) ;
                cyClient = HIWORD (lParam) ;
                return 0 ;

        case   WM_COMMAND:
```

```
        switch (LOWORD (wParam))
        {
        case IDM_FILE_OPEN:

                    // Show the File Open dialog box

                   if (!GetOpenFileName (&ofn))
                  return 0 ;


  // If there's an existing packed DIB, free the memory

  if (hBitmap)
                  {
                              DeleteObject (hBitmap) ;
                              hBitmap = NULL ;
        }


 // If there's an existing logical palette, delete it

  if (hPalette)
                  {
```

```
                    DeleteObject (hPalette) ;
                    hPalette = NULL ;
               }

                    // Load the packed DIB into memory

SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
ShowCursor (TRUE) ;

               pPackedDib = PackedDibLoad (szFileName)

               ShowCursor (FALSE) ;
               SetCursor (LoadCursor (NULL, IDC_ARROW

              if (pPackedDib)
             {
   // Create palette from the DIB and select it into DC

hPalette = PackedDibCreatePalette (pPackedDib) ;

                    hdc = GetDC (hwnd) ;
```

```
                    if (hPalette)
                    {
                SelectPalette (hdc, hPalette, FALSE) ;
                                    RealizePalette (hdc) ;
                    }
                                     // Create the DDB from t
            hBitmap = CreateDIBitmap(hdc,(PBITMAPINFOHEAD
                  CBM_INIT,PackedDibGetBitsPtr (pPackedDib),
pPackedDib, DIB_RGB_COLORS) ;
                                    ReleaseDC (hwnd, hdc) ;

                // Free the packed-DIB memory

                                    free (pPackedDib) ;
                    }
                    else
                    {
            MessageBox (    hwnd, TEXT ("Cannot load DIB file"),
                    szAppName, 0) ;
```

```
        }
                                InvalidateRect (hwnd, NULL, TRUE
                                return 0 ;
                }
                break ;
        case   WM_PAINT:
                hdc = BeginPaint (hwnd, &ps) ;

                if (hPalette)
                {
                                SelectPalette (hdc, hPalette, FALSE
                                RealizePalette (hdc) ;
                }
                if (hBitmap)
                {
                                GetObject (hBitmap, sizeof (BITMAP
                                hdcMem = CreateCompatibleDC (hdc) ;
                                SelectObject (hdcMem, hBitmap) ;
```

```
                    BitBlt (hdc,0,0,bitmap.bmWidth, b
                        hdcMem,0,    0,    SRCCOPY)

                DeleteDC (hdcMem) ;

            }
            EndPaint (hwnd, &ps) ;
            return 0 ;

        case   WM_QUERYNEWPALETTE:
            if (!hPalette)
                    return FALSE ;

            hdc = GetDC (hwnd) ;
            SelectPalette (hdc, hPalette, FALSE) ;
            RealizePalette (hdc) ;
            InvalidateRect (hwnd, NULL, TRUE) ;

            ReleaseDC (hwnd, hdc) ;
            return TRUE ;

        case   WM_PALETTECHANGED:
```

```
                if (!hPalette || (HWND) wParam == hwnd)
                                        break ;

                hdc = GetDC (hwnd) ;
                SelectPalette (hdc, hPalette, FALSE) ;
               RealizePalette (hdc) ;
               UpdateColors (hdc) ;

                ReleaseDC (hwnd, hdc) ;
                break ;


    case   WM_DESTROY:
                if (hBitmap)
                                        DeleteObject (hBitmap) ;

                if (hPalette)
                                       DeleteObject (hPalette) ;

                                PostQuitMessage (0) ;
                                return 0 ;
```

```
        }
        return DefWindowProc (hwnd, message, wParam, lParam)
}
```

SHOWDIB7.RC

```
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
//////////////////////////////////////////////////////////////////////////////
// Menu
SHOWDIB7 MENU DISCARDABLE
BEGIN
  POPUP "&File"
   BEGIN
                    MENUITEM "&Open",                   IDM_FILE_OPEN
   END
END
```

RESOURCE.H

```
// Microsoft Developer Studio generated include file.

// Used by ShowDib7.rc

#define IDM_FILE_OPEN      40001
```

SHOWDIB7packed DIBDIBFileOpenpacked WM_COMMANDSHOWDIB7CreateDIBitmapDIBDDB CreateDIBitmapDDB

CreateDIBitmappacked DIBpPackedDibSHOWDIB7 hBitmaphPalette

WM_PAINTGetObjectBitBltDDBCreateDIBitmap

packed DIBWindowsWindowsDDBDIB

DIB

16-18SHOWDIB8DIB

16-18 SHOWDIB8

```
SHOWDIB8.C

/*------------------------------------------------------------------------

  SHOWDIB8.C --        Shows DIB converted to DIB section

                                                (c) Charles Petzold, 1998

------------------------------------------------------------------------*/

#include <windows.h>
```

```
#include "..\\ShowDib3\\PackeDib.h"
#include "resource.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
TCHAR szAppName[] = TEXT ("ShowDib8") ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                          PSTR szCmdLine, int iCm
{
        HWND                hwnd ;
        MSG                 msg ;
        WNDCLASS            wndclass ;

        wndclass.style                  = CS_HREDRAW | CS_VR
        wndclass.lpfnWndProc                = WndProc ;
        wndclass.cbClsExtra                 = 0 ;
        wndclass.cbWndExtra                 = 0 ;
        wndclass.hInstance                  = hInstance ;
        wndclass.hIcon                  = LoadIcon (NULL, IDI_
        wndclass.hCursor                = LoadCursor (NULL,
        wndclass.hbrBackground          = (HBRUSH) GetStockO
```

```
    wndclass.lpszMenuName  = szAppName ;
    wndclass.lpszClassName = szAppName ;

    if (!RegisterClass (&wndclass))
    {
          MessageBox (  NULL, TEXT ("This program requi
                                          szAppName, M
          return 0 ;
    }

    hwnd = CreateWindow (szAppName, TEXT ("Show DIB #8
          WS_OVERLAPPEDWINDOW,
          CW_USEDEFAULT, CW_USEDEFAULT,
          CW_USEDEFAULT, CW_USEDEFAULT,
          NULL, NULL, hInstance, NULL) ;

    ShowWindow (hwnd, iCmdShow) ;
    UpdateWindow (hwnd) ;

    while (GetMessage (&msg, NULL, 0, 0))
```

```
        {
                    TranslateMessage (&msg) ;
                    DispatchMessage (&msg) ;
        }
        return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
        static HBITMAP                  hBitmap ;
        static HPALETTE                  hPalette ;
        static int                      cxClient, cyClient ;
        static OPENFILENAME         ofn ;
        static PBYTE                    pBits ;
        static TCHAR                    szFileName [MAX_PATH], szT
        static TCHAR                    szFilter[] = TEXT ("Bitmap Fil
    TEXT ("All Files (*.*)\0*.*\0\0") ;
        BITMAP                  bitmap ;
        BITMAPINFO                      *      pPackedDib ;
```

```
HDC                         hdc, hdcMem ;
     PAINTSTRUCT                ps ;

     switch (message)
     {
     case   WM_CREATE:
                 ofn.lStructSize       = sizeof (OPENFILENAME) ;
                ofn.hwndOwner         = hwnd ;
                 ofn.hInstance         = NULL ;
                 ofn.lpstrFilter       = szFilter ;
                 ofn.lpstrCustomFilter = NULL ;
                 ofn.nMaxCustFilter    = 0 ;
                 ofn.nFilterIndex      = 0 ;
                 ofn.lpstrFile         = szFileName ;
                 ofn.nMaxFile          = MAX_PATH ;
                 ofn.lpstrFileTitle    = szTitleName ;
                 ofn.nMaxFileTitle     = MAX_PATH ;
                ofn.lpstrInitialDir   = NULL ;
                 ofn.lpstrTitle        = NULL ;
```

```
                ofn.Flags             = 0 ;
                ofn.nFileOffset       = 0 ;
                ofn.nFileExtension    = 0 ;
                ofn.lpstrDefExt       = TEXT ("bmp") ;
                ofn.lCustData         = 0 ;
                ofn.lpfnHook          = NULL ;
                ofn.lpTemplateName    = NULL ;

                return 0 ;

        case   WM_SIZE:
                cxClient = LOWORD (lParam) ;
                cyClient = HIWORD (lParam) ;
               return 0 ;

        case   WM_COMMAND:
                switch (LOWORD (wParam))
                {
                case   IDM_FILE_OPEN:

                                        // Show the File Open dialog box
```

```
                    if (!GetOpenFileName (&ofn))
                        return 0 ;

    // If there's an existing packed DIB, free the memory
                    if (hBitmap)
                    {
                        DeleteObject (hBit
                        hBitmap = NULL ;
                    }
    // If there's an existing logical palette, delete it
                if (hPalette)
                    {
                        DeleteObject (hPal
                        hPalette = NULL ;
                    }
    // Load the packed DIB into memory
```

```
                                    SetCursor (LoadCursor (NULL, IDC
                                     ShowCursor (TRUE) ;

                                    pPackedDib = PackedDibLoad (szF

                                     ShowCursor (FALSE) ;
                                     SetCursor (LoadCursor (NULL, IDC

                                     if (pPackedDib)
                                     {
                          // Create the DIB section from the DIB

hBitmap = CreateDIBSection (NULL,pPackedDib,DIB_RGB_COLOR

                     // Copy the bits

                     CopyMemory (pBits,     PackedDibGetBitsPtr  (pPacke
                                         PackedDibGetBitsSize (pPackedDib)) ;

                      // Create palette from the DIB

                     hPalette = PackedDibCreatePalette (pPackedDib) ;
```

```
            // Free the packed-DIB memory

                                    free (pPackedDib) ;
                                }
                                else
                                {
    MessageBox (       hwnd, TEXT ("Cannot load DIB file"),
                    szAppName, 0) ;
                                    }
                                 InvalidateRect (hwnd, NULL, TRUE
                                return 0 ;
                }
                break ;
case    WM_PAINT:
                hdc = BeginPaint (hwnd, &ps) ;

                if (hPalette)
                {
                            SelectPalette (hdc, hPalette, FALSE) ;
```

```
                    RealizePalette (hdc) ;
               }
               if (hBitmap)
               {
                    GetObject (hBitmap, sizeof (BITMAP), &

                   hdcMem = CreateCompatibleDC (hdc)
                    SelectObject (hdcMem, hBitmap) ;

          BitBlt (      hdc,    0, 0, bitmap.bmWidth, bitmap.bmH
                   hdcMem, 0, 0, SRCCOPY) ;

                    DeleteDC (hdcMem) ;
               }
               EndPaint (hwnd, &ps) ;
               return 0 ;

     case   WM_QUERYNEWPALETTE:
               if (!hPalette)
                         return FALSE ;
```

```
                hdc = GetDC          (hwnd) ;
                SelectPalette        (hdc, hPalette, FALSE) ;
                RealizePalette       (hdc) ;
                InvalidateRect       (hwnd, NULL, TRUE) ;

                ReleaseDC (hwnd, hdc) ;
                return TRUE ;

        case   WM_PALETTECHANGED:
                if (!hPalette || (HWND) wParam == hwnd)
                                    break ;

                hdc = GetDC (hwnd) ;
                SelectPalette (hdc, hPalette, FALSE) ;
                RealizePalette (hdc) ;
                UpdateColors (hdc) ;

                ReleaseDC (hwnd, hdc) ;
                break ;
```

```
        case   WM_DESTROY:
                    if (hBitmap)
                                DeleteObject (hBitmap) ;

                    if (hPalette)
                                 DeleteObject (hPalette) ;

                    PostQuitMessage (0) ;
                    return 0 ;
   }
        return DefWindowProc (hwnd, message, wParam, lParam)
}
```

```
SHOWDIB8.RC
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
//////////////////////////////////////////////////////////////////////////////
// Menu
SHOWDIB8 MENU DISCARDABLE
```

```
BEGIN
  POPUP "&File"
  BEGIN
                MENUITEM "&Open",                IDM_FILE_OPEI
  END
END
```

```
RESOURCE.H
// Microsoft Developer Studio generated include file.
// Used by ShowDib8.rc
#define IDM_FILE_OPEN            40001
```

SHOWDIB7SHOWDIB8WM_PAINThBitmaphPalette
GetObjectBitBlt

FileOpenpacked
SHOWDIB8packed DIBCreateDIBSectionCreateDIBSectionDIB
CreateDIBSectionDIB_PAL_COLORS

CreateDIBSectionSHOWDIB8packed DIBCreateDIBSection
PackedDibCreatePaletteSHOWDIB8GetDIBColorTable

DIB

GDIDIBWindows

PACKEDIBpacked DIBDIBget

C++DIBpacked DIBDIBCC++

C++CWindows

HDIBHDIB

```
typedef void * HDIB ;
```

HDIB

HDIBpacked DIB

```
typedef struct
{
        BITMAPINFO    *      pPackedDib ;
        int                          cx, cy, cBitsPerPixel, cBytesPerRo
       BYTE                      * pBits ;
{
DIBSTRUCTURE, * PDIBSTRUCTURE ;
```

packed DIBDIBpPackedDibDibGetPixelPointer

```
BYTE * DibGetPixelPointer (HDIB hdib, int x, int y)
{
        PDIBSTRUCTURE pdib = hdib ;
         return pdib->pBits + y * pdib->cBytesPerRow +
```

```
                                x * pdib->cBitsPerPixel / 8 ;
}
```

PACKEDIB.Cget pixel

packed DIBDIBDIBpacked

DIBSTRUCT

DIBHELP.CDIBDIBHELPDIBHELP.C

```
typedef struct
{
        PBYTE        * ppRow ;       // array of row pointers
        int          iSignature ;  // = "Dib "
        HBITMAP      hBitmap ;     // handle returned from CreateD
        BYTE         *      pBits ;     // pointer to bitmap bits
        DIBSECTION    ds ;          // DIBSECTION structure
        int          iRShift[3] ;  // right-shift values for color masks
        int         iLShift[3] ;  // left-shift values for color masks
}
DIBSTRUCT, * PDIBSTRUCT ;
```

DIBHELP.CDIBDibDIBHELP

hBitmapCreateDIBSection
BitBltStretchBlt

DIBSTRUCTCreateDIBSection

DIBSTRUCTDIBSECTIONCreateDIBSectionGetObjectDIBSECTION

```
GetObject (hBitmap, sizeof (DIBSECTION), &ds) ;
```

DIBSECTIONWINGDI.H

```
typedef struct tagDIBSECTION {
  BITMAP                  dsBm ;
 BITMAPINFOHEADER      dsBmih ;
 DWORD                dsBitfields[3] ;    // Color masks
  HANDLE          dshSection ;
  DWORD           dsOffset ;
}
DIBSECTION, * PDIBSECTION ;
```

BITMAPCreateBitmapIndirectGetObjectDDB
BITMAPINFOHEADERCreateDIBSectionDIBSECTION
BITMAPINFOHEADERBITMAPCOREHEADERDIBHELP.COS/2
DIB

1632DIBBITMAPINFOHEADERbiCompressionBI_BITFIELDS1632
RGBDIBSECTION

DIBSECTIONDIBDIBHELPCreateDIBSection

DIBSTRUCT1632DIB

DIBSTRUCTDIBDIBDIBDIBDIBDIBSTRUCT
pBits

DIBHELP.CDIBSTRUCTDIB16-19DIBHELP.C

16-19　DIBHELP.C

```
DIBHELP.C

/*--------------------------------------------------------------------------

  DIBHELP.C -- DIB Section Helper Routines

                                                (c) Charles Petzold, 1998

--------------------------------------------------------------------------*/

#include <windows.h>

#include "dibhelp.h"


#define HDIB_SIGNATURE (* (int *) "Dib ")

typedef struct

{
```

```
        PBYTE                   *      ppRow ;    // must be first field
        int                     iSignature ;
        HBITMAP                 hBitmap ;
        BYTE                    *      pBits ;
        DIBSECTION              ds ;
        int                     iRShift[3] ;
        int                     iLShift[3] ;
}
DIBSTRUCT, * PDIBSTRUCT ;

/*-------------------------------------------------------------------------
   DibIsValid:  Returns TRUE if hdib points to a valid DIBSTRUCT
-------------------------------------------------------------------------*/

BOOL DibIsValid (HDIB hdib)
{
        PDIBSTRUCT pdib = hdib ;
        if (pdib == NULL)
                return FALSE ;
```

```
        if (IsBadReadPtr (pdib, sizeof (DIBSTRUCT)))
                return FALSE ;
        if (pdib->iSignature != HDIB_SIGNATURE)
                return FALSE ;
        return TRUE ;
}

/*-------------------------------------------------------------------------
  DibBitmapHandle:      Returns the handle to the DIB section bit
-------------------------------------------------------------------------*/

HBITMAP DibBitmapHandle (HDIB hdib)
{
        if (!DibIsValid (hdib))
                return NULL ;
        return ((PDIBSTRUCT) hdib)->hBitmap ;
}

/*-------------------------------------------------------------------------
  DibWidth:  Returns the bitmap pixel width
```

```
-------------------------------------------------------------------------*/

int DibWidth (HDIB hdib)
{
        if (!DibIsValid (hdib))
                return 0 ;
        return ((PDIBSTRUCT) hdib)->ds.dsBm.bmWidth ;
}

/*-------------------------------------------------------------------------
  DibHeight:  Returns the bitmap pixel height
-------------------------------------------------------------------------*/

int DibHeight (HDIB hdib)
{
   if (!DibIsValid (hdib))
                return 0 ;
        return ((PDIBSTRUCT) hdib)->ds.dsBm.bmHeight ;
}

/*-------------------------------------------------------------------------
```

```
  DibBitCount:    Returns the number of bits per pixel
-------------------------------------------------------------------------*/

int DibBitCount (HDIB hdib)
{
        if (!DibIsValid (hdib))
                return 0 ;

        return ((PDIBSTRUCT) hdib)->ds.dsBm.bmBitsPixel ;
}

/*-------------------------------------------------------------------------
  DibRowLength:  Returns the number of bytes per row of pixels
-------------------------------------------------------------------------*/

int DibRowLength (HDIB hdib)
{
  if (!DibIsValid (hdib))
                return 0 ;
        return 4 * ((DibWidth (hdib) * DibBitCount (hdib) + 31) / 3
```

```
}

/*----------------------------------------------------------------------
  DibNumColors:  Returns the number of colors in the color table
----------------------------------------------------------------------*/

int DibNumColors (HDIB hdib)
{
        PDIBSTRUCT pdib = hdib ;

        if (!DibIsValid (hdib))
                return 0 ;

        if (pdib->ds.dsBmih.biClrUsed != 0)
   {
                return pdib->ds.dsBmih.biClrUsed ;
   }
        else if (DibBitCount (hdib) <= 8)
   {
             return 1 << DibBitCount (hdib) ;
```

```
   }

        return 0 ;

}

/*-------------------------------------------------------------------------

  DibMask:  Returns one of the color masks

-------------------------------------------------------------------------*/

DWORD       DibMask (HDIB hdib, int i)

{

        PDIBSTRUCT pdib = hdib ;

        if (!DibIsValid (hdib) || i < 0 || i > 2)

                return 0 ;


        return pdib->ds.dsBitfields[i] ;

}

/*-------------------------------------------------------------------------

  DibRShift:  Returns one of the right-shift values

-------------------------------------------------------------------------*/
```

```
int DibRShift (HDIB hdib, int i)
{
        PDIBSTRUCT pdib = hdib ;

        if (!DibIsValid (hdib) || i < 0 || i > 2)
                return 0 ;

        return pdib->iRShift[i] ;
}

/*-------------------------------------------------------------------------
  DibLShift:  Returns one of the left-shift values
-------------------------------------------------------------------------*/

int DibLShift (HDIB hdib, int i)
{
        PDIBSTRUCT pdib = hdib ;

        if (!DibIsValid (hdib) || i < 0 || i > 2)
                return 0 ;
```

```
        return pdib->iLShift[i] ;
}

/*-------------------------------------------------------------------------
  DibCompression:  Returns the value of the biCompression field
-------------------------------------------------------------------------*/
int DibCompression (HDIB hdib)
{
        if (!DibIsValid (hdib))
                return 0 ;

        return ((PDIBSTRUCT) hdib)->ds.dsBmih.biCompression ;
}

/*-------------------------------------------------------------------------
  DibIsAddressable:  Returns TRUE if the DIB is not compressed
-------------------------------------------------------------------------*/

BOOL       DibIsAddressable (HDIB hdib)
{
```

```
        int iCompression ;

        if (!DibIsValid (hdib))
                return FALSE ;

        iCompression = DibCompression (hdib) ;

        if (   iCompression == BI_RGB || iCompression == BI_BITF
                return TRUE ;

        return FALSE ;
}

/*-------------------------------------------------------------------------
  These functions return the sizes of various components of the D
        AS THEY WOULD APPEAR in a packed DIB. These function
    the DIB section to a packed DIB and in saving DIB files.
-------------------------------------------------------------------------*/

DWORD DibInfoHeaderSize (HDIB hdib)
{
```

```
        if (!DibIsValid (hdib))
                return 0 ;
        return ((PDIBSTRUCT) hdib)->ds.dsBmih.biSize ;
}

DWORD       DibMaskSize (HDIB hdib)
{
       PDIBSTRUCT pdib = hdib ;
        if (!DibIsValid (hdib))
                return 0 ;

   if (pdib->ds.dsBmih.biCompression == BI_BITFIELDS)
                return 3 * sizeof (DWORD) ;

        return 0 ;
}

DWORD DibColorSize (HDIB hdib)
{
   return DibNumColors (hdib) * sizeof (RGBQUAD) ;
}
```

```
DWORD DibInfoSize (HDIB hdib)
{
        return DibInfoHeaderSize(hdib) + DibMaskSize(hdib) + Di
}

DWORD DibBitsSize (HDIB hdib)
{
        PDIBSTRUCT pdib = hdib ;

        if (!DibIsValid (hdib))
                return 0 ;

        if (pdib->ds.dsBmih.biSizeImage != 0)
   {
                return pdib->ds.dsBmih.biSizeImage ;
   }
        return DibHeight (hdib) * DibRowLength (hdib) ;
}

DWORD DibTotalSize (HDIB hdib)
```

```
{
        return DibInfoSize (hdib) + DibBitsSize (hdib) ;
}

/*-------------------------------------------------------------------------
  These functions return pointers to the various components of t
        section.
-------------------------------------------------------------------------*/
BITMAPINFOHEADER * DibInfoHeaderPtr (HDIB hdib)
{
        if (!DibIsValid (hdib))
                return NULL ;
        return & (((PDIBSTRUCT) hdib)->ds.dsBmih) ;
}

DWORD * DibMaskPtr (HDIB hdib)
{
        PDIBSTRUCT pdib = hdib ;
        if (!DibIsValid (hdib))
```

```
                    return 0 ;

          return pdib->ds.dsBitfields ;

}

void * DibBitsPtr (HDIB hdib)

{

          if (!DibIsValid (hdib))

                    return NULL ;

          return ((PDIBSTRUCT) hdib)->pBits ;

}

/*-------------------------------------------------------------------------

  DibSetColor:  Obtains entry from the DIB color table

-------------------------------------------------------------------------*/

BOOL DibGetColor (HDIB hdib, int index, RGBQUAD * prgb)

{

          PDIBSTRUCT pdib = hdib ;

          HDC                         hdcMem ;

          int                         iReturn ;
```

```
        if (!DibIsValid (hdib))
                return 0 ;
        hdcMem = CreateCompatibleDC (NULL) ;
        SelectObject (hdcMem, pdib->hBitmap) ;
        iReturn = GetDIBColorTable (hdcMem, index, 1, prgb) ;
        DeleteDC (hdcMem) ;
        return iReturn ? TRUE : FALSE ;
}

/*-------------------------------------------------------------------------
  DibGetColor:  Sets an entry in the DIB color table
-------------------------------------------------------------------------*/

BOOL DibSetColor (HDIB hdib, int index, RGBQUAD * prgb)
{
        PDIBSTRUCT    pdib = hdib ;
        HDC                     hdcMem ;
        int                     iReturn ;

        if (!DibIsValid (hdib))
```

```
                return 0 ;

        hdcMem = CreateCompatibleDC (NULL) ;

        SelectObject (hdcMem, pdib->hBitmap) ;

        iReturn = SetDIBColorTable (hdcMem, index, 1, prgb) ;

        DeleteDC (hdcMem) ;

        return iReturn ? TRUE : FALSE ;
}
```

DIBHELP.CDibIsValidDIBSTRUCTDibIsValidHDIB(
DIBHELP.HPDIBSTRUCT

BOOLDibIsAddressableDibIsNotCompressedDIB

DibInfoHeaderSizeDIBpacked DIBDIBpa

DIBHELP.CDibInfoHeaderPtrBITMAPINFOHEADERBITMAPINFO
DIBDIBBITMAPINFOHEADERDIBSECTION
CreateDIBSectionGetDIBColorTableSetDIBColorTableDIBDIBHELP
DibGetColorDibSetColor

DIBHELP.CDibCopyToInfoBITMAPINFO

packed DIBDIBDIB16-20DIBHELP.C

16-20 DIBHELP.C

```

```

```
DIBHELP.C

/*-------------------------------------------------------------------------
  DibPixelPtr:  Returns a pointer to the pixel at position (x, y)
-------------------------------------------------------------------------*/

BYTE * DibPixelPtr (HDIB hdib, int x, int y)
{
          if (!DibIsAddressable (hdib))
                    return NULL ;
          if (x < 0 || x >= DibWidth (hdib) || y < 0 || y >= DibHeigh
                    return NULL ;
          return (((PDIBSTRUCT) hdib)->ppRow)[y] + (x * DibBitCou
}

/*-------------------------------------------------------------------------
  DibGetPixel:  Obtains a pixel value at (x, y)
-------------------------------------------------------------------------*/

DWORD DibGetPixel (HDIB hdib, int x, int y)
{
          PBYTE pPixel ;
```

```
        if (!(pPixel = DibPixelPtr (hdib, x, y)))
                return 0 ;
        switch (DibBitCount (hdib))
        {
        case        1:  return 0x01 & (* pPixel >> (7 - (x & 7))) ;
        case        4:  return 0x0F & (* pPixel >> (x & 1 ? 0 : 4)) ;
        case        8:  return * pPixel ;
        case        16: return * (WORD *) pPixel ;
        case        24: return 0x00FFFFFF & * (DWORD *) pPixel ;
        case        32: return * (DWORD *) pPixel ;
        }
  return 0 ;
}

/*-------------------------------------------------------------------------
  DibSetPixel:  Sets a pixel value at (x, y)
-------------------------------------------------------------------------*/

BOOL DibSetPixel (HDIB hdib, int x, int y, DWORD dwPixel)
```

```
{
        PBYTE pPixel ;
        if (!(pPixel = DibPixelPtr (hdib, x, y)))
                return FALSE ;
        switch (DibBitCount (hdib))
   {
        case  1:     *     pPixel &= ~(1     << (7 - (x & 7))) ;
                     *     pPixel |= dwPixel     << (7 - (x & 7)) ;
                                   break ;
        case  4:  * pPixel &= 0x0F    << (x & 1 ? 4 : 0) ;
                  *     pPixel |= dwPixel     << (x & 1 ? 0 : 4) ;
                                   break ;
        case 8:      *     pPixel = (BYTE) dwPixel ;
                                   break ;

        case 16:     *     (WORD *) pPixel = (WORD) dwPixel ;
                                   break ;

        case 24:     *     (RGBTRIPLE *) pPixel = * (RGBTRIPLE *)
                                   break ;
```

```
        case 32:      *      (DWORD *) pPixel = dwPixel ;
                                        break ;
        default:
                    return FALSE ;
        }
        return TRUE ;
}

/*---------------------------------------------------------------------
  DibGetPixelColor:  Obtains the pixel color at (x, y)
---------------------------------------------------------------------*/

BOOL DibGetPixelColor (HDIB hdib, int x, int y, RGBQUAD * prgb
{
        DWORD                   dwPixel ;
        int                     iBitCount ;
        PDIBSTRUCT    pdib = hdib ;

                    // Get bit count; also use this as a validity check
```

```
      if (0 == (iBitCount = DibBitCount (hdib)))
                  return FALSE ;
                  // Get the pixel value
    dwPixel = DibGetPixel (hdib, x, y) ;
                  // If the bit-count is 8 or less, index the color tab
      if (iBitCount <= 8)
                  return DibGetColor (hdib, (int) dwPixel, prgb) ;
                  // If the bit-count is 24, just use the pixel
     else   if (iBitCount == 24)
{
           *     (RGBTRIPLE *) prgb = * (RGBTRIPLE *) & dwPixe
                 prgb->rgbReserved = 0 ;
}
           // If the bit-count is 32 and the biCompression field is
           //  just use the pixel
      else   if (iBitCount == 32 &&
                                  pdib->ds.dsBmih.biCompression
```

```
        {
             *      prgb = * (RGBQUAD *) & dwPixel ;
        }

             // Otherwise, use the mask and shift values
             //   (for best performance, don't use DibMask and Dib
        else
        {
        prgb->rgbRed = (BYTE)(((pdib->ds.dsBitfields[0] & dwPix
               >> pdib->iRShift[0]) << pdib->iLShift[0]) ;

        prgb->rgbGreen=(BYTE((pdib->ds.dsBitfields[1] & dwPixe
              >> pdib->iRShift[1]) << pdib->iLShift[1]) ;

        prgb->rgbBlue=(BYTE)(((pdib->ds.dsBitfields[2] & dwPixe
               >> pdib->iRShift[2]) << pdib->iLShift[2]) ;
        }
  return TRUE ;
}

/*---------------------------------------------------------------------
```

```
  DibSetPixelColor:  Sets the pixel color at (x, y)
------------------------------------------------------------------------------*/

BOOL DibSetPixelColor (HDIB hdib, int x, int y, RGBQUAD * prgb)
{
          DWORD                    dwPixel ;
          int                      iBitCount ;
          PDIBSTRUCT    pdib = hdib ;

               // Don't do this function for DIBs with color tables

          iBitCount = DibBitCount (hdib) ;
          if (iBitCount <= 8)
                    return FALSE ;
               // The rest is just the opposite of DibGetPixelColor
          else if (iBitCount == 24)
          {
                    * (RGBTRIPLE *) & dwPixel = * (RGBTRIPLE *) pr
                    dwPixel &= 0x00FFFFFF ;
          }
```

```
      else if (iBitCount == 32 &&
                                    pdib->ds.dsBmih.biCompression
{
                  * (RGBQUAD *) & dwPixel = * prgb ;
}

      else
{
   dwPixel  = (((DWORD)    prgb->rgbRed >> pdib->iLShift[0]
   << pdib->iRShift[0]) ;

   dwPixel |= (((DWORD)    prgb->rgbGreen >> pdib->iLShift[
   << pdib->iRShift[1]) ;

   dwPixel |= (((DWORD)    prgb->rgbBlue >> pdib->iLShift[2
   << pdib->iRShift[2]) ;
      }

      DibSetPixel (hdib, x, y, dwPixel) ;
     return TRUE ;
```

```
}
```

DIBHELP.CDibPixelPtrDIBSTRUCTppRowDIB

```
((PDIBSTRUCT) hdib)->pprow)[0]
```

DIB

```
(((PDIBSTRUCT) hdib)->ppRow)[y] + (x * DibBitCount (hdib) >>
```

(x,y)DIBxyDIBNULLDibPixelPtr

DibGetPixelDibSetPixelDibPixelPtr81632DIB14DIB

DibGetColorRGBQUAD148DIBDIB162432DIBRGB
DibSetPixelRGBQUAD162432DIB

16-21DIBHELPDIBDIBpacked

16-21 DIBHELP.C

```
DIBHELP.C

/*--------------------------------------------------------------------------

   Calculating shift values from color masks is required by the

   DibCreateFromInfo function.

--------------------------------------------------------------------------*/

static int MaskToRShift (DWORD dwMask)
```

```
{
        int iShift ;
        if (dwMask == 0)
                return 0 ;
        for (iShift = 0 ; !(dwMask & 1) ; iShift++)
                dwMask >>= 1 ;
        return iShift ;
}

static int MaskToLShift (DWORD dwMask)
{
        int iShift ;
        if (dwMask == 0)
                return 0 ;
        while (!(dwMask & 1))
                dwMask >>= 1 ;
        for (iShift = 0 ; dwMask & 1 ; iShift++)
                dwMask >>= 1 ;
        return 8 - iShift ;
```

```
}
/*--------------------------------------------------------------------------
   DibCreateFromInfo: All DIB creation functions ultimately call tl
   This function is responsible for calling CreateDIBSection, alloc
   memory for DIBSTRUCT, and setting up the row pointer.
--------------------------------------------------------------------------*/

HDIB DibCreateFromInfo (BITMAPINFO * pbmi)
{
        BYTE                    *       pBits ;
        DIBSTRUCT       *       pdib ;
        HBITMAP                 hBitmap ;
        int                     i, iRowLength, cy, y ;

        hBitmap = CreateDIBSection (NULL, pbmi, DIB_RGB_COL(
        if (hBitmap == NULL)
                return NULL ;
        if (NULL == (pdib = malloc (sizeof (DIBSTRUCT))))
        {
```

```
                DeleteObject (hBitmap) ;
                return NULL ;
}

     pdib->iSignature      = HDIB_SIGNATURE ;
     pdib->hBitmap         = hBitmap ;
     pdib->pBits           = pBits ;

     GetObject (hBitmap, sizeof (DIBSECTION), &pdib->ds) ;
          // Notice that we can now use the DIB information fu
          //   defined above.

          // If the compression is BI_BITFIELDS, calculate shifts

     if (DibCompression (pdib) == BI_BITFIELDS)
     {
          for (i = 0 ; i < 3 ; i++)
          {
                    pdib->iLShift[i] = MaskToLShift (pdib->ds.d
                    pdib->iRShift[i] = MaskToRShift (pdib->ds.
```

```
        }
    }

        // If the compression is BI_RGB, but bit-count is 16 or
        //   set the bitfields and the masks
    else if (DibCompression (pdib) == BI_RGB)
    {
        if (DibBitCount (pdib) == 16)
        {
                    pdib->ds.dsBitfields[0] = 0x00007C00 ;
                   pdib->ds.dsBitfields[1] = 0x000003E0 ;
                    pdib->ds.dsBitfields[2] = 0x0000001F ;

                    pdib->iRShift [0]    =          10    ;
                    pdib->iRShift [1]    =          5     ;
                    pdib->iRShift [2]    =          0     ;

                    pdib->iLShift [0]    =          3     ;
                    pdib->iLShift [1]    =          3     ;
                    pdib->iLShift [2]    =          3     ;
```

```
            }
            else if (DibBitCount (pdib) == 24 || DibBitCount (pdib
            {
                        pdib->ds.dsBitfields[0] = 0x00FF0000 ;
                        pdib->ds.dsBitfields[1] = 0x0000FF00 ;
                        pdib->ds.dsBitfields[2] = 0x000000FF ;

                        pdib->iRShift [0]    =          16    ;
                        pdib->iRShift [1]    =          8     ;
                        pdib->iRShift [2]    =          0     ;

                        pdib->iLShift [0]    =          0     ;
                        pdib->iLShift [1]    =          0     ;
                        pdib->iLShift [2]    =          0     ;
            }
}
            // Allocate an array of pointers to each row in the DIB
      cy = DibHeight (pdib) ;
      if (NULL == (pdib->ppRow = malloc (cy * sizeof (BYTE *))
      {
```

```
                free (pdib) ;
                DeleteObject (hBitmap) ;
                return NULL ;
        }

             // Initialize them.
        iRowLength = DibRowLength (pdib) ;
        if (pbmi->bmiHeader.biHeight > 0)                    // ie, l
        {
                for (y = 0 ; y < cy ; y++)
                                pdib->ppRow[y] = pBits + (cy - y
        }
        else
// top down
   {
                for (y = 0 ; y < cy ; y++)
                                pdib->ppRow[y] = pBits + y * iRow
        }
        return pdib ;
```

```
}

/*----------------------------------------------------------------------
  DibDelete:  Frees all memory for the DIB section
----------------------------------------------------------------------*/

BOOL DibDelete (HDIB hdib)
{
    DIBSTRUCT * pdib = hdib ;
          if (!DibIsValid (hdib))
                    return FALSE ;
          free (pdib->ppRow) ;
          DeleteObject (pdib->hBitmap) ;
          free (pdib) ;
          return TRUE ;
}

/*----------------------------------------------------------------------
  DibCreate: Creates an HDIB from explicit arguments
----------------------------------------------------------------------*/
```

```
HDIB DibCreate (int cx, int cy, int cBits, int cColors)
{
        BITMAPINFO    *      pbmi ;
         DWORD                          dwInfoSize ;
         HDIB                           hDib ;
         int                            cEntries ;

         if (cx <= 0 || cy <= 0 ||
                     ((cBits !=  1) && (cBits !=  4) && (cBits != 8) &&
                           (cBits != 16) && (cBits != 24) && (cBits !=
   {
               return NULL ;
  }

   if (   cColors != 0)
                     cEntries = cColors ;
  else   if (cBits <= 8)
                     cEntries = 1 << cBits ;
```

```
    dwInfoSize = sizeof (BITMAPINFOHEADER) + (cEntries - 1

    if (NULL == (pbmi = malloc (dwInfoSize)))
    {
         return NULL ;
    }

    ZeroMemory (pbmi, dwInfoSize) ;

    pbmi->bmiHeader.biSize                  = sizeof (BITMAPI
    pbmi->bmiHeader.biWidth                  = cx ;
    pbmi->bmiHeader.biHeight                 = cy ;
    pbmi->bmiHeader.biPlanes                 = 1 ;
    pbmi->bmiHeader.biBitCount                = cBits ;
    pbmi->bmiHeader.biCompression               = BI_RGB ;
    pbmi->bmiHeader.biSizeImage               = 0 ;
    pbmi->bmiHeader.biXPelsPerMeter             = 0 ;
    pbmi->bmiHeader.biYPelsPerMeter             = 0 ;
    pbmi->bmiHeader.biClrUsed                = cColors ;
    pbmi->bmiHeader.biClrImportant              = 0 ;
```

```
        hDib = DibCreateFromInfo (pbmi) ;
        free (pbmi) ;

        return hDib ;
}

/*-------------------------------------------------------------------------
   DibCopyToInfo:              Builds BITMAPINFO structure.
                                          Used by DibCopy and Di
-------------------------------------------------------------------------*/

static BITMAPINFO * DibCopyToInfo (HDIB hdib)
{
        BITMAPINFO          *      pbmi ;
        int                             i, iNumColors ;
        RGBQUAD                 *      prgb ;
        if (!DibIsValid (hdib))
                return NULL ;
                // Allocate the memory
        if (NULL == (pbmi = malloc (DibInfoSize (hdib))))
```

```
                return NULL ;

                // Copy the information header

        CopyMemory (pbmi, DibInfoHeaderPtr   (hdib), sizeof (BIT


                // Copy the possible color masks

        prgb = (RGBQUAD *) ((BYTE *) pbmi + sizeof (BITMAPINF

        if (DibMaskSize (hdib))

        {

                CopyMemory (prgb, DibMaskPtr (hdib), 3 * sizeo

                prgb = (RGBQUAD *) ((BYTE *) prgb + 3 * sizeof

  }

                // Copy the color table

        iNumColors = DibNumColors (hdib) ;

        for (i = 0 ; i < iNumColors ; i++)

                DibGetColor (hdib, i, prgb + i) ;

        return pbmi ;

}
```

```
/*----------------------------------------------------------------------
  DibCopy:  Creates a new DIB section from an existing DIB secti
            possibly swapping the DIB width and height.
----------------------------------------------------------------------*/

HDIB DibCopy (HDIB hdibSrc, BOOL fRotate)
{
        BITMAPINFO          *      pbmi ;
        BYTE                *      pBitsSrc, * pBitsDst ;
        HDIB        hdibDst ;

        if (!DibIsValid (hdibSrc))
                return NULL ;
        if (NULL == (pbmi = DibCopyToInfo (hdibSrc)))
                return NULL ;
        if (fRotate)
        {
                pbmi->bmiHeader.biWidth = DibHeight (hdibSrc
                pbmi->bmiHeader.biHeight = DibWidth (hdibSrc
        }
```

```
        hdibDst = DibCreateFromInfo (pbmi) ;
        free (pbmi) ;

        if (   hdibDst == NULL)
                return NULL ;

                        // Copy the bits

        if (!fRotate)
        {
                        pBitsSrc = DibBitsPtr (hdibSrc) ;
                        pBitsDst = DibBitsPtr (hdibDst) ;

                       CopyMemory (pBitsDst, pBitsSrc, DibBitsSiz
        }
   return hdibDst ;
}

/*--------------------------------------------------------------------------
  DibCopyToPackedDib is generally used for saving DIBs and for
```

```
  transferring DIBs to the clipboard. In the second case, the secc
  argument should be set to TRUE so that the memory is allocate
  with the GMEM_SHARE flag.
----------------------------------------------------------------------*/

BITMAPINFO * DibCopyToPackedDib (HDIB hdib, BOOL fUseGloba
{
        BITMAPINFO          *      pPackedDib ;
        BYTE                *      pBits ;
        DWORD                      dwDibSize ;
        HDC                        hdcMem ;
        HGLOBAL                    hGlobal ;
        int                        iNumColors ;
        PDIBSTRUCT                 pdib = hdib ;
        RGBQUAD                    *      prgb ;

        if (!DibIsValid (hdib))
                return NULL ;
                        // Allocate memory for packed DIB
        dwDibSize = DibTotalSize (hdib) ;
```

```
        if (fUseGlobal)
        {
                hGlobal = GlobalAlloc (GHND | GMEM_SHARE, d
                pPackedDib = GlobalLock (hGlobal) ;
        }
        else
{
                pPackedDib = malloc (dwDibSize) ;
        }

        if (pPackedDib == NULL)
                return NULL ;
                // Copy the information header
        CopyMemory (pPackedDib, &pdib->ds.dsBmih, sizeof (BIT
        prgb = (RGBQUAD *) ((BYTE *) pPackedDib + sizeof (BITM
                // Copy the possible color masks
        if (pdib->ds.dsBmih.biCompression == BI_BITFIELDS)
        {
             CopyMemory (prgb, pdib->ds.dsBitfields, 3 * sizeof (
```

```
        prgb = (RGBQUAD *) ((BYTE *) prgb + 3 * sizeof (DW
    }
                // Copy the color table
    if (iNumColors = DibNumColors (hdib))
    {
              hdcMem = CreateCompatibleDC (NULL) ;
                SelectObject (hdcMem, pdib->hBitmap) ;
                GetDIBColorTable (hdcMem, 0, iNumColors, prgb
              DeleteDC (hdcMem) ;
    }

    pBits = (BYTE *) (prgb + iNumColors) ;
                // Copy the bits
    CopyMemory (pBits, pdib->pBits, DibBitsSize (pdib)) ;
                // If last argument is TRUE, unlock global memo
               //   cast it to pointer in preparation for return

    if (fUseGlobal)
    {
```

```
                    GlobalUnlock (hGlobal) ;
                    pPackedDib = (BITMAPINFO *) hGlobal ;
          }
    return pPackedDib ;
}

/*-------------------------------------------------------------------------
  DibCopyFromPackedDib is generally used for pasting DIBs from
    clipboard.
-------------------------------------------------------------------------*/

HDIB DibCopyFromPackedDib (BITMAPINFO * pPackedDib)
{
          BYTE                    *      pBits ;
          DWORD                          dwInfoSize, dwMaskSize, dwC
          int                          iBitCount ;
          PDIBSTRUCT           pdib ;

                    // Get the size of the information header and do
```

```
        dwInfoSize = pPackedDib->bmiHeader.biSize ;
        if (   dwInfoSize != sizeof (BITMAPCOREHEADER) &&
                    dwInfoSize != sizeof (BITMAPINFOHEADER) &&
                    dwInfoSize != sizeof (BITMAPV4HEADER) &&
                    dwInfoSize != sizeof (BITMAPV5HEADER))
        {
                    return NULL ;
        }
                    // Get the possible size of the color masks
        if (dwInfoSize == sizeof (BITMAPINFOHEADER) &&
                    pPackedDib->bmiHeader.biCompression == BI_
        {
                    dwMaskSize = 3 * sizeof (DWORD) ;
        }
        else
        {
                    dwMaskSize = 0 ;
        }
```

```
                // Get the size of the color table
        if (dwInfoSize == sizeof (BITMAPCOREHEADER))
        {
                iBitCount = ((BITMAPCOREHEADER *) pPackedD

               if (iBitCount <= 8)
                {
                dwColorSize = (1 << iBitCount) * sizeof (RGBTR
                }
                else
                                dwColorSize = 0 ;
}
        else                            // all non-OS/2 compatible DIBs
        {
               if (pPackedDib->bmiHeader.biClrUsed > 0)
                {
          dwColorSize = pPackedDib->bmiHeader.biClrUsed * size
                }
                else if (pPackedDib->bmiHeader.biBitCount <=
```

```
                {
                        dwColorSize = (1 << pPackedDib
                }
                else
                {
                        dwColorSize = 0 ;
                }
        }
                // Finally, get the pointer to the bits in the packe
        pBits = (BYTE *) pPackedDib + dwInfoSize + dwMaskSize
                // Create the HDIB from the packed-DIB pointer
        pdib = DibCreateFromInfo (pPackedDib) ;
                // Copy the pixel bits
        CopyMemory (pdib->pBits, pBits, DibBitsSize (pdib)) ;
        return pdib ;
}

/*---------------------------------------------------------------------
  DibFileLoad:  Creates a DIB section from a DIB file
```

```
-----------------------------------------------------------------------*/

HDIB DibFileLoad (const TCHAR * szFileName)

{

        BITMAPFILEHEADER        bmfh ;

        BITMAPINFO              *    pbmi ;

        BOOL                                    bSuccess ;

        DWORD                                   dwInfoSize, dw

        HANDLE                                  hFile ;

        HDIB                                    hDib ;

         // Open the file: read access, prohibit write access

        hFile = CreateFile (szFileName, GENERIC_READ, FILE_SHA

                    OPEN_EXISTING, FILE_FLAG_SEQUENTIAL_SCAN

       if (hFile == INVALID_HANDLE_VALUE)

                    return NULL ;

                    // Read in the BITMAPFILEHEADER

        bSuccess = ReadFile ( hFile, &bmfh, sizeof (BITMAPFILEHE

                              &dwBytesRead, NULL) ;
```

```
    if (!bSuccess || (dwBytesRead != sizeof (BITMAPFILEHEAD
                                        || (bmfh.bfType != * (WORD
    {
                CloseHandle (hFile) ;
                return NULL ;
}
                // Allocate memory for the information structure
    dwInfoSize = bmfh.bfOffBits - sizeof (BITMAPFILEHEADER
    if (NULL == (pbmi = malloc (dwInfoSize)))
    {
                CloseHandle (hFile) ;
                return NULL ;
    }

    bSuccess = ReadFile (hFile, pbmi, dwInfoSize, &dwBytesR
    if (!bSuccess || (dwBytesRead != dwInfoSize))
{
          CloseHandle (hFile) ;
          free (pbmi) ;
```

```
        return NULL ;
    }
        // Create the DIB
    hDib = DibCreateFromInfo (pbmi) ;
    free (pbmi) ;

    if (hDib == NULL)
    {
        CloseHandle (hFile) ;
        return NULL ;
    }
        // Read in the bits
    dwBitsSize = bmfh.bfSize - bmfh.bfOffBits ;
    bSuccess = ReadFile ( hFile, ((PDIBSTRUCT) hDib)->pBits,
                    dwBitsSize, &dwBytesRead, NULL) ;
    CloseHandle (hFile) ;
    if (!bSuccess || (dwBytesRead != dwBitsSize))
    {
        DibDelete (hDib) ;
```

```
            return NULL ;
        }
        return hDib ;
}

/*-------------------------------------------------------------------------
  DibFileSave:  Saves a DIB section to a file
-------------------------------------------------------------------------*/

BOOL DibFileSave (HDIB hdib, const TCHAR * szFileName)
{
        BITMAPFILEHEADER            bmfh ;
        BITMAPINFO                *     pbmi ;
        BOOL                      bSuccess ;
        DWORD                     dwTotalSize, dwBytesWritten ;
        HANDLE                    hFile ;

        hFile = CreateFile (szFileName, GENERIC_WRITE, 0, NULL,
                            CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL
        if (hFile == INVALID_HANDLE_VALUE)
```

```
            return FALSE ;

    dwTotalSize                 = DibTotalSize (hdib) ;

    bmfh.bfType                  = * (WORD *) "BM" ;

    bmfh.bfSize                 = sizeof (BITMAPFILEHEADER)

    bmfh.bfReserved1            = 0 ;

    bmfh.bfReserved2            = 0 ;

    bmfh.bfOffBits              = bmfh.bfSize - DibBitsSize (hdib

                // Write the BITMAPFILEHEADER

    bSuccess = WriteFile (hFile, &bmfh, sizeof (BITMAPFILEHE

                    &dwBytesWritten, NULL) ;

    if (!bSuccess || (dwBytesWritten != sizeof (BITMAPFILEHE

    {

                CloseHandle (hFile) ;

                DeleteFile (szFileName) ;

                return FALSE ;

    }

                // Get entire DIB in packed-DIB format
```

```
        if (NULL == (pbmi = DibCopyToPackedDib (hdib, FALSE)))
   {
                CloseHandle (hFile) ;
                DeleteFile (szFileName) ;
                return FALSE ;
        }
               // Write out the packed DIB
        bSuccess = WriteFile (hFile, pbmi, dwTotalSize, &dwBytes
        CloseHandle (hFile) ;
        free (pbmi) ;

        if (!bSuccess || (dwBytesWritten != dwTotalSize))
        {
                DeleteFile (szFileName) ;
                return FALSE ;
        }
        return TRUE ;
}

/*---------------------------------------------------------------------
```

```
  DibCopyToDdb:  For more efficient screen displays

-----------------------------------------------------------------------*/

HBITMAP DibCopyToDdb (HDIB hdib, HWND hwnd, HPALETTE hPa

{

        BITMAPINFO          *      pbmi ;

        HBITMAP                         hBitmap ;

        HDC                                 hdc ;

        if (!DibIsValid (hdib))

                return NULL ;

        if (NULL == (pbmi = DibCopyToInfo (hdib)))

                return NULL ;

        hdc = GetDC (hwnd) ;

        if (hPalette)

        {

                SelectPalette (hdc, hPalette, FALSE) ;

                RealizePalette (hdc) ;

        }
```

```
        hBitmap = CreateDIBitmap (hdc, DibInfoHeaderPtr (hdib)
                                                DibBitsPtr (hdib), pbm

        ReleaseDC (hwnd, hdc) ;

        free (pbmi) ;

        return hBitmap ;
}
```

DIBHELP.C1632DIB

DibCreateFromInfoDIBHELPCreateDIBSectionDIBSTRUCT
DibCreateFromInfoBITMAPINFOCreateDIBSection
DIBSTRUCTDIBSTRUCTppRowDIBDIBppRowDIB

DibDeleteDibCreateFromInfo

DibCreateDibCreateFromInfo00

DibCopyDIBDIBDibCreateInfoBITMAPINFODibCopyBOOLDIB
DIB

DibCopyToPackedDibDibCopyFromPackedDibDIBDibFileLoadDIBDIB
DibFileSaveDIB

DibCopyToDdbDIBGDICreateDIBitmapSHOWDIB7

DIBHELP

DIBHELP.H16-22

```
DIBHELP.H

/*----------------------------------------------------------------------
  DIBHELP.H header file for DIBHELP.C
----------------------------------------------------------------------*/

typedef void * HDIB ;

          // Functions in DIBHELP.C

BOOL DibIsValid (HDIB hdib) ;

HBITMAP DibBitmapHandle (HDIB hdib) ;

int DibWidth (HDIB hdib) ;

int DibHeight (HDIB hdib) ;

int DibBitCount (HDIB hdib) ;

int DibRowLength (HDIB hdib) ;

int DibNumColors (HDIB hdib) ;

DWORD DibMask (HDIB hdib, int i) ;

int DibRShift (HDIB hdib, int i) ;

int DibLShift (HDIB hdib, int i) ;

int DibCompression (HDIB hdib) ;
```

```
BOOL DibIsAddressable (HDIB hdib) ;

DWORD DibInfoHeaderSize (HDIB hdib) ;

DWORD DibMaskSize (HDIB hdib) ;

DWORD DibColorSize (HDIB hdib) ;

DWORD DibInfoSize (HDIB hdib) ;

DWORD DibBitsSize (HDIB hdib) ;

DWORD DibTotalSize (HDIB hdib) ;

BITMAPINFOHEADER * DibInfoHeaderPtr (HDIB hdib) ;

DWORD * DibMaskPtr (HDIB hdib) ;

void * DibBitsPtr (HDIB hdib) ;

BOOL DibGetColor (HDIB hdib, int index, RGBQUAD * prgb) ;

BOOL DibSetColor (HDIB hdib, int index, RGBQUAD * prgb) ;

BYTE * DibPixelPtr (HDIB hdib, int x, int y) ;

DWORD DibGetPixel (HDIB hdib, int x, int y) ;

BOOL DibSetPixel (HDIB hdib, int x, int y, DWORD dwPixel) ;

BOOL DibGetPixelColor (HDIB hdib, int x, int y, RGBQUAD * prgb

BOOL DibSetPixelColor (HDIB hdib, int x, int y, RGBQUAD * prgb)

HDIB DibCreateFromInfo (BITMAPINFO * pbmi) ;

BOOL DibDelete (HDIB hdib) ;
```

```
HDIB DibCreate (int cx, int cy, int cBits, int cColors) ;
HDIB DibCopy (HDIB hdibSrc, BOOL fRotate) ;
BITMAPINFO * DibCopyToPackedDib (HDIB hdib, BOOL fUseGloba
HDIB DibCopyFromPackedDib (BITMAPINFO * pPackedDib) ;
HDIB DibFileLoad (const TCHAR * szFileName) ;
BOOL DibFileSave (HDIB hdib, const TCHAR * szFileName) ;
HBITMAP DibCopyToDdb (HDIB hdib, HWND hwnd, HPALETTE hPa
HDIB DibCreateFromDdb (HBITMAP hBitmap) ;

/*----------------------------------------------------------------------
  Quickie no-bounds-checked pixel gets and sets
----------------------------------------------------------------------*/

#define     DibPixelPtr1(hdib, x, y)      (((* (PBYTE **) hdib) [y]) +
#define     DibPixelPtr4(hdib, x, y)      (((* (PBYTE **) hdib) [y]) +
#define     DibPixelPtr8(hdib, x, y)      (((* (PBYTE **) hdib) [y]) +
#define     DibPixelPtr16(hdib, x, y)  \
                                    ((WORD *) (((* (PBYTE **) hdib) [y

#define DibPixelPtr24(hdib, x, y)  \
```

```
                                        ((RGBTRIPLE *) (((* (PBYTE *
#define DibPixelPtr32(hdib, x, y)  \
                                            ((DWORD *) (((* (PBYTE

#define DibGetPixel1(hdib, x, y)   \
                                (0x01 & (* DibPixelPtr1 (hdib, x, y) >>

#define DibGetPixel4(hdib, x, y)   \
                                (0x0F & (* DibPixelPtr4 (hdib, x, y) >>

#define    DibGetPixel8(hdib, x, y)                (* DibPixelPtr8
#define    DibGetPixel16(hdib, x, y)    (* DibPixelPtr16     (hdib,
#define    DibGetPixel24(hdib, x, y)    (* DibPixelPtr24     (hdib,
#define    DibGetPixel32(hdib, x, y)    (* DibPixelPtr32     (hdib,

#define DibSetPixel1(hdib, x, y, p)
            ((* DibPixelPtr1 (hdib, x, y) &= ~( 1<< (7 - ((x) & 7))
            (* DibPixelPtr1 (hdib, x, y) |=     ((p) << (7 - ((x) & 7)

#define DibSetPixel4(hdib, x, y, p)
            ((* DibPixelPtr4 (hdib, x, y) &= (0x0F <<    ((x) & 1 ?
```

```
            (* DibPixelPtr4 (hdib, x, y) |= ((p) <<     ((x) & 1 ? 0 :

#define DibSetPixel8(hdib, x, y, p)  (* DibPixelPtr8 (hdib, x, y) =

#define DibSetPixel16(hdib, x, y, p) (* DibPixelPtr16 (hdib, x, y)

#define DibSetPixel24(hdib, x, y, p) (* DibPixelPtr24 (hdib, x, y)

#define DibSetPixel32(hdib, x, y, p) (* DibPixelPtr32 (hdib, x, y)
```

HDIB(void*)HDIBDIBHELP.C

DIBHELP.CDibPixelPtrDibGetPixelDibSetPixel
DibBitCountDIBSTRUCT

DIBDibGetPixel1DibGetPixel4DibGetPixel8inline

DIBHELP.HDibPixelPtrDibGetPixelDibSetPixel

DIBBLE

DIBBLE16-23DIBHELPDIBBLEDIBBLEMDI
multiple document interface

16-23 DIBBLE

```
DIBBLE.C

/*------------------------------------------------------------------------

  DIBBLE.C --   Bitmap and Palette Program

                                        (c) Charles Petzold, 1998
```

```
------------------------------------------------------------------------*/

#include <windows.h>
#include "dibhelp.h"
#include "dibpal.h"
#include "dibconv.h"
#include "resource.h"

#define WM_USER_SETSCROLLS             (WM_USER + 1)
#define WM_USER_DELETEDIB              (WM_USER + 2)
#define WM_USER_DELETEPAL              (WM_USER + 3)
#define WM_USER_CREATEPAL              (WM_USER + 4)

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
TCHAR szAppName[] = TEXT ("Dibble") ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                                PSTR szCmdLine, int iCr
{
        HACCEL             hAccel ;
         HWND               hwnd ;
         MSG                msg ;
```

```
WNDCLASS            wndclass ;

wndclass.style                      = CS_HREDRAW | CS_V
wndclass.lpfnWndProc                    = WndProc ;
wndclass.cbClsExtra                     = 0 ;
wndclass.cbWndExtra                     = 0 ;
wndclass.hInstance                      = hInstance ;
wndclass.hIcon                        = LoadIcon (NULL, IDI_
wndclass.hCursor                        = LoadCursor (NULL,
wndclass.hbrBackground            = (HBRUSH) GetStockO
wndclass.lpszMenuName             = szAppName ;
wndclass.lpszClassName      = szAppName ;

if (!RegisterClass (&wndclass))
{
          MessageBox (  NULL, TEXT ("This program requi
                                          szAppName, M

          return 0 ;
}
```

```
hwnd   =      CreateWindow (szAppName, szAppName,
    WS_OVERLAPPEDWINDOW | WM_VSCROLL | WM_HSCR
   CW_USEDEFAULT, CW_USEDEFAULT,
   CW_USEDEFAULT, CW_USEDEFAULT,
    NULL, NULL, hInstance, NULL) ;

ShowWindow (hwnd, iCmdShow) ;
UpdateWindow (hwnd) ;

hAccel = LoadAccelerators (hInstance, szAppName) ;

while (GetMessage (&msg, NULL, 0, 0))
{
      if (!TranslateAccelerator (hwnd, hAccel, &msg))
      {
                  TranslateMessage (&msg) ;
                  DispatchMessage (&msg) ;
      }
}
return msg.wParam ;
```

```
}

/*----------------------------------------------------------------------
  DisplayDib:          Displays or prints DIB actual size or stretche
                                          depending on menu selectio
----------------------------------------------------------------------*/

int DisplayDib (   HDC hdc, HBITMAP hBitmap, int x, int y,
                                          int cxClient, int cyClient,
                                          WORD wShow, BOOL fHalfto
{
          BITMAP       bitmap ;
          HDC                hdcMem ;
          int                cxBitmap, cyBitmap, iReturn ;
          GetObject    (hBitmap, sizeof (BITMAP), &bitmap) ;
          cxBitmap           =     bitmap.bmWidth ;
          cyBitmap           =     bitmap.bmHeight ;

          SaveDC (hdc) ;
          if (fHalftonePalette)
```

```
                SetStretchBltMode (hdc, HALFTONE) ;
        else
                SetStretchBltMode (hdc, COLORONCOLOR) ;
          hdcMem = CreateCompatibleDC (hdc) ;
          SelectObject (hdcMem, hBitmap) ;

          switch (wShow)
          {
          case   IDM_SHOW_NORMAL:
                if (fHalftonePalette)
                        iReturn = StretchBlt (hdc,    0, 0, min (cxCl
                         min (cyClient, cyBitmap - y),
               hdcMem, x, y, min (cxClient, cxBitmap - x),
                        min (cyClient, cyBitmap - y),
               SRCCOPY);
                else
                        iReturn = BitBlt (hdc,0, 0, min (cxClient, cx
                   min (cyClient, cyBitmap - y),
hdcMem, x, y, SRCCOPY) ;
```

```
                    break ;


          case   IDM_SHOW_CENTER:
                    if (fHalftonePalette)
                    iReturn = StretchBlt (hdc, (cxClient - cxBitmap)
                                        (cyClient - cyBitmap) / 2,
cxBitmap, cyBitmap,
                      hdcMem, 0, 0, cxBitmap, cyBitmap, SRCCOPY)
                    else
                                iReturn = BitBlt (hdc,       (cxClient - c
                    cyClient - cyBitmap) / 2,
                     cxBitmap, cyBitmap,
                     hdcMem, 0, 0, SRCCOPY) ;
                    break ;
          case   IDM_SHOW_STRETCH:
                    iReturn = StretchBlt (hdc,    0, 0, cxClient, cyClie
                hdcMem,       0, 0, cxBitmap, cyBitmap, SRCCOPY) ;
                    break ;
```

```
        case   IDM_SHOW_ISOSTRETCH:
                   SetMapMode (hdc, MM_ISOTROPIC) ;
                    SetWindowExtEx (hdc, cxBitmap, cyBitmap, NUI
                    SetViewportExtEx (hdc, cxClient, cyClient, NULL
                    SetWindowOrgEx (hdc, cxBitmap / 2, cyBitmap /
                    SetViewportOrgEx (hdc, cxClient / 2, cyClient / 2

                    iReturn = StretchBlt (hdc,    0, 0, cxBitmap, cyB
              hdcMem,0, 0, cxBitmap, cyBitmap, SRCCOPY) ;
                    break ;
        }
   DeleteDC (hdcMem) ;
   RestoreDC (hdc, -1) ;
   return iReturn ;
}

/*------------------------------------------------------------------------
  DibFlipHorizontal: Calls non-optimized DibSetPixel and DibGetP
------------------------------------------------------------------------*/
```

```
HDIB DibFlipHorizontal (HDIB hdibSrc)
{
        HDIB hdibDst ;
        int  cx, cy, x, y ;

        if (!DibIsAddressable (hdibSrc))
                return NULL ;
       if (NULL == (hdibDst = DibCopy (hdibSrc, FALSE)))
                return NULL ;
        cx = DibWidth  (hdibSrc) ;
        cy = DibHeight (hdibSrc) ;

   for (x = 0 ; x < cx ; x++)
        for (y = 0 ; y < cy ; y++)
        {
              DibSetPixel (hdibDst, x, cy - 1 - y, DibGetPixel (hdibS
        }

        return hdibDst ;
```

```
}

/*----------------------------------------------------------------------------
  DibRotateRight: Calls optimized DibSetPixelx and DibGetPixelx
----------------------------------------------------------------------------*/

HDIB DibRotateRight (HDIB hdibSrc)
{
         HDIB   hdibDst ;
         int    cx, cy, x, y ;

         if (!DibIsAddressable (hdibSrc))
                  return NULL ;

         if (NULL == (hdibDst = DibCopy (hdibSrc, TRUE)))
                  return NULL ;

        cx = DibWidth (hdibSrc) ;
         cy = DibHeight (hdibSrc) ;

         switch (DibBitCount (hdibSrc))
   {
```

```
        case       1:
                       for (  x = 0 ; x < cx ; x++)
                       for (  y = 0 ; y < cy ; y++)
                                          DibSetPixel1 (hdibDst, c
                                          DibGetPixel1 (hdibSrc, x,
                       break ;

        case       4:
                       for (  x = 0 ; x < cx ; x++)
                       for (  y = 0 ; y < cy ; y++)
                                          DibSetPixel4 (hdibDst, c
                                          DibGetPixel4 (hdibSrc, x,
                      break ;

        case       8:
                       for (  x = 0 ; x < cx ; x++)
                       for (  y = 0 ; y < cy ; y++)
                                          DibSetPixel8 (hdibDst, c
                                          DibGetPixel8 (hdibSrc, x,
```

```
                    break ;

        case        16:

                    for (x = 0 ; x < cx ; x++)

                    for (y = 0 ; y < cy ; y++)

                                DibSetPixel16 (hdibDst, cy -

                                    DibGetPixel16 (hdibSrc,

                    break ;

        case        24:

                    for (x = 0 ; x < cx ; x++)

                    for (y = 0 ; y < cy ; y++)

                                DibSetPixel24 (hdibDst, cy -

                                    DibGetPixel24 (hdibSrc,

                    break ;

        case        32:

                    for (x = 0 ; x < cx ; x++)

                    for (y = 0 ; y < cy ; y++)

                                DibSetPixel32 (hdibDst, cy -

                                    DibGetPixel32 (hdibSrc,
```

```
                        break ;
        }
        return hdibDst ;
}

/*----------------------------------------------------------------------
  PaletteMenu: Uncheck and check menu item on palette menu
----------------------------------------------------------------------*/

void PaletteMenu (HMENU hMenu, WORD wItemNew)
{
        static WORD wItem = IDM_PAL_NONE ;
        CheckMenuItem (hMenu, wItem, MF_UNCHECKED) ;
        wItem = wItemNew ;
        CheckMenuItem (hMenu, wItem, MF_CHECKED) ;
}

LRESULT CALLBACK WndProc ( HWND hwnd, UINT message, WPA
{
        static BOOL                     fHalftonePalette ;
```

```
    static       DOCINFO                  di = {sizeof(DOCINFO),
    static HBITMAP            hBitmap ;
    static HDIB                        hdib ;
    static HMENU                       hMenu ;
    static HPALETTE                  hPalette ;
    static int                         cxClient, cyClient, iVscroll,
    static OPENFILENAME ofn ;
    static PRINTDLG                  printdlg = { sizeof (PRINTD
    static TCHAR                       szFileName [MAX_PATH], s
   static TCHAR                        szFilter[]=   TEXT ("Bitma
      TEXT ("All Files (*.*)\0*.*\0\0") ;
    static TCHAR               *          szCompression[] = {
              TEXT("BI_RGB"),TEXT("BI_RLE8"),TEXT("BI_RLE4"
TEXT("BI_BITFIELDS"),TEXT("Unknown")} ;
    static WORD         wShow = IDM_SHOW_NORMAL ;
    BOOL               fSuccess ;
    BYTE                    *          pGlobal ;
    HDC               hdc, hdcPrn ;
```

```
HGLOBAL             hGlobal ;
HDIB                hdibNew ;
int                 iEnable, cxPage, cyPage, iConvert ;
PAINTSTRUCT         ps ;
SCROLLINFO          si ;
TCHAR               szBuffer [256] ;

switch (message)
{
case   WM_CREATE:

                    // Save the menu handle in a static var

          hMenu = GetMenu (hwnd) ;

          // Initialize the OPENFILENAME structure for the
          //   and File Save dialog boxes.

          ofn.lStructSize       = sizeof (OPENFILENAME) ;
          ofn.hwndOwner         = hwnd ;
          ofn.hInstance         = NULL ;
```

```
ofn.lpstrFilter       = szFilter ;
ofn.lpstrCustomFilter = NULL ;
ofn.nMaxCustFilter    = 0 ;
ofn.nFilterIndex      = 0 ;
ofn.lpstrFile         = szFileName ;
ofn.nMaxFile          = MAX_PATH ;
ofn.lpstrFileTitle    = szTitleName ;
ofn.nMaxFileTitle     = MAX_PATH ;
ofn.lpstrInitialDir   = NULL ;
ofn.lpstrTitle        = NULL ;
ofn.Flags             = OFN_OVERWRITEPROMPT ;
ofn.nFileOffset       = 0 ;
ofn.nFileExtension    = 0 ;
ofn.lpstrDefExt       = TEXT ("bmp") ;
ofn.lCustData         = 0 ;
ofn.lpfnHook          = NULL ;
ofn.lpTemplateName    = NULL ;
return 0 ;
```

```
        case   WM_DISPLAYCHANGE:
                   SendMessage (hwnd, WM_USER_DELETEPAL, 0,
                   SendMessage (hwnd, WM_USER_CREATEPAL, TR
                   return 0 ;

         case   WM_SIZE:
                         // Save the client area width and height in
                   cxClient = LOWORD (lParam) ;
                   cyClient = HIWORD (lParam) ;

                   wParam = FALSE ;
                  // fall through
                         // WM_USER_SETSCROLLS:  Programmer-de
                        // Set the scroll bars. If the display mode is
                         //   make them invisible. If wParam is TRUE
                         //   scroll bar position.

         case   WM_USER_SETSCROLLS:
                   if (hdib == NULL || wShow != IDM_SHOW_NORM
```

```
                {
                        si.cbSize           = sizeof (SCROL
                        si.fMask            = SIF_RANGE ;
                       si.nMin             = 0 ;
                        si.nMax             = 0 ;
                        SetScrollInfo (hwnd, SB_VERT, &s
                        SetScrollInfo (hwnd, SB_HORZ, &s
                }
                else
                {
                       // First the vertical scroll

                        si.cbSize           = sizeof (SCROL
                        si.fMask                = SIF_ALL ;

                    GetScrollInfo (hwnd, SB_VERT, &si) ;
                        si.nMin                 = 0 ;
                        si.nMax                 = DibHeight
                        si.nPage                = cyClient
                        if ((BOOL) wParam)
```

```
                si.nPos = 0 ;

            SetScrollInfo (hwnd, SB_VERT, &s
            GetScrollInfo (hwnd, SB_VERT, &s

            iVscroll = si.nPos ;

                // Then the horizontal scroll

            GetScrollInfo (hwnd, SB_HORZ, &
            si.nMin             = 0 ;
            si.nMax              = DibWidth (hdi
           si.nPage             = cxClient ;

            if ((BOOL) wParam)
                si.nPos = 0 ;

            SetScrollInfo (hwnd, SB_HORZ, &s
            GetScrollInfo (hwnd, SB_HORZ, &

            iHscroll = si.nPos ;
```

```
                }
                return 0 ;

                // WM_VSCROLL: Vertically scroll the DIB

        case   WM_VSCROLL:
                si.cbSize = sizeof (SCROLLINFO) ;
                si.fMask  = SIF_ALL ;
                GetScrollInfo (hwnd, SB_VERT, &si) ;

                iVscroll = si.nPos ;

                switch (LOWORD (wParam))
                {
                case SB_LINEUP:    si.nPos -  = 1 ;        break ;
                case SB_LINEDOWN:  si.nPos +  = 1 ;        break
                case SB_PAGEUP:    si.nPos -  = si.nPage ;break
               case SB_PAGEDOWN:  si.nPos +  = si.nPage ;bre
                case SB_THUMBTRACK:si.nPos          = si.nTrack
                default:
```

```
                }
                si.fMask = SIF_POS ;
                SetScrollInfo (hwnd, SB_VERT, &si, TRUE) ;
                GetScrollInfo (hwnd, SB_VERT, &si) ;
                if (si.nPos != iVscroll)
                {
                        ScrollWindow (hwnd, 0, iVscroll - si.nPo
                         iVscroll = si.nPos ;
                        UpdateWindow (hwnd) ;
                }
                return 0 ;

                // WM_HSCROLL: Horizontally scroll the DIB

        case   WM_HSCROLL:
                si.cbSize = sizeof (SCROLLINFO) ;
                si.fMask  = SIF_ALL ;
                GetScrollInfo (hwnd, SB_HORZ, &si) ;

               iHscroll = si.nPos ;
```

```
switch (LOWORD (wParam))
{
case SB_LINELEFT:  si.nPos -=1 ;     break ;
case SB_LINERIGHT: si.nPos +=1 ;     break ;
case SB_PAGELEFT:  si.nPos -=si.nPage ;break ;
case SB_PAGERIGHT: si.nPos +=si.nPage ;break ;
case SB_THUMBTRACK:si.nPos  =si.nTrackPos ;b
default:                                    brea
}

si.fMask = SIF_POS ;
SetScrollInfo (hwnd, SB_HORZ, &si, TRUE) ;
GetScrollInfo (hwnd, SB_HORZ, &si) ;

if (si.nPos != iHscroll)
{
        ScrollWindow (hwnd, iHscroll - si.nPos,
        iHscroll = si.nPos ;
         UpdateWindow (hwnd) ;
```

```
                }
                return 0 ;

                // WM_INITMENUPOPUP:  Enable or Gray menu it

case   WM_INITMENUPOPUP:
                if (hdib)
                                iEnable = MF_ENABLED ;
                else
                                iEnable = MF_GRAYED ;
  EnableMenuItem (hMenu, IDM_FILE_SAVE,                 iEnab
  EnableMenuItem (hMenu, IDM_FILE_PRINT,                iEnab
  EnableMenuItem (hMenu, IDM_FILE_PROPERTIES,             iE
  EnableMenuItem (hMenu, IDM_EDIT_CUT,                  iEnab
  EnableMenuItem (hMenu, IDM_EDIT_COPY,                  iEna
  EnableMenuItem (hMenu, IDM_EDIT_DELETE,                 iEna

                if (DibIsAddressable (hdib))
                               iEnable = MF_ENABLED ;
                else
```

```
                                iEnable = MF_GRAYED ;

EnableMenuItem (hMenu, IDM_EDIT_ROTATE,        iEnable)
 EnableMenuItem (hMenu, IDM_EDIT_FLIP,          iEnable) ;
 EnableMenuItem (hMenu, IDM_CONVERT_01,         iEnable
 EnableMenuItem (hMenu, IDM_CONVERT_04,         iEnable
EnableMenuItem (hMenu, IDM_CONVERT_08,          iEnable)
 EnableMenuItem (hMenu, IDM_CONVERT_16,         iEnable
 EnableMenuItem (hMenu, IDM_CONVERT_24,         iEnable
 EnableMenuItem (hMenu, IDM_CONVERT_32,         iEnable

                switch (DibBitCount (hdib))
                 {
                 case  1:     EnableMenuItem (hMenu, IDM_CONV
                case  4:     EnableMenuItem (hMenu, IDM_CONVE
                 case  8:     EnableMenuItem (hMenu, IDM_CONV
                 case   16: EnableMenuItem (hMenu, IDM_CONVE
                case   24: EnableMenuItem (hMenu, IDM_CONVE
                 case   32: EnableMenuItem (hMenu, IDM_CONVE
```

```
                }

                if (hdib && DibColorSize (hdib) > 0)
                                iEnable = MF_ENABLED ;
                else
                                iEnable = MF_GRAYED ;

                EnableMenuItem (hMenu, IDM_PAL_DIBTABLE,
        if (DibIsAddressable (hdib) && DibBitCount (hdib) > 8
                                iEnable = MF_ENABLED ;
                else
                                iEnable = MF_GRAYED ;

        EnableMenuItem      (hMenu, IDM_PAL_OPT_POP4,
        EnableMenuItem      (hMenu, IDM_PAL_OPT_POP5,
        EnableMenuItem      (hMenu, IDM_PAL_OPT_POP6,
        EnableMenuItem      (hMenu, IDM_PAL_OPT_MEDCUT,
        EnableMenuItem      (hMenu, IDM_EDIT_PASTE,
            IsClipboardFormatAvailable (CF_DIB) ? MF_ENABLED :

                return 0 ;
```

```
            // WM_COMMAND:  Process all menu commands

        case   WM_COMMAND:

                iConvert = 0 ;

                switch (LOWORD (wParam))

                {

               case   IDM_FILE_OPEN:

          // Show the File Open dialog box

                                    if (!GetOpenFileName (&ofn))

                                                return 0 ;


          // If there's an existing DIB and palette, delete them

                                     SendMessage (hwnd, WM_USER_I


         // Load the DIB into memory

                                    SetCursor (LoadCursor (NULL, IDC
```

```
                    ShowCursor (TRUE) ;

                    hdib = DibFileLoad (szFileName) ;

                    ShowCursor (FALSE) ;
                    SetCursor (LoadCursor (NULL, IDC

                              // Reset the scroll bars

                    SendMessage (hwnd, WM_USER_S

                              // Create the palette and

                    SendMessage (hwnd, WM_USER_C

                    if (!hdib)
                    {
MessageBox (hwnd, TEXT ("Cannot load DIB file!"),
          szAppName, MB_OK | MB_ICONEXCLAMATION)
                    }
                    InvalidateRect (hwnd, NULL, TRUE
                    return 0 ;
```

```
case   IDM_FILE_SAVE:

                // Show the File Save dialog box

                        if     (!     GetSaveFileName (&ofn)

                                return 0 ;

                // Save the DIB to memory

                        SetCursor (LoadCursor (NULL, IDC

                        ShowCursor (TRUE) ;

                        fSuccess = DibFileSave (hdib, szF

                        ShowCursor (FALSE) ;

                        SetCursor (LoadCursor (NULL, IDC

                        if (!fSuccess)

     MessageBox (    hwnd, TEXT ("Cannot save DIB file!")

                szAppName, MB_OK | MB_ICONEXCLAMATION

                        return 0 ;
```

```
                case   IDM_FILE_PRINT:
                                if (!hdib)
                                        return 0 ;

                                // Get printer DC

                printdlg.Flags = PD_RETURNDC | PD_NOPAGENU

                                if (!PrintDlg (&printdlg))
                                        return 0 ;

                                 if (NULL == (hdcPrn = printdlg.hD
                                 {
          MessageBox(    hwnd, TEXT ("Cannot obtain Printer D
                  szAppName, MB_ICONEXCLAMATION | MB_OK)
                                        return 0 ;
                                 }
                // Check if the printer can print bitmaps

    if (!(RC_BITBLT & GetDeviceCaps (hdcPrn, RASTERCAPS)))
                                 {
```

```
                    DeleteDC (hdcPrn) ;
MessageBox (     hwnd, TEXT ("Printer cannot print bitmaps
          szAppName, MB_ICONEXCLAMATION | MB_OK) ;
                    return 0 ;
                }
                     // Get size of printable area
                cxPage = GetDeviceCaps (hdcPrn
                cyPage = GetDeviceCaps (hdcPrn

                fSuccess = FALSE ;

                     // Send the DIB to the printe

              SetCursor (LoadCursor (NULL, IDC_WAI
               ShowCursor (TRUE) ;

               if ((StartDoc (hdcPrn, &di) > 0) && (St
          {
        DisplayDib (hdcPrn, DibBitmapHandle (hdib), 0,
              cxPage, cyPage, wShow, FALSE) ;
```

```
                    if (EndPage (hdcPrn) > 0)
                    {
                                fSuccess = TR
                                EndDoc (hdcP
                    }
          }
          ShowCursor (FALSE) ;
          SetCursor (LoadCursor (NULL, IDC_ARR

          DeleteDC (hdcPrn) ;

          if (   !fSuccess)
MessageBox (    hwnd, TEXT ("Could not print bitmap
          szAppName, MB_ICONEXCLAMATION | MB_OK
          return 0 ;

case   IDM_FILE_PROPERTIES:
          if (!hdib)
                    return 0 ;
```

```
            wsprintf (szBuffer, TEXT ("Pixel width:\t%i\n")
                      TEXT ("Pixel height:\t%i\n")
                     TEXT ("Bits per pixel:\t%i\n")
                      TEXT ("Number of colors:\t%i\n")
                     TEXT ("Compression:\t%s\n"),
          DibWidth (hdib), DibHeight (hdib),
          DibBitCount (hdib), DibNumColors (hdib),
          szCompression [min (3, DibCompression (hdib))])

                      MessageBox (  hwnd, szBuffer, szAppName
          MB_ICONEXCLAMATION | MB_OK) ;
                      return 0 ;

     case   IDM_APP_EXIT:
                      SendMessage (hwnd, WM_CLOSE, 0, 0
                      return 0 ;

     case   IDM_EDIT_COPY:
     case   IDM_EDIT_CUT:
```

```
                    if (!(hGlobal = DibCopyToPackedDib (h
                              return 0 ;

                    OpenClipboard (hwnd) ;
                    EmptyClipboard () ;
                    SetClipboardData (CF_DIB, hGlobal) ;
                    CloseClipboard () ;

                    if (LOWORD (wParam) == IDM_EDIT_C
          return 0 ;
           // fall through for IDM_EDIT_CUT
     case   IDM_EDIT_DELETE:
                    SendMessage (hwnd, WM_USER_DELE
                    InvalidateRect (hwnd, NULL, TRUE) ;
                    return 0 ;

     case   IDM_EDIT_PASTE:
                    OpenClipboard (hwnd) ;

                    hGlobal = GetClipboardData (CF_DIB)
                    pGlobal = GlobalLock (hGlobal) ;
```

```
            // If there's an existing DIB and palette,delete th
            // Then convert the packed DIB to an HDIB.

                        if (pGlobal)
                        {
                    SendMessage (hwnd, WM_USER_DELETEDIB, 0, 0
                    hdib = DibCopyFromPackedDib ((BITMAPINFO *)
                     SendMessage (hwnd, WM_USER_CREATEPAL, TR
                        }

                        GlobalUnlock (hGlobal) ;
                        CloseClipboard () ;
                              // Reset the scroll bars

                        SendMessage (hwnd, WM_USER_SETS
                        InvalidateRect (hwnd, NULL, TRUE) ;
                        return 0 ;

        case   IDM_EDIT_ROTATE:
                        if (hdibNew = DibRotateRight (hdib))
```

```
                    {
                DibDelete (hdib) ;
                DeleteObject (hBitmap) ;
                hdib = hdibNew ;
                hBitmap = DibCopyToDdb (hdib, hwnd, hPa
               SendMessage (hwnd, WM_USER_SETSCROL
                InvalidateRect (hwnd, NULL, TRUE) ;
                    }
                    else
                    {
          MessageBox (   hwnd, TEXT ("Not enough memory")
                szAppName, MB_OK | MB_ICONEXCLAMATIC
                    }
                    return 0 ;

        case   IDM_EDIT_FLIP:
                    if (hdibNew = DibFlipHorizontal (hdib))
                    {
                DibDelete (hdib) ;
```

```
                DeleteObject (hBitmap) ;
                hdib = hdibNew ;
                hBitmap = DibCopyToDdb (hdib, hwnd, hPa
                InvalidateRect (hwnd, NULL, TRUE) ;
                }
                else
                {
MessageBox (   hwnd, TEXT ("Not enough memory")
                szAppName, MB_OK | MB_ICONEXCLAMATI
                }
                return 0 ;

case   IDM_SHOW_NORMAL:
case   IDM_SHOW_CENTER:
case   IDM_SHOW_STRETCH:
case   IDM_SHOW_ISOSTRETCH:
                CheckMenuItem (hMenu, wShow, MF_U
                wShow = LOWORD (wParam) ;
                CheckMenuItem (hMenu, wShow, MF_C
```

```
                    SendMessage (hwnd, WM_USER_SETS

                    InvalidateRect (hwnd, NULL, TRUE) ;
                    return 0 ;

          case   IDM_CONVERT_32:  iConvert += 8 ;
          case   IDM_CONVERT_24:  iConvert += 8 ;
          case   IDM_CONVERT_16:  iConvert += 8 ;
          case   IDM_CONVERT_08:  iConvert += 4 ;
          case   IDM_CONVERT_04:  iConvert += 3 ;
          case   IDM_CONVERT_01:  iConvert += 1 ;
                    SetCursor (LoadCursor (NULL, IDC_WA
                    ShowCursor (TRUE) ;

                    hdibNew = DibConvert (hdib, iConvert

                    ShowCursor (FALSE) ;
                    SetCursor (LoadCursor (NULL, IDC_ARR

                    if (hdibNew)
                    {
```

```
                SendMessage (hwnd, WM_USER_DELETEDIB, 0, 0) ;

                                hdib = hdibNew ;

                SendMessage (hwnd, WM_USER_CREATEPAL, TRUE, (

               InvalidateRect (hwnd, NULL, TRUE) ;

                                }

                                else

                                {

                 MessageBox (  hwnd, TEXT ("Not enough memory")

                            szAppName, MB_OK | MB_ICONEXCLAMATIO

                                        }

                                         return 0 ;

                      case   IDM_APP_ABOUT:

           MessageBox (    hwnd, TEXT ("Dibble (c) Charles Petzold, 1

                     szAppName, MB_OK | MB_ICONEXCLAMATION) ;

                            return 0 ;

                     }


                            // All the other WM_COMMAND messages a
```

```
            //     items. Any existing palette is deleted,
            //     is set to the hourglass.

          SendMessage (hwnd, WM_USER_DELETEPAL, 0,
          SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
          ShowCursor (TRUE) ;
            // Notice that all messages for palette item
            //     with break rather than return. This is t
            //     additional processing later on.

          switch (LOWORD (wParam))
          {
          case   IDM_PAL_DIBTABLE:
                        hPalette = DibPalDibTable (hdib)
                        break ;

          case   IDM_PAL_HALFTONE:
                        hdc = GetDC (hwnd) ;

                       if (hPalette = CreateHalftonePalet
```

```
                                        fHalftonePalette = TRUE

                                ReleaseDC (hwnd, hdc) ;
                                break ;

                        case   IDM_PAL_ALLPURPOSE:
                                hPalette = DibPalAllPurpose () ;
                                break ;

    case IDM_PAL_GRAY2:hPalette    = DibPalUniformGrays   (2);
    case IDM_PAL_GRAY3:hPalette    = DibPalUniformGrays  (3);
    case IDM_PAL_GRAY4:hPalette    = DibPalUniformGrays  (4);
    case IDM_PAL_GRAY8:hPalette    = DibPalUniformGrays   (8);
    case IDM_PAL_GRAY16:hPalette      = DibPalUniformGrays  (1
    case IDM_PAL_GRAY32:hPalette      = DibPalUniformGrays  (3
    case IDM_PAL_GRAY64:hPalette      = DibPalUniformGrays  (6
    case IDM_PAL_GRAY128:hPalette     = DibPalUniformGrays (1
    case IDM_PAL_GRAY256:hPalette     = DibPalUniformGrays (2
    case IDM_PAL_RGB222:hPalette      = DibPalUniformColors (2
    case IDM_PAL_RGB333:hPalette      = DibPalUniformColors (3
```

```
    case IDM_PAL_RGB444:hPalette      = DibPalUniformColors (4
    case IDM_PAL_RGB555:hPalette      = DibPalUniformColors (5
    case IDM_PAL_RGB666:hPalette      = DibPalUniformColors (6
    case IDM_PAL_RGB775:hPalette      = DibPalUniformColors (7
    case IDM_PAL_RGB757:hPalette      = DibPalUniformColors (7
    case IDM_PAL_RGB577:hPalette      = DibPalUniformColors (5
    case IDM_PAL_RGB884:hPalette      = DibPalUniformColors (8
    case IDM_PAL_RGB848:hPalette      = DibPalUniformColors (8
    case IDM_PAL_RGB488:hPalette      = DibPalUniformColors (4
    case IDM_PAL_OPT_POP4:hPalette   = DibPalPopularity (hdib,
    case IDM_PAL_OPT_POP5:hPalette   = DibPalPopularity (hdib,
    case IDM_PAL_OPT_POP6:hPalette   = DibPalPopularity (hdib,
    case IDM_PAL_OPT_MEDCUT:hPalette = DibPalMedianCut  (hd
                    }

                    // After processing Palette items from the menu,
                    //     is restored to an arrow, the menu item is ch
                    //     the window is invalidated.

                    hBitmap = DibCopyToDdb (hdib, hwnd, hPalette
```

```
                ShowCursor (FALSE) ;
                SetCursor (LoadCursor (NULL, IDC_ARROW)) ;

                if (hPalette)
                                PaletteMenu (hMenu, (LOWORD (

               InvalidateRect (hwnd, NULL, TRUE) ;
                return 0 ;

                // This programmer-defined message deletes an
                //   in preparation for getting a new one.  Invoke
                //   File Open command, Edit Paste command, ar

case   WM_USER_DELETEDIB:
                if (hdib)
               {
                                DibDelete (hdib) ;
                                hdib = NULL ;
                }
                SendMessage (hwnd, WM_USER_DELETEPAL, 0,
```

```
                return 0 ;

               // This programmer-defined message deletes an
                //          in preparation for defining a new one.

        case   WM_USER_DELETEPAL:
                if (hPalette)
                {
                                DeleteObject (hPalette) ;
                                hPalette = NULL ;
                                fHalftonePalette = FALSE ;
                                PaletteMenu (hMenu, IDM_PAL_NO
                }
                if (hBitmap)
                                DeleteObject (hBitmap) ;

                      return 0 ;

                // Programmer-defined message to create a new
               //          a new DIB.  If wParam == TRUE, create
```

```
case   WM_USER_CREATEPAL:
            if (hdib)
            {
                        hdc = GetDC (hwnd) ;

            if (!(RC_PALETTE & GetDeviceCaps (hdc, RASTER
                        {
      PaletteMenu (hMenu, IDM_PAL_NONE) ;
                        }
                        else if (hPalette = DibPalDibTable
                    {
      PaletteMenu (hMenu, IDM_PAL_DIBTABLE) ;
                        }
                       else if (hPalette = CreateHalftoneP
                        {
                                fHalftonePalette = TRUE
      PaletteMenu (hMenu, IDM_PAL_HALFTONE) ;
                        }
                        ReleaseDC (hwnd, hdc) ;
```

```
                        if ((BOOL) wParam)
        hBitmap = DibCopyToDdb (hdib, hwnd, hPalette) ;
            }
            return 0 ;

    case   WM_PAINT:
            hdc = BeginPaint (hwnd, &ps) ;

            if (hPalette)
           {
                        SelectPalette (hdc, hPalette, FALS
                        RealizePalette (hdc) ;
            }
            if (hBitmap)
       {
                    DisplayDib (  hdc,
            fHalftonePalette ? DibBitmapHandle (hdib) : hBi
            iHscroll, iVscroll,
            cxClient, cyClient,
```

```
                wShow, fHalftonePalette) ;
                }
                EndPaint (hwnd, &ps) ;
                return 0 ;

        case   WM_QUERYNEWPALETTE:
                if (!hPalette)
                                return FALSE ;

                hdc = GetDC (hwnd) ;
                SelectPalette (hdc, hPalette, FALSE) ;
                RealizePalette (hdc) ;
                InvalidateRect (hwnd, NULL, TRUE) ;

               ReleaseDC (hwnd, hdc) ;
                return TRUE ;

         case   WM_PALETTECHANGED:
                if (!hPalette || (HWND) wParam == hwnd)
                                break ;
```

```
                hdc = GetDC (hwnd) ;
                SelectPalette (hdc, hPalette, FALSE) ;
                RealizePalette (hdc) ;
                UpdateColors (hdc) ;

                ReleaseDC (hwnd, hdc) ;
                break ;

        case   WM_DESTROY:
                if (hdib)
                                DibDelete (hdib) ;

                if (hBitmap)
                                DeleteObject (hBitmap) ;

                if (hPalette)
                                DeleteObject (hPalette) ;

                PostQuitMessage (0) ;
                return 0 ;
        }
```

```
        return DefWindowProc (hwnd, message, wParam, lParam)
}
```

```
DIBBLE.RC
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
//////////////////////////////////////////////////////////////////////////////
// Menu
DIBBLE MENU DISCARDABLE BEGIN POPUP "&File"
  BEGIN
        MENUITEM "&Open...\tCtrl+O",       IDM_FILE_OPEN
        MENUITEM "&Save...\tCtrl+S",       IDM_FILE_SAVE
       MENUITEM SEPARATOR
        MENUITEM "&Print...\tCtrl+P",      IDM_FILE_PRINT
        MENUITEM SEPARATOR
        MENUITEM "Propert&ies...",         IDM_FILE_PROPERTIES
        MENUITEM SEPARATOR
        MENUITEM "E&xit",                  IDM_APP_EXIT
```

```
 END
POPUP "&Edit"
BEGIN
        MENUITEM "Cu&t\tCtrl+X",      IDM_EDIT_CUT
        MENUITEM "&Copy\tCtrl+C",       IDM_EDIT_COPY
MENUITEM "&Paste\tCtrl+V",    IDM_EDIT_PASTE
       MENUITEM "&Delete\tDelete",   IDM_EDIT_DELETE
       MENUITEM SEPARATOR
   MENUITEM "&Flip",                    IDM_EDIT_FLIP
       MENUITEM "&Rotate",          IDM_EDIT_ROTATE
END
POPUP "&Show"
BEGIN
   MENUITEM "&Actual Size",   IDM_SHOW_NORMAL, CHECKED
   MENUITEM "&Center",        IDM_SHOW_CENTER
   MENUITEM "&Stretch to Window",           IDM_SHOW_STRET
   MENUITEM "Stretch &Isotropically",IDM_SHOW_ISOSTRETCH
END
```

```
POPUP "&Palette"
BEGIN
    MENUITEM "&None",                    IDM_PAL_NONE, CHECK
    MENUITEM "&Dib ColorTable",      IDM_PAL_DIBTABLE
    MENUITEM "&Halftone",    IDM_PAL_HALFTONE
    MENUITEM "&All-Purpose",         IDM_PAL_ALLPURPOSE
        POPUP "&Gray Shades"
        BEGIN
    MENUITEM "&1. 2 Grays",          IDM_PAL_GRAY2
    MENUITEM "&2. 3 Grays",      IDM_PAL_GRAY3
    MENUITEM "&3. 4 Grays",      IDM_PAL_GRAY4
    MENUITEM "&4. 8 Grays",      IDM_PAL_GRAY8
    MENUITEM "&5. 16 Grays",     IDM_PAL_GRAY16
    MENUITEM "&6. 32 Grays",     IDM_PAL_GRAY32
    MENUITEM "&7. 64 Grays",     IDM_PAL_GRAY64
    MENUITEM "&8. 128 Grays",    IDM_PAL_GRAY128
    MENUITEM "&9. 256 Grays",    IDM_PAL_GRAY256
        END
        POPUP "&Uniform Colors"
```

```
    BEGIN
MENUITEM "&1. 2R x 2G x 2B (8)",              IDM_PAL_RGB2
MENUITEM "&2. 3R x 3G x 3B (27)",        IDM_PAL_RGB333
MENUITEM "&3. 4R x 4G x 4B (64)",          IDM_PAL_RGB444
MENUITEM "&4. 5R x 5G x 5B (125)",       IDM_PAL_RGB555
MENUITEM "&5. 6R x 6G x 6B (216)",       IDM_PAL_RGB666
MENUITEM "&6. 7R x 7G x 5B (245)",      IDM_PAL_RGB775
MENUITEM "&7. 7R x 5B x 7B (245)",     IDM_PAL_RGB757
MENUITEM "&8. 5R x 7G x 7B (245)",      IDM_PAL_RGB577
MENUITEM "&9. 8R x 8G x 4B (256)",       IDM_PAL_RGB884
MENUITEM "&A. 8R x 4G x 8B (256)",       IDM_PAL_RGB848
    MENUITEM "&B. 4R x 8G x 8B (256)",       IDM_PAL_RGB48
    END
    POPUP "&Optimized"
  BEGIN
MENUITEM "&1. Popularity Algorithm (4 bits)"IDM_PAL_OPT_
MENUITEM "&2. Popularity Algorithm (5 bits)"IDM_PAL_OPT_
MENUITEM "&3. Popularity Algorithm (6 bits)"IDM_PAL_OPT_
```

```
      MENUITEM "&4. Median Cut Algorithm ", IDM_PAL_OPT_MED
         END
   END
   POPUP "Con&vert"
   BEGIN
      MENUITEM "&1. to 1 bit per pixel",        IDM_CONVERT_01
     MENUITEM "&2. to 4 bits per pixel",       IDM_CONVERT_04
      MENUITEM "&3. to 8 bits per pixel",      IDM_CONVERT_08
      MENUITEM "&4. to 16 bits per pixel",      IDM_CONVERT_16
      MENUITEM "&5. to 24 bits per pixel",      IDM_CONVERT_24
     MENUITEM "&6. to 32 bits per pixel",      IDM_CONVERT_32
   END
   POPUP "&Help"
   BEGIN
               MENUITEM "&About",
                IDM_APP_ABOUT
   END
END
//////////////////////////////////////////////////////////////////////////
```

```
// Accelerator
DIBBLE ACCELERATORS DISCARDABLE
BEGIN
  "C",          IDM_EDIT_COPY,          VIRTKEY, CONTROL, NOINVER
  "O",      IDM_FILE_OPEN,          VIRTKEY, CONTROL, NOINVERT
  "P",      IDM_FILE_PRINT,         VIRTKEY, CONTROL, NOINVERT
  "S",      IDM_FILE_SAVE,          VIRTKEY, CONTROL, NOINVERT
  "V",      IDM_EDIT_PASTE,         VIRTKEY, CONTROL, NOINVERT
  VK_DELETE,            IDM_EDIT_DELETE,        VIRTKEY, NOINVERT
  "X",      IDM_EDIT_CUT,           VIRTKEY, CONTROL, NOINVERT
END
```

```
RESOURCE.H
// Microsoft Developer Studio generated include file.
// Used by Dibble.rc
#define IDM_FILE_OPEN              40001
#define IDM_FILE_SAVE              40002
#define IDM_FILE_PRINT             40003
#define IDM_FILE_PROPERTIES        40004
```

```
#define IDM_APP_EXIT              40005
#define IDM_EDIT_CUT               40006
#define IDM_EDIT_COPY              40007
#define IDM_EDIT_PASTE             40008
#define IDM_EDIT_DELETE            40009
#define IDM_EDIT_FLIP             40010
#define IDM_EDIT_ROTATE            40011
#define IDM_SHOW_NORMAL            40012
#define IDM_SHOW_CENTER            40013
#define IDM_SHOW_STRETCH           40014
#define IDM_SHOW_ISOSTRETCH        40015
#define IDM_PAL_NONE              40016
#define IDM_PAL_DIBTABLE           40017
#define IDM_PAL_HALFTONE           40018
#define IDM_PAL_ALLPURPOSE         40019
#define IDM_PAL_GRAY2              40020
#define IDM_PAL_GRAY3              40021
#define IDM_PAL_GRAY4              40022
```

```
#define IDM_PAL_GRAY8              40023
#define IDM_PAL_GRAY16             40024
#define IDM_PAL_GRAY32             40025
#define IDM_PAL_GRAY64             40026
#define IDM_PAL_GRAY128            40027
#define IDM_PAL_GRAY256            40028
#define IDM_PAL_RGB222             40029
#define IDM_PAL_RGB333             40030
#define IDM_PAL_RGB444             40031
#define IDM_PAL_RGB555             40032
#define IDM_PAL_RGB666             40033
#define IDM_PAL_RGB775             40034
#define IDM_PAL_RGB757             40035
#define IDM_PAL_RGB577             40036
#define IDM_PAL_RGB884             40037
#define IDM_PAL_RGB848             40038
#define IDM_PAL_RGB488             40039
#define IDM_PAL_OPT_POP4           40040
#define IDM_PAL_OPT_POP5           40041
```

```
#define IDM_PAL_OPT_POP6          40042
#define IDM_PAL_OPT_MEDCUT        40043
#define IDM_CONVERT_01            40044
#define IDM_CONVERT_04            40045
#define IDM_CONVERT_08            40046
#define IDM_CONVERT_16            40047
#define IDM_CONVERT_24            40048
#define IDM_CONVERT_32            40049
#define IDM_APP_ABOUT             40050
```

DIBBLEDIBCONVDIBCONV.CDIBCONV.H248DIBPAL
DIBPAL.CDIBPAL.H

DIBBLEWndProchdibHDIBhPaletteHPALETTEhBitmapHBITMAP
HDIBDIBHELPHPALETTEDIBPALCreateHalftonePaletteHBITMAP
DIBHELP.CDibCopyToDdb256DIB

DIBBLE

DIBBLEIDM_FILE_LOADIDM_FILE_SAVEWM_COMMANDDIB
DIBBLEGetOpenFileNameGetSaveFileName

FileSaveDIBBLEDibFileSaveFileOpenDIBBLE
HDIBWM_USER_DELETEDIBDibDeleteDeleteObjectDIBBLE
DIBHELPDibFileLoadWM_USER_SETSCROLLS

WM_USER_CREATEPALWM_USER_CREATEPALDIBDDB

DIBBLEDIBDIBDIBBLEShow

WM_PAINTFilePrintDIBBLEDisplayDibDisplayDibBitBlt StretchBltSetDIBitsToDeviceStretchDIBitsWM_PAINTDibCopyToDdb WM_USER_CREATEPALDDBFilePrintDIBBLE DisplayDibDIBDIBHELP.CDibBitmapHandle

DIBBLEfHalftonePaletteBOOLCreateHalftonePaletteHPALETTETRUE DisplayDibStretchBltBitBltDIBfHalftonePaletteWM_PAINTDIB DisplayDibDibCopyToDdbSHOWDIB5

DIBBLEDIBDIBWM_PAINTWndProcDisplayDib

CutCopyDIBBLEDIBHELPDibCopyToPackedDibDIB

DIBBLEDIBDibCopyFromPackedDibHDIB

DIBBLEEditCutCopyPasteDeleteFlip RotateFlipRotate90DIBDIB

FlipDibFlipHorizontalDIBBLE.CDibCopyDIBDIBDIB DibGetPixelDibSetPixelDIBHELP.C

DibGetPixelDibSetPixelDIBHELP.HDibGetPixelDibSetPixel DibRotateRightDibCopyTRUEDibCopyDIBDIB DibCopyDibRotateRightDIBDIBDIB148162432

Flip HorizontalRotate RightFlip VerticalRotate LeftRotate 180°DIBBLE

DIBBLE256DIBDIBBLEPaletteWindows16-24DIBPAL

16-24 DIBPAL

```
DIBPAL.H
/*--------------------------------------------------------------------------
  DIBPAL.H header file for DIBPAL.C
--------------------------------------------------------------------------*/
HPALETTE DibPalDibTable (HDIB hdib) ;
HPALETTE DibPalAllPurpose (void) ;
HPALETTE DibPalUniformGrays (int iNum) ;
HPALETTE DibPalUniformColors (int iNumR, int iNumG, int iNumB
HPALETTE DibPalVga (void) ;
HPALETTE DibPalPopularity (HDIB hdib, int iRes) ;
HPALETTE DibPalMedianCut (HDIB hdib, int iRes) ;
```

```
DIBPAL.C
/*--------------------------------------------------------------------------
  DIBPAL.C --   Palette-Creation Functions
```

```
                                        (c) Charles Petzold, 1998
----------------------------------------------------------------------*/
#include <windows.h>
#include "dibhelp.h"
#include "dibpal.h"

/*----------------------------------------------------------------------
  DibPalDibTable: Creates a palette from the DIB color table
----------------------------------------------------------------------*/

HPALETTE DibPalDibTable (HDIB hdib)
{
   HPALETTE                  hPalette ;
          int                        i, iNum ;
          LOGPALETTE    *      plp ;
          RGBQUAD                   rgb ;

          if (0 == (iNum = DibNumColors (hdib)))
                    return NULL ;
          plp = malloc (sizeof (LOGPALETTE) + (iNum - 1) * sizeof (
```

```
        plp->palVersion       = 0x0300 ;
        plp->palNumEntries    = iNum ;

        for (i = 0 ; i < iNum ; i++)
        {
                DibGetColor (hdib, i, &rgb) ;
                plp->palPalEntry[i].peRed   = rgb.rgbRed ;
                plp->palPalEntry[i].peGreen = rgb.rgbGreen ;
                plp->palPalEntry[i].peBlue  = rgb.rgbBlue ;
                plp->palPalEntry[i].peFlags = 0 ;
        }
        hPalette = CreatePalette (plp) ;
        free (plp) ;
        return hPalette ;
}
/*---------------------------------------------------------------------
  DibPalAllPurpose: Creates a palette suitable for a wide variety
                    of images; the palette has 247 entries, but
                     duplicates or match the standard 20 colors
```

```
-------------------------------------------------------------------------*/

HPALETTE DibPalAllPurpose (void)
{
        HPALETTE                hPalette ;
        int                      i, incr, R, G, B ;
        LOGPALETTE * plp ;

        plp = malloc (sizeof (LOGPALETTE) + 246 * sizeof (PALET
        plp->palVersion             = 0x0300 ;
        plp->palNumEntries    = 247 ;

                // The following loop calculates 31 gray shades,
                //                 will match the standard 20 colors

        for (i = 0, G = 0, incr = 8 ; G <= 0xFF ; i++, G += incr)
   {
                plp->palPalEntry[i].peRed           = (BYTE) G ;
                plp->palPalEntry[i].peGreen   = (BYTE) G ;
                plp->palPalEntry[i].peBlue    = (BYTE) G ;
                plp->palPalEntry[i].peFlags   = 0 ;
```

```
                incr = (incr == 9 ? 8 : 9) ;
}

      // The following loop is responsible for 216 entries, but 8
            //                    them will match the standard 20 c
            //                    4 of them will match the gray sha

for (R = 0 ; R <= 0xFF ; R += 0x33)
      for (G = 0 ; G <= 0xFF ; G += 0x33)
      for (B = 0 ; B <= 0xFF ; B += 0x33)
      {
                plp->palPalEntry[i].peRed               = (BYTE)
                plp->palPalEntry[i].peGreen        = (BYTE) G ;
                plp->palPalEntry[i].peBlue         = (BYTE) B ;
                plp->palPalEntry[i].peFlags        = 0 ;

                i++ ;
}
hPalette = CreatePalette (plp) ;
```

```
    free (plp) ;
    return hPalette ;
}

/*--------------------------------------------------------------------
    DibPalUniformGrays:  Creates a palette of iNum grays, uniform
--------------------------------------------------------------------*/

HPALETTE DibPalUniformGrays (int iNum)
{
          HPALETTE                    hPalette ;
          int                         i ;
          LOGPALETTE    *      plp ;

          plp = malloc (sizeof (LOGPALETTE) + (iNum - 1) * sizeof (
          plp->palVersion                  = 0x0300 ;
          plp->palNumEntries           = iNum ;

        for (i = 0 ; i < iNum ; i++)
          {
                    plp->palPalEntry[i].peRed          =
```

```
                    plp->palPalEntry[i].peGreen   =
                    plp->palPalEntry[i].peBlue  = (BYTE) (i * 255 / (i
                    plp->palPalEntry[i].peFlags = 0 ;
          }
        hPalette = CreatePalette (plp) ;
          free (plp) ;
          return hPalette ;
}

/*----------------------------------------------------------------------
   DibPalUniformColors: Creates a palette of iNumR x iNumG x iN
----------------------------------------------------------------------*/

HPALETTE DibPalUniformColors (int iNumR, int iNumG, int iNumB
{
          HPALETTE                  hPalette ;
          int                       i, iNum, R, G, B ;
          LOGPALETTE * plp ;

          iNum = iNumR * iNumG * iNumB ;
```

```
plp = malloc (sizeof (LOGPALETTE) + (iNum - 1) * sizeof (

plp->palVersion    = 0x0300 ;

plp->palNumEntries = iNumR * iNumG * iNumB ;

i = 0 ;

for (R = 0 ; R < iNumR ; R++)

for (G = 0 ; G < iNumG ; G++)

for (B = 0 ; B < iNumB ; B++)

{

     plp->palPalEntry[i].peRed     = (BYTE) (R * 255 / (iNu

     plp->palPalEntry[i].peGreen   = (BYTE) (G * 255 / (iN

     plp->palPalEntry[i].peBlue    = (BYTE) (B * 255 / (iNu

     plp->palPalEntry[i].peFlags   = 0 ;

          i++ ;

}

hPalette = CreatePalette (plp) ;

free (plp) ;

return hPalette ;
```

```
}

/*--------------------------------------------------------------------------
   DibPalVga:  Creates a palette based on standard 16 VGA color
--------------------------------------------------------------------------*/

HPALETTE DibPalVga (void)
{
   static RGBQUAD rgb [16] = { 0x00, 0x00, 0x00,0x00,
                               0x00, 0x00, 0x80, 0x00,
                              0x00, 0x80, 0x00, 0x00,
                               0x00, 0x80, 0x80, 0x00,
                               0x80, 0x00, 0x00, 0x00,
                               0x80, 0x00, 0x80, 0x00,
                               0x80, 0x80, 0x00, 0x00,
                              0x80, 0x80, 0x80, 0x00,
                               0xC0, 0xC0, 0xC0, 0x00,
                               0x00, 0x00, 0xFF, 0x00,
                               0x00, 0xFF, 0x00, 0x00,
                               0x00, 0xFF, 0xFF, 0x00,
```

```
                0xFF, 0x00, 0x00, 0x00,
                0xFF, 0x00, 0xFF, 0x00,
                0xFF, 0xFF, 0x00, 0x00,
                0xFF, 0xFF, 0xFF, 0x00 } ;
 HPALETTE                hPalette ;
int                     i ;
 LOGPALETTE             * plp ;

 plp = malloc (sizeof (LOGPALETTE) + 15 * sizeof (PALETTE
 plp->palVersion            = 0x0300 ;
 plp->palNumEntries    = 16 ;

 for (i = 0 ; i < 16 ; i++)
 {
          plp->palPalEntry[i].peRed     = rgb[i].rgbRed ;
           plp->palPalEntry[i].peGreen   = rgb[i].rgbGreen
           plp->palPalEntry[i].peBlue    = rgb[i].rgbBlue ;
           plp->palPalEntry[i].peFlags  = 0 ;
 }
```

```
        hPalette = CreatePalette (plp) ;
        free (plp) ;
        return hPalette ;
}

/*-------------------------------------------------------------------------
  Macro used in palette optimization routines
-------------------------------------------------------------------------*/

#define PACK_RGB(R,G,B,iRes) ((int) (R) | ((int) (G) <<  (iRes)) |
                        ((int) (B) << ((iRes) + (iRes))))

/*-------------------------------------------------------------------------
  AccumColorCounts: Fills up piCount (indexed by a packed RGB
   with counts of pixels of that color.
-------------------------------------------------------------------------*/

static void AccumColorCounts (HDIB hdib, int * piCount, int iRes)
{
        int          x, y, cx, cy ;
        RGBQUADrgb ;
```

```
        cx = DibWidth (hdib) ;
        cy = DibHeight (hdib) ;

        for (y = 0 ; y < cy ; y++)
        for (x = 0 ; x < cx ; x++)
        {
                DibGetPixelColor (hdib, x, y, &rgb) ;

                rgb.rgbRed   >>= (8 - iRes) ;
                rgb.rgbGreen >>= (8 - iRes) ;
                rgb.rgbBlue  >>= (8 - iRes) ;

                ++piCount [PACK_RGB (rgb.rgbRed, rgb.rgbGre
        }
}

/*--------------------------------------------------------------------
  DibPalPopularity:  Popularity algorithm for optimized colors
--------------------------------------------------------------------*/
```

```
HPALETTE DibPalPopularity (HDIB hdib, int iRes)

{

        HPALETTE                hPalette ;

        int                     i, iArraySize, iEntry, iCount, iIndex

        int                 *   piCount ;

        LOGPALETTE    *     plp ;

                // Validity checks


        if (DibBitCount (hdib) < 16)

                return NULL ;

        if (iRes < 3 || iRes > 8)

             return NULL ;

              // Allocate array for counting pixel colors

        iArraySize = 1 << (3 * iRes) ;

        iMask = (1 << iRes) - 1 ;

        if (NULL == (piCount = calloc (iArraySize, sizeof (int))))

                 return NULL ;

                // Get the color counts
```

```
AccumColorCounts (hdib, piCount, iRes) ;
            // Set up a palette
plp = malloc (sizeof (LOGPALETTE) + 235 * sizeof (PALET
plp->palVersion = 0x0300 ;
for (iEntry = 0 ; iEntry < 236 ; iEntry++)
{
            for (i = 0, iCount = 0 ; i < iArraySize ; i++)
                        if (piCount[i] > iCount)

                        {
                                    iCount = piCount[i] ;
                                    iIndex = i ;
                        }
            if (iCount == 0)
                        break ;
            R = (iMask &  iIndex) << (8 - iRes) ;
G = (iMask & (iIndex >> iRes )) << (8 - iRes) ;
B = (iMask & (iIndex >> (iRes + iRes)))<< (8 - iRes) ;
```

```
                    plp->palPalEntry[iEntry].peRed      = (BYTE) R ;
                    plp->palPalEntry[iEntry].peGreen    = (BYTE) G
                    plp->palPalEntry[iEntry].peBlue     = (BYTE) B ;
                    plp->palPalEntry[iEntry].peFlags    = 0 ;

                    piCount [iIndex] = 0 ;
    }
              // On exit from the loop iEntry will be the number of s
          plp->palNumEntries = iEntry ;
               // Create the palette, clean up, and return the palette
          hPalette = CreatePalette (plp) ;
          free (piCount) ;
          free (plp) ;

          return hPalette ;
}

/*------------------------------------------------------------------------
  Structures used for implementing median cut algorithm
------------------------------------------------------------------------*/
```

```
typedef struct                          // defines dimension of a bo
{
        int Rmin, Rmax, Gmin, Gmax, Bmin, Bmax ;
}
MINMAX ;
typedef struct                          // for Compare routine for q
{
        int             iBoxCount ;
        RGBQUAD     rgbBoxAv ;
}

BOXES ;
/*-------------------------------------------------------------------
  FindAverageColor: In a box
-------------------------------------------------------------------*/

static int FindAverageColor (     int * piCount, MINMAX mm,
                                  int iRes, RGBQUAD * prgb)
{
```

```
int R, G, B, iR, iG, iB, iTotal, iCount ;

            // Initialize some variables

iTotal = iR = iG = iB = 0 ;

            // Loop through all colors in the box

for (R = mm.Rmin ; R <= mm.Rmax ; R++)

for (G = mm.Gmin ; G <= mm.Gmax ; G++)

for (B = mm.Bmin ; B <= mm.Bmax ; B++)

{

                        // Get the number of pixels of that colo

           iCount = piCount [PACK_RGB (R, G, B, iRes)] ;

                        // Weight the pixel count by the color v

            iR += iCount * R ;

            iG += iCount * G ;

            iB += iCount * B ;


            iTotal += iCount ;

}

           // Find the average color

prgb->rgbRed        = (BYTE) ((iR / iTotal) << (8 - iRes)) ;
```

```
        prgb->rgbGreen       = (BYTE) ((iG / iTotal) << (8 - iRes))
        prgb->rgbBlue        = (BYTE) ((iB / iTotal) << (8 - iRes)) ;

                    // Return the total number of pixels in the box

        return iTotal ;
}

/*-----------------------------------------------------------------------
  CutBox:  Divide a box in two
-----------------------------------------------------------------------*/
static void CutBox (int * piCount, int iBoxCount, MINMAX mm,
                int iRes, int iLevel, BOXES * pboxes, int * piEntry)
{
        int          iCount, R, G, B ;
        MINMAX mmNew ;

                    // If the box is empty, return

        if (iBoxCount == 0)
```

```
                return ;

                // If the nesting level is 8, or the box is one pixel
                // to find the average color in the box and save i
                // the number of pixels of that color

    if (iLevel == 8 || (  mm.Rmin == mm.Rmax &&
                    mm.Gmin == mm.Gmax &&
                    mm.Bmin == mm.Bmax))
{
        pboxes[*piEntry].iBoxCount =
                FindAverageColor (piCount, mm, iRes, &pboxes[
                (*piEntry) ++ ;
    }
            // Otherwise, if blue is the largest side, split it
    else if ((mm.Bmax - mm.Bmin > mm.Rmax - mm.Rmin) &
                            (mm.Bmax - mm.Bmin > mm.Gm
    {
                        // Initialize a counter and loop through
        iCount = 0 ;
```

```
for (B = mm.Bmin ; B < mm.Bmax ; B++)
{
                        // Accumulate all the pixels for ea
              for (  R = mm.Rmin ; R <= mm.Rmax ; R+
              for (  G = mm.Gmin ; G <= mm.Gmax ; G+
              iCount += piCount [PACK_RGB (R, G, B, iRe

              // If it's more than half the box count, we're

            if (i  Count >= iBoxCount / 2)
                        break ;

       //     If the next blue value will be the max, we're
                                  if (   B == mm.Bmax - 1
                                        break ;
}
                    // Cut the two split boxes.
                    //   The second argument to CutBox is
                    //   The third argument is the new min
```

```
            mmNew = mm ;
            mmNew.Bmin = mm.Bmin ;
            mmNew.Bmax = B ;

            CutBox (      piCount, iCount, mmNew, iRes, iLevel +
                                        pboxes, piEntry) ;

            mmNew.Bmin = B + 1 ;
            mmNew.Bmax = mm.Bmax ;

            CutBox (      piCount, iBoxCount - iCount, mmNew, iR
                                        pboxes, piEntry) ;
}
            // Otherwise, if red is the largest side, split it (just like
      else if (mm.Rmax - mm.Rmin > mm.Gmax - mm.Gmin)
{
            iCount = 0 ;
            for (R = mm.Rmin ; R < mm.Rmax ; R++)
            {
                              for (B = mm.Bmin ; B <= mm.Bm
```

```
                                for (G = mm.Gmin ; G <= mm.Gm
                            iCount += piCount [PACK_RGB (R, G, B
                                if (iCount >= iBoxCount / 2)
                                        break ;
                                if (R == mm.Rmax - 1)
                                        break ;
            }
            mmNew = mm ;
            mmNew.Rmin = mm.Rmin ;
            mmNew.Rmax = R ;

            CutBox (      piCount, iCount, mmNew, iRes, iLevel +
                                        pboxes, piEntry) ;

            mmNew.Rmin    = R + 1 ;
           mmNew.Rmax    = mm.Rmax ;
            CutBox (      piCount, iBoxCount - iCount, mmNew, iR
                                        pboxes, piEntry) ;
}
                        // Otherwise, split along the green size
```

```
else
{
     iCount = 0 ;
     for (G = mm.Gmin ; G < mm.Gmax ; G++)
    {
                for (  B = mm.Bmin ; B <= mm.Bmax ; B+
                for (  R = mm.Rmin ; R <= mm.Rmax ; R+
                iCount += piCount [PACK_RGB (R, G, B, iRe

                if (   iCount >= iBoxCount / 2)
                          break ;

                if (   G == mm.Gmax - 1)
                           break ;
     }
     mmNew = mm ;
     mmNew.Gmin = mm.Gmin ;
     mmNew.Gmax = G ;

     CutBox (      piCount, iCount, mmNew, iRes, iLevel +
```

```
                                        pboxes, piEntry) ;

            mmNew.Gmin    = G + 1 ;
            mmNew.Gmax    = mm.Gmax ;

            CutBox (      piCount, iBoxCount - iCount, mmNew, iR
                                        pboxes, piEntry) ;
        }
}

/*---------------------------------------------------------------------
  Compare routine for qsort
---------------------------------------------------------------------*/

static int Compare (const BOXES * pbox1, const BOXES * pbox2)
{
        return pbox1->iBoxCount - pbox2->iBoxCount ;
}

/*---------------------------------------------------------------------
  DibPalMedianCut:  Creates palette based on median cut algorit
```

```
-----------------------------------------------------------------------*/
HPALETTE DibPalMedianCut (HDIB hdib, int iRes)
{
        BOXES                   boxes [256] ;
        HPALETTE                hPalette ;
        int                     i, iArraySize, iCount, R, G, B, iTotCount
        int                     *      piCount ;
       LOGPALETTE    *      plp ;
        MINMAX                  mm ;

                // Validity checks


        if (DibBitCount (hdib) < 16)
                return NULL ;
        if (iRes < 3 || iRes > 8)
                return NULL ;
                // Accumulate counts of pixel colors
   iArraySize = 1 << (3 * iRes) ;
   if (NULL == (piCount = calloc (iArraySize, sizeof (int))))
```

```
                return NULL ;

AccumColorCounts (hdib, piCount, iRes) ;

            // Find the dimensions of the total box

iDim = 1 << iRes ;

mm.Rmin = mm.Gmin = mm.Bmin = iDim - 1 ;

mm.Rmax = mm.Gmax = mm.Bmax = 0 ;

iTotCount = 0 ;

    for (R = 0 ; R < iDim ; R++)

    for (G = 0 ; G < iDim ; G++)

    for (B = 0 ; B < iDim ; B++)

                if ((iCount = piCount [PACK_RGB (R, G, B, iRes)]

                {

                                iTotCount += iCount ;

                                if (R < mm.Rmin) mm.Rmin = R ;

                                if (G < mm.Gmin) mm.Gmin = G

                                if (B < mm.Bmin) mm.Bmin = B ;

                                if (R > mm.Rmax) mm.Rmax = R

                                if (G > mm.Gmax) mm.Gmax = G
```

```
                        if (B > mm.Bmax) mm.Bmax = B ;
                }

        // Cut the first box (iterative function).
           // On return, the boxes structure will have up to 25(
        //     one for each of the boxes, and the number of pix
        //  each box.
        //  The iEntry value will indicate the number of non-e

CutBox (piCount, iTotCount, mm, iRes, 0, boxes, &iEntry)
free (piCount) ;

        // Sort the RGB table by the number of pixels for eacl
qsort (boxes, iEntry, sizeof (BOXES), Compare) ;
plp = malloc (sizeof (LOGPALETTE) + (iEntry - 1) * sizeof (
if (plp == NULL)
                return NULL ;
plp->palVersion             = 0x0300 ;
plp->palNumEntries    = iEntry ;
```

```
        for (i = 0 ; i < iEntry ; i++)
        {
                plp->palPalEntry[i].peRed  = boxes[i].rgbBoxAv.
                plp->palPalEntry[i].peGreen= boxes[i].rgbBoxAv
                plp->palPalEntry[i].peBlue = boxes[i].rgbBoxAv.
                plp->palPalEntry[i].peFlags= 0 ;
    }

    hPalette = CreatePalette (plp) ;
    free (plp) ;
    return hPalette ;
}
```

DibPalDibTableDIBSHOWDIB3PACKEDIB.CPackedDibCreatePalette SHOWDIB3DIB8162432DIB

256DIBBLEDibPalDibTableDIBDIBDIBBLE CreateHalftonePalettefHalftonePaletteTRUEWM_USER_CREATEPAL

DIBPAL.CDibPalAllPurposeSHOWDIB4CreateAllPurposePalette DIBBLE

256WindowsWinsows

DibPalUniformGrays00-00-00FF-FF-FF3 42

8163264664

8162432DIB

8162432DIBDIBGDIDIBDDB824DIB
SHOWDIB1SHOWDIB4

824DIBDIBDDBGDIDIB20RGBDIB

232DIBBLESHOWDIB411GDI23220DIB

8241632DIBDIB2568DIBPaul
Color Image Quantization for Frame Buffer Displays19827Computer Graphics

256RGBDibPalAllPurposeDIBDibPalCreateUniformColorsRGB

84RGB0x000x240x490x6D0x920xB60xDB0xFF
0x000x550xAA0xFF25660x000x330x660x99
0xCC0xFF63216

DIBBLE

Popularity

Popularity256256RGBDIBPALDibPalPopularity

2464MB24

n686256KB1MB532,76856DIBBLE

Median Cut

DIBPAL.CDibPalMedianCutPaul HeckbertMedian CutPopularity

RGB448163264128256

256RGB

octree quantizationJeff Prosise19968Mic
MSDNCD

DIBBLEDIBDIBCONVDibConvert16-25

16-25 DIBCONV

DIBCONV.H

```
/*--------------------------------------------------------------------------
  DIBCONV.H header file for DIBCONV.C
--------------------------------------------------------------------------*/

HDIB DibConvert (HDIB hdibSrc, int iBitCountDst) ;
```

DIBCONV.C

```
/*--------------------------------------------------------------------------
  DIBCONV.C --  Converts DIBs from one format to another
                                        (c) Charles Petzold, 1998
--------------------------------------------------------------------------*/
```

```
#include <windows.h>
#include "dibhelp.h"
#include "dibpal.h"
#include "dibconv.h"

HDIB DibConvert (HDIB hdibSrc, int iBitCountDst)
{
        HDIB                    hdibDst ;
        HPALETTE                hPalette ;
        int                     i, x, y, cx, cy, iBitCountSrc, cColo
        PALETTEENTRY pe ;
        RGBQUAD                 rgb ;
        WORD                    wNumEntries ;

        cx = DibWidth (hdibSrc) ;
        cy = DibHeight (hdibSrc) ;
        iBitCountSrc = DibBitCount (hdibSrc) ;

        if (iBitCountSrc == iBitCountDst)
                return NULL ;
```

```
        // DIB with color table to DIB with larger color table:
    if ((iBitCountSrc < iBitCountDst) && (iBitCountDst <= 8))
    {
        cColors = DibNumColors (hdibSrc) ;
        hdibDst = DibCreate (cx, cy, iBitCountDst, cColors) ;

        for (i = 0 ; i < cColors ; i++)
        {
                        DibGetColor (hdibSrc, i, &rgb) ;
                        DibSetColor (hdibDst, i, &rgb) ;
       }

        for (x = 0 ; x < cx ; x++)
        for (y = 0 ; y < cy ; y++)
        {
              DibSetPixel (hdibDst, x, y, DibGetPixel (hdibSrc,
        }
    }
       // Any DIB to DIB with no color table
    else if (iBitCountDst >= 16)
```

```
{
     hdibDst = DibCreate (cx, cy, iBitCountDst, 0) ;
     for (x = 0 ; x < cx ; x++)
     for (y = 0 ; y < cy ; y++)
     {
                    DibGetPixelColor (hdibSrc, x, y, &rgb) ;
                   DibSetPixelColor (hdibDst, x, y, &rgb) ;
 }
}
     // DIB with no color table to 8-bit DIB
else if (iBitCountSrc >= 16 && iBitCountDst == 8)
{
          hPalette = DibPalMedianCut (hdibSrc, 6) ;
          GetObject (hPalette, sizeof (WORD), &wNumEnt

          hdibDst = DibCreate (cx, cy, 8, wNumEntries) ;
          for (i = 0 ; i < (int) wNumEntries ; i++)
          {
                    GetPaletteEntries (hPalette, i, 1, &pe) ;
```

```
                    rgb.rgbRed      = pe.peRed ;
                    rgb.rgbGreen    = pe.peGreen
                    rgb.rgbBlue     = pe.peBlue ;
                    rgb.rgbReserved = 0 ;

                    DibSetColor (hdibDst, i, &rgb) ;
               }

               for (x = 0 ; x < cx ; x++)
               for (y = 0 ; y < cy ; y++)
               {
                         DibGetPixelColor (hdibSrc, x, y, &

                         DibSetPixel (hdibDst, x, y,
                    GetNearestPaletteIndex (hPalette,
                     RGB (rgb.rgbRed, rgb.rgbGreen, rgb.rgbBlu
               }
               DeleteObject (hPalette) ;
          }
                    // Any DIB to monochrome DIB
```

```
else if (iBitCountDst == 1)
{
        hdibDst = DibCreate (cx, cy, 1, 0) ;
        hPalette = DibPalUniformGrays (2) ;

        for (i = 0 ; i < 2 ; i++)
        {
                        GetPaletteEntries (hPalette, i, 1, &

                         rgb.rgbRed   = pe.peRed ;
                         rgb.rgbGreen = pe.peGreen ;
                         rgb.rgbBlue  = pe.peBlue ;
                        rgb.rgbReserved = 0 ;

                         DibSetColor (hdibDst, i, &rgb) ;
        }

        for (x = 0 ; x < cx ; x++)
        for (y = 0 ; y < cy ; y++)
        {
```

```
                                DibGetPixelColor (hdibSrc, x, y, &

                                DibSetPixel (hdibDst, x, y,
                        GetNearestPaletteIndex (hPalette,
                        RGB (rgb.rgbRed, rgb.rgbGreen, rgb.rgbBlu
                }
                DeleteObject (hPalette) ;
        }
                // All non-monochrome DIBs to 4-bit DIB
        else if (iBitCountSrc >= 8 && iBitCountDst == 4)
        {
                hdibDst = DibCreate (cx, cy, 4, 0) ;
                hPalette = DibPalVga () ;

                for (i = 0 ; i < 16 ; i++)
                {
                                GetPaletteEntries (hPalette, i, 1, &
                                rgb.rgbRed                      = pe.
                                rgb.rgbGreen                = pe.peG
```

```
                    rgb.rgbBlue                 = pe
                    rgb.rgbReserved             = 0 ;

                    DibSetColor (hdibDst, i, &rgb) ;
               }

               for (x = 0 ; x < cx ; x++)
               for (y = 0 ; y < cy ; y++)
               {
                    DibGetPixelColor (hdibSrc, x, y, &

                         DibSetPixel (hdibDst, x, y,
                    GetNearestPaletteIndex (hPalette,
                    RGB (rgb.rgbRed, rgb.rgbGreen, rgb.rgbBlue))) ;
               }
               DeleteObject (hPalette) ;
          }
               // Should not be necessary
          else
               hdibDst = NULL ;
```

```
        return hdibDst ;

}
```

DIB

DIBDIB1DIB48DIB4DIB8DIBDibCreateDIBDIB

DIB162432DIBDibGetPixelColorDibSetPixelColorDIB

DIB162432DIB8DibConvertDibPalMedianCutDIB
RGBDibGetPixelColorDIBGetNearestPaletteIndex8DIBDibSetPixel
DIB

DIBDIBDIBGetNearestPaletteIndexDIB018DIB4DIB
DibPalVgaDIBGetNearestPaletteIndex

DIBBLE

Microsoft Windows

Windows 3.1TrueTypeTrueTypeApple
TrueTypeWindowsWYSIWYGwhat
TrueType

Windows

```
TextOut (hdc, xStart, yStart, pString, iCount) ;
```

xStartyStartWindowsTextOutNULL

TextOutxStartyStartSetTextAlignTA_LEFTTA_RIGHTTA_CENTER
xStartTA_LEFTSetTextAlignTA_RIGHTTextOutxStart
TA_CENTERxStart

TA_TOPTA_BOTTOMTA_BASELINETA_TOPyStart
TA_BOTTOMyStartTA_BASELINEyStartpqy

TA_UPDATECPSetTextAlignWindowsTextOutxStartyStartMoveToEx
LineToTA_UPDATECPTextOutTA_LEFTTA_RIGHTTextOut
TA_CENTERTextOut

SYSMETSTextOutTabbedTextOut

```
TabbedTextOut (    hdc, xStart, yStart, pString, iCount,
```

```
                                        iNumTabs, piTabStops, xTabOrigin
```

‘\t’0x09TabbedTextOut

TabbedTextOutTextOut854080120

0NULL1130306090…x

ExtTextOutExt

```
ExtTextOut (hdc, xStart, yStart, iOptions, &rect,
                                        pString, iCount, pxDistance) ;
```

iOptionsETO_CLIPPEDiOptionsETO_OPAQUE

NULL

DrawTextHELLOWINRECT

```
DrawText (hdc, pString, iCount, &rect, iFormat) ;
```

DrawTextDrawTextNULLiCount-1Windows

iFormat0Windowscarriage return‘\r’0x0Dlinefe
carriage returnlinefeedWindows

iFormatDrawTextiFormatDT_LEFTDT_RIGHT
DT_CENTERDT_LEFT0

carriage returnlinefeedDT_SINGLELINEWindowscarriage
linefeedDT_SINGLELINEDT_TOP(DT_BOTTOM)
(DT_VCETERV)

Windowscarriage returnlinefeedDT_WORDBREAK

WindowsWindowsDT_NOCLIPWindows
DT_EXTERNALLEADING

‘\t’0x09DT_EXPANDTABSDT_TABSTOPiFormat
DT_TABSTOPiFormat

DT_TABSTOPDrawTextEx

```
DrawTextEx (hdc, pString, iCount, &rect, iFormat, &drawtextpara
```

DRAWTEXTPARAMS

```
typedef struct tagDRAWTEXTPARAMS

{

  UINT   cbSize ;                    // size of structure

   int    iTabLength ;              // size of each tab stop

   int    iLeftMargin ;             // left margin

   int    iRightMargin ;            // right margin

   UINT   uiLengthDrawn ;           // receives number of charact

} DRAWTEXTPARAMS, * LPDRAWTEXTPARAMS ;
```

SerTextAlign

```
SetTextColor (hdc, rgbColor) ;
```

```

```

WindowsrgbColorGetTextColor

Windows

```
SetBkMode (hdc, iMode) ;
```

iModeOPAQUETRANSPARENTOPAQUEWindows

```
SetBkColor (hdc, rgbColor) ;
```

rgbColor

TRANSPARENTTRANSPARENTWindowsWindows

WindowsWHITE_BRUSHWindowsWNDCLASS

```
wndclass.hbrBackground = COLOR_WINDOW + 1 ;
```

```
SetTextColor (hdc, GetSysColor (COLOR_WINDOWTEXT)) ;
SetBkColor (hdc, GetSysColor (COLOR_WINDOW)) ;
```

```
case       WM_SYSCOLORCHANGE :
          InvalidateRect (hwnd, NULL, TRUE) ;
```

```
        break ;
```

0Windows

```
SetTextCharacterExtra (hdc, iExtra) ;
```

iExtraWindows0iExtraWindows─iExtra0
GetTextCharacterExtraWindows

TextOutTabbedTextOutExtTextOutDrawTextDrawTextExWindows
Windows

```
hFont = GetStockObject (iFont) ;
```

iFont

```
SelectObject (hdc, hFont) ;
```

```
SelectObject (hdc, GetStockObject (iFont)) ;
```

GetStockObjectSYSTEM_FONTANSIGetStockObject
SYSTEM_FIXED_FONTWindows

OEM_FIXED_FONTWindowsMS-DOSIBM-PCWindows
DEFULT_GUI_FONT

GetTextMetrics

GetStockObjectWindows

Windows

WindowsGDIGDI

GDITrueType

WindowsGDIGDI

System SYSTEM_FONT

FixedSys SYSTEM_FIXED_FONT

Terminal OEM_FIXED_FONT

Courier

MS Serif

MS Sans SerifDEFAULT_GUI_FONT

Small Fonts

6CourierSerifsans
MSMicrosoftSerifMS Sans SerifTms RmnTimes RomanH
HelveticaSmall Fonts

Windows3.1GDIWindows
ModernRomanScript

GDIGDIWindowsWindows

Truetype

TrueType

TrueTypeWindowsTrueType

TrueTypeWindowsWindowsTrueTypeH

Windows13TrueType

Courier New

Courier New Bold

Courier New Italic

Courier New Bold Italic

Times New Roman

Times New Roman Bold

Times New Roman Italic

Times New Roman Bold Italic

Arial

Arial Bold

Arial Italic

Arial Bold Italic

Symbol

WindowsLucida Sans Unicode

Courier NewTimes New
ArialHelveticasans serifSymbol

TrueTypeCourierTimes New RomanArial
WindowsTrueType

Windows

1/72…

GetTextMetrics4-3FONTMETRIC

TEXTMETRICtmExternalLeadingleadingtmInternalLeading
tmExternalLeading

812TEXTMETRICtmHeighttmHeighttmInternalLeading

Windows 981210tmHeight1620
tmHeighttmInternalLeading1316…
120dpi

LOGPIXELSXLOGPIXELSYGetDeviceCaps96120…

Windows NTWindows NTGetDeviceCapsLOGPIXELSX…
ORZRESHORZSIZE25.4LOGPIXELSYVERTRES
VERTSIZEWindowsHORZRESHORZSIZEVERTRESVERTSIZE
LOGPIXELSXLOGPIXELSYWindows

Windows NTWindowsWindows

```
SetMapMode (hdc, MM_ANISOTROPIC) ;

SetWindowExtEx     (hdc, 1440, 1440, NULL) ;

SetViewportExt     (hdc, GetDeviceCaps (hdc, LOGPIXELSX),

                         GetDeviceCaps (hdc, LOGPIXELSY), NULL) ;
```

2012240MM_TWIPSy

GDI

GDIHFONTSelectObjectWindows
Windows

CreateFontCreateFontIndirectCreateFontIndirectLOGFONT14
CreateFont14LOGFONT14Windows14CreateFont
CreateFontIndirect

LOGFONTCreateFontIndirect

- LOGFONTSelectObjectWindows

-
- ChooseFont LOGFONT

1. CreateFontCreateFontIndirectHFONT
2. SelectObjectWindows
3. GetTextMetrics
4. DeleteObject

GetTextFace

```
GetTextFace (hdc, sizeof (szFaceName) / sizeof (TCHAR), szFaceN
```

GetTextMetrics

```
GetTextMetrics (hdc, &textmetric) ;
```

textmetricTEXTMETRIC20

LOGFONTTEXTMETRICLOGFONTTEXTMETRIC

PICKFONT

17-1PICKFONTLOGFONT

17-1 PICKFONT

```
PICKFONT.C

/*------------------------------------------------------------------------
```

```
  PICKFONT.C --        Create Logical Font

                                                (c) Charles Petzold, 199
----------------------------------------------------------------------*/
#include <windows.h>
#include "resource.h"

   // Structure shared between main window and dialog box
typedef struct
{
         int                        iDevice, iMapMode ;
         BOOL                          fMatchAspect ;
         BOOL                          fAdvGraphics ;
         LOGFONT                    lf ;
         TEXTMETRIC           tm ;
        TCHAR                          szFaceName [LF_FULLFACESIZE
}
DLGPARAMS ;
         // Formatting for BCHAR fields of TEXTMETRIC structure
#ifdef UNICODE
```

```
#define BCHARFORM TEXT ("0x%04X")
#else
#define BCHARFORM TEXT ("0x%02X")
#endif

          // Global variables
HWND  hdlg ;
TCHAR szAppName[] = TEXT ("PickFont") ;

          // Forward declarations of functions
LRESULT     CALLBACK WndProc             (HWND, UINT, WPARAM,
BOOL             CALLBACK DlgProc      (HWND, UINT, WPARAM, LP
void SetLogFontFromFields          (HWND hdlg, DLGPARAMS * pdp
void SetFieldsFromTextMetric       (HWND hdlg, DLGPARAMS * pdp
void MySetMapMode  (HDC hdc, int iMapMode) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                                PSTR szCmdLine, int iCr
{
          HWND                          hwnd ;
```

```
MSG                 msg ;
WNDCLASS            wndclass ;

wndclass.style                  = CS_HREDRAW | CS_VRED
wndclass.lpfnWndProc                = WndProc ;
wndclass.cbClsExtra                 = 0 ;
wndclass.cbWndExtra                 = 0 ;
wndclass.hInstance                  = hInstance ;
wndclass.hIcon                  = LoadIcon (NULL, IDI_
wndclass.hCursor                = LoadCursor (NULL,
wndclass.hbrBackground          = (HBRUSH) GetStockO
wndclass.lpszMenuName           = szAppName ;
wndclass.lpszClassName          = szAppName ;

if (!RegisterClass (&wndclass))
{
        MessageBox (  NULL, TEXT ("This program requi
                                szAppName, MB_IC
```

```
                return 0 ;
}

     hwnd = CreateWindow ( szAppName, TEXT ("PickFont: Cr
                    WS_OVERLAPPEDWINDOW | WS_CLIPCHILD
                    CW_USEDEFAULT, CW_USEDEFAULT,
                    CW_USEDEFAULT, CW_USEDEFAULT,
                   NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
{
          if (hdlg == 0 || !IsDialogMessage (hdlg, &msg))
          {
                    TranslateMessage (&msg) ;
                    DispatchMessage (&msg) ;
          }
```

```
        }
        return msg.wParam ;
}

LRESULT CALLBACK WndProc ( HWND hwnd, UINT message, WPA
{
        static DLGPARAMS dp ;
        static TCHAR                szText[] =    TEXT ("\x41\x42\x43
                TEXT ("\x61\x62\x63\x64\x65 ")

                TEXT ("\xC0\xC1\xC2\xC3\xC4\xC5 ")
                TEXT ("\xE0\xE1\xE2\xE3\xE4\xE5 ")
#ifdef UNICODE
                TEXT ("\x0390\x0391\x0392\x0393\x0394\x0395 "
                TEXT ("\x03B0\x03B1\x03B2\x03B3\x03B4\x03B5
                TEXT ("\x0410\x0411\x0412\x0413\x0414\x0415
                TEXT ("\x0430\x0431\x0432\x0433\x0434\x0435
                TEXT ("\x5000\x5001\x5002\x5003\x5004")
#endif
```

```
                    ;
    HDC                                     hdc ;
    PAINTSTRUCT                         ps ;
    RECT                                    rect ;

    switch (message)
    {
    case WM_CREATE:
                dp.iDevice = IDM_DEVICE_SCREEN ;
                hdlg = CreateDialogParam (((LPCREATESTRUCT)
       szAppName, hwnd, DlgProc, (LPARAM) &dp) ;
                return 0 ;
    case   WM_SETFOCUS:
                SetFocus (hdlg) ;
                return 0 ;

    case   WM_COMMAND:
           switch (LOWORD (wParam))
                {
```

```
                case   IDM_DEVICE_SCREEN:
                case   IDM_DEVICE_PRINTER:
                       CheckMenuItem (GetMenu (hwnd), dp.iDev
                       dp.iDevice = LOWORD (wParam) ;
                      CheckMenuItem (GetMenu (hwnd), dp.iDevi
                       SendMessage (hwnd, WM_COMMAND, IDOK
                      return 0 ;
                }
                break ;

     case   WM_PAINT:
                hdc = BeginPaint (hwnd, &ps) ;
                          // Set graphics mode so escapement w
     SetGraphicsMode (hdc, dp.fAdvGraphics ? GM_ADVANCED
                           // Set the mapping mode and the map
                MySetMapMode (hdc, dp.iMapMode) ;
                SetMapperFlags (hdc, dp.fMatchAspect) ;
```

```
                    // Find the point to begin drawing text

                GetClientRect (hdlg, &rect) ;
                rect.bottom += 1 ;
                DPtoLP (hdc, (PPOINT) &rect, 2) ;

                    // Create and select the font; display th

            SelectObject (hdc, CreateFontIndirect (&dp.lf)) ;
            TextOut (hdc, rect.left, rect.bottom, szText, lstrlen (sz

            DeleteObject (SelectObject (hdc, GetStockObject (SY
        EndPaint (hwnd, &ps) ;
                return 0 ;

    case   WM_DESTROY:
                PostQuitMessage (0) ;
                return 0 ;
    }
    return DefWindowProc (hwnd, message, wParam, lParam)
```

```
}

BOOL CALLBACK DlgProc (    HWND hdlg, UINT message, WPARA
{
        static DLGPARAMS            *      pdp ;
        static PRINTDLG                    pd = { sizeof (PRINTDLG) }
        HDC                                          hdcDevice ;
        HFONT                                       hFont ;


   switch (message)
        {
        case   WM_INITDIALOG:
                    // Save pointer to dialog-parameters structure in

                     pdp = (DLGPARAMS *) lParam ;

                     SendDlgItemMessage (hdlg,     IDC_LF_FACENAM
LF_FACESIZE - 1, 0) ;
      CheckRadioButton (hdlg,IDC_OUT_DEFAULT, IDC_OUT_OUTL
                                IDC_OUT_DEFAULT) ;
```

```
CheckRadioButton (hdlg,IDC_DEFAULT_QUALITY, IDC_PROO
                        IDC_DEFAULT_QUALITY) ;
CheckRadioButton (hdlg,IDC_DEFAULT_PITCH, IDC_VARIABL
                        IDC_DEFAULT_PITCH) ;
CheckRadioButton (hdlg,IDC_FF_DONTCARE, IDC_FF_DECOR
                        IDC_FF_DONTCARE) ;
CheckRadioButton (hdlg,IDC_MM_TEXT, IDC_MM_LOGTWIPS
                        IDC_MM_TEXT) ;
 SendMessage (hdlg, WM_COMMAND, IDOK, 0) ;
                                                // fall through
  case   WM_SETFOCUS:
            SetFocus (GetDlgItem (hdlg, IDC_LF_HEIGHT)) ;
            return FALSE ;

   case      WM_COMMAND:
            switch (LOWORD (wParam))
           {
            case   IDC_CHARSET_HELP:
                        MessageBox (  hdlg,
```

```
                TEXT   ("0         =      Ansi\n")
                TEXT   ("1         =      Default\n")
                TEXT   ("2         =      Symbol\n")
                TEXT   ("128 =      Shift JIS (Japanese)\n")
                TEXT   ("129 =      Hangul (Korean)\n")
                TEXT   ("130 =      Johab (Korean)\n")
                TEXT   ("134 =      GB 2312 (Simplified Chinese)\n")
                TEXT   ("136 =      Chinese Big 5 (Traditional Chinese
                TEXT   ("177 =      Hebrew\n")
                TEXT   ("178 =      Arabic\n")
                TEXT   ("161 =      Greek\n")
                TEXT   ("162 =      Turkish\n")
               TEXT   ("163 =      Vietnamese\n")
                TEXT   ("204 =      Russian\n")
                TEXT   ("222 =      Thai\n")
                TEXT   ("238 =      East European\n")
                TEXT   ("255 =      OEM"),
                  szAppName, MB_OK | MB_ICONINFORMATION) ;
                                    return TRUE ;
```

```
                    // These radio buttons set the lfO

                case   IDC_OUT_DEFAULT:
                        pdp->lf.lfOutPrecision = OUT_DEF
                        return TRUE ;

                case   IDC_OUT_STRING:
                        pdp->lf.lfOutPrecision = OUT_STR
                        return TRUE ;

                case   IDC_OUT_CHARACTER:
                        pdp->lf.lfOutPrecision = OUT_CHA
                       return TRUE ;

                case   IDC_OUT_STROKE:
                        pdp->lf.lfOutPrecision = OUT_STR
                        return TRUE ;

                case   IDC_OUT_TT:
                        pdp->lf.lfOutPrecision = OUT_TT_
```

```
                    return TRUE ;

            case   IDC_OUT_DEVICE:
                    pdp->lf.lfOutPrecision = OUT_DEV
                    return TRUE ;

            case   IDC_OUT_RASTER:
                    pdp->lf.lfOutPrecision = OUT_RAS
                    return TRUE ;

            case   IDC_OUT_TT_ONLY:
                    pdp->lf.lfOutPrecision = OUT_TT_
                    return TRUE ;

            case   IDC_OUT_OUTLINE:
                    pdp->lf.lfOutPrecision = OUT_OUT
                    return TRUE ;

                // These three radio buttons set the lfQ

            case   IDC_DEFAULT_QUALITY:
                    pdp->lf.lfQuality = DEFAULT_QUA
```

```
                    return TRUE ;

            case   IDC_DRAFT_QUALITY:
                        pdp->lf.lfQuality = DRAFT_QUALITY
                        return TRUE ;

            case   IDC_PROOF_QUALITY:
                        pdp->lf.lfQuality = PROOF_QUALI
                        return TRUE ;

                    // These three radio buttons set the low
                    //   of the lfPitchAndFamily field

            case   IDC_DEFAULT_PITCH:
                        pdp->lf.lfPitchAndFamily =
(0xF0 & pdp->lf.lfPitchAndFamily) | DEFAULT_PITCH ;
                        return TRUE ;

            case   IDC_FIXED_PITCH:
                        pdp->lf.lfPitchAndFamily =
(0xF0 & pdp->lf.lfPitchAndFamily) | FIXED_PITCH ;
```

```
                    return TRUE ;

            case   IDC_VARIABLE_PITCH:
                        pdp->lf.lfPitchAndFamily =
(0xF0 & pdp->lf.lfPitchAndFamily) | VARIABLE_PITCH
                        return TRUE ;

                        // These six radio buttons set the
                        //   of the lfPitchAndFamily field

           case   IDC_FF_DONTCARE:
                        pdp->lf.lfPitchAndFamily =
(0x0F & pdp->lf.lfPitchAndFamily) | FF_DONTCARE ;
                    return TRUE ;

            case   IDC_FF_ROMAN:
                        pdp->lf.lfPitchAndFamily =
(0x0F & pdp->lf.lfPitchAndFamily) | FF_ROMAN ;
                        return TRUE ;

            case   IDC_FF_SWISS:
```

```
                    pdp->lf.lfPitchAndFamily =
(0x0F & pdp->lf.lfPitchAndFamily) | FF_SWISS ;
                    return TRUE ;

    case   IDC_FF_MODERN:
                    pdp->lf.lfPitchAndFamily =
(0x0F & pdp->lf.lfPitchAndFamily) | FF_MODERN ;
                    return TRUE ;

    case   IDC_FF_SCRIPT:
                    pdp->lf.lfPitchAndFamily =
(0x0F & pdp->lf.lfPitchAndFamily) | FF_SCRIPT ;
                    return TRUE ;

    case   IDC_FF_DECORATIVE:
                    pdp->lf.lfPitchAndFamily =
(0x0F & pdp->lf.lfPitchAndFamily) | FF_DECORATIVE
                    return TRUE ;

                    // Mapping mode:
```

```
                case   IDC_MM_TEXT:
                case   IDC_MM_LOMETRIC:
                case   IDC_MM_HIMETRIC:
                case   IDC_MM_LOENGLISH:
                case   IDC_MM_HIENGLISH:
                case   IDC_MM_TWIPS:
                case   IDC_MM_LOGTWIPS:
                                pdp->iMapMode = LOWORD (wPa
                                return TRUE ;

                                // OK button pressed
                                // -----------------

            case   IDOK:
                                    // Get LOGFONT structure

                                SetLogFontFromFields (hdlg, pdp)

        // Set Match-Aspect and Advanced Graphics flags

        pdp->fMatchAspect = IsDlgButtonChecked      (hdlg
```

```
pdp->fAdvGraphics = IsDlgButtonChecked      (hdlg

      // Get Information Context

if (pdp->iDevice == IDM_DEVICE_SCREEN)
{
      hdcDevice = CreateIC (TEXT ("DISPLAY"), NULL
}
           else
         {
               pd.hwndOwner = hdlg ;
                pd.Flags = PD_RETURNDEFAULT | PD_R
                pd.hDevNames = NULL ;
                pd.hDevMode = NULL ;

                PrintDlg (&pd) ;

                hdcDevice = pd.hDC ;
     }
                // Set the mapping mode and the map
```

```
                MySetMapMode (hdcDevice, pdp->iMapMo
        SetMapperFlags (hdcDevice, pdp->fMatchAspect) ;

                    // Create font and select it into IC

                hFont = CreateFontIndirect (&pdp->lf) ;
                SelectObject (hdcDevice, hFont) ;

                    // Get the text metrics and face name

                GetTextMetrics (hdcDevice, &pdp->tm) ;
                GetTextFace (hdcDevice, LF_FULLFACESIZE
        DeleteDC (hdcDevice) ;
                DeleteObject (hFont) ;

                // Update dialog fields and invalidate main

                SetFieldsFromTextMetric (hdlg, pdp) ;
                InvalidateRect (GetParent (hdlg), NULL, TR
                return TRUE ;
        }
```

```
            break ;
        }
  return FALSE ;
}
void SetLogFontFromFields (HWND hdlg, DLGPARAMS * pdp)
{
        pdp->lf.lfHeight  =   GetDlgItemInt (hdlg, IDC_LF_HEIGHT,
        pdp->lf.lfWidth   =   GetDlgItemInt (hdlg, IDC_LF_WIDTH,
        pdp->lf.lfEscapement=GetDlgItemInt (hdlg, IDC_LF_ESCA
        pdp->lf.lfOrientation=GetDlgItemInt (hdlg,IDC_LF_ORIENT
       pdp->lf.lfWeight  =GetDlgItemInt (hdlg, IDC_LF_WEIGHT, N
        pdp->lf.lfCharSet =GetDlgItemInt (hdlg, IDC_LF_CHARSET
        pdp->lf.lfItalic    =IsDlgButtonChecked(hdlg,IDC_LF_ITALIC
        pdp->lf.lfUnderline =IsDlgButtonChecked (hdlg, IDC_LF_U
        pdp->lf.lfStrikeOut =IsDlgButtonChecked (hdlg, IDC_LF_ST
        GetDlgItemText (hdlg, IDC_LF_FACENAME, pdp->lf.lfFaceN
}

void SetFieldsFromTextMetric (HWND hdlg, DLGPARAMS * pdp)
```

```
{
        TCHAR               szBuffer [10] ;
        TCHAR *             szYes = TEXT ("Yes") ;
        TCHAR *             szNo   = TEXT ("No") ;
        TCHAR *             szFamily [] = {TEXT ("Don't Know"),
TEXT ("Roman"),
                TEXT ("Swiss"), TEXT ("Modern"),
                TEXT ("Script"), TEXT ("Decorative"),
                 TEXT ("Undefined") } ;

        SetDlgItemInt (hdlg, IDC_TM_HEIGHT, pdp->tm.tmHeight,
        SetDlgItemInt (hdlg, IDC_TM_ASCENT, pdp->tm.tmAscent
        SetDlgItemInt (hdlg, IDC_TM_DESCENT,pdp->tm.tmDesce
        SetDlgItemInt (hdlg, IDC_TM_INTLEAD,pdp->tm.tmInterna
        SetDlgItemInt (hdlg, IDC_TM_EXTLEAD,pdp->tm.tmExtern
        SetDlgItemInt (hdlg, IDC_TM_AVECHAR,pdp->tm.tmAveCh
        SetDlgItemInt (hdlg, IDC_TM_MAXCHAR,        pdp->tm.tm
        SetDlgItemInt (hdlg, IDC_TM_WEIGHT,         pdp->tm.tmW
        SetDlgItemInt (hdlg, IDC_TM_OVERHANG,pdp->tm.tmOve
```

```
        SetDlgItemInt (hdlg, IDC_TM_DIGASPX,        pdp->tm.tmD
        SetDlgItemInt (hdlg, IDC_TM_DIGASPY,        pdp->tm.tmD

        wsprintf (szBuffer, BCHARFORM, pdp->tm.tmFirstChar) ;
        SetDlgItemText (hdlg, IDC_TM_FIRSTCHAR, szBuffer) ;

        wsprintf (szBuffer, BCHARFORM, pdp->tm.tmLastChar) ;
        SetDlgItemText (hdlg, IDC_TM_LASTCHAR, szBuffer) ;

        wsprintf (szBuffer, BCHARFORM, pdp->tm.tmDefaultChar
        SetDlgItemText (hdlg, IDC_TM_DEFCHAR, szBuffer) ;

        wsprintf (szBuffer, BCHARFORM, pdp->tm.tmBreakChar)
        SetDlgItemText (hdlg, IDC_TM_BREAKCHAR, szBuffer) ;

        SetDlgItemText (hdlg, IDC_TM_ITALIC, pdp->tm.tmItalic
        SetDlgItemText (hdlg, IDC_TM_UNDER,      pdp->tm.tmUr
        SetDlgItemText (hdlg, IDC_TM_STRUCK, pdp->tm.tmStruc

        SetDlgItem    Text (hdlg, IDC_TM_VARIABLE,
                                  TMPF_FIXED_PITCH & pdp->tm.tm
```

```
        SetDlgItem    Text (hdlg, IDC_TM_VECTOR,
                                TMPF_VECTOR & pdp->tm.tmPitc

        SetDlgItem    Text (hdlg, IDC_TM_TRUETYPE,
                                TMPF_TRUETYPE & pdp->tm.tmPi

        SetDlgItem    Text (hdlg, IDC_TM_DEVICE,
                                TMPF_DEVICE & pdp->tm.tmPitch

        SetDlgItem    Text (hdlg, IDC_TM_FAMILY,
                                szFamily [min (6, pdp->tm.tmPitc

        SetDlgItemInt (hdlg, IDC_TM_CHARSET,       pdp->tm.tmC
        SetDlgItemText(hdlg, IDC_TM_FACENAME, pdp->szFaceNa
}

void MySetMapMode (HDC hdc, int iMapMode)
{
        switch (iMapMode)
        {
        case   IDC_MM_TEXT:         SetMapMode (hdc, MM_TEXT)
```

```
        case   IDC_MM_LOMETRIC:      SetMapMode (hdc, MM_LOM
        case   IDC_MM_HIMETRIC:      SetMapMode (hdc, MM_HIM
        case   IDC_MM_LOENGLISH:     SetMapMode (hdc, MM_LOI
        case   IDC_MM_HIENGLISH:     SetMapMode (hdc, MM_HIE
        case   IDC_MM_TWIPS:         SetMapMode (hdc, MM_TWIPS
        case   IDC_MM_LOGTWIPS:
                    SetMapMode (hdc, MM_ANISOTROPIC) ;
                    SetWindowExtEx (hdc, 1440, 1440, NULL) ;
                   SetViewportExtEx (hdc,GetDeviceCaps (hdc, LOG
              GetDeviceCaps (hdc, LOGPIXELSY), NULL) ;
                    break ;
        }
}
```

```
PICKFONT.RC
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
//////////////////////////////////////////////////////////////////////////////
```

```
// Dialog

PICKFONT DIALOG DISCARDABLE  0, 0, 348, 308

STYLE WS_CHILD | WS_VISIBLE | WS_BORDER

FONT 8, "MS Sans Serif"

BEGIN

   LTEXT                        "&Height:",IDC_STATIC,8,10,44,8

   EDITTEXT                   IDC_LF_HEIGHT,64,8,24,12,ES_AUTOHS

   LTEXT                           "&Width",IDC_STATIC,8,26,44,8

   EDITTEXT                     IDC_LF_WIDTH,64,24,24,12,ES_AU

   LTEXT                           "Escapement:",IDC_STATIC,8,42

   EDITTEXT                     IDC_LF_ESCAPE,64,40,24,12,ES_AU

   LTEXT                           "Orientation:",IDC_STATIC,8,58,

   EDITTEXT                     IDC_LF_ORIENT,64,56,24,12,ES_AU

   LTEXT                           "Weight:",IDC_STATIC,8,74,44,8

   EDITTEXT                     IDC_LF_WEIGHT,64,74,24,12,ES_AU

   GROUPBOX                      "Mapping Mode",IDC_STATIC,97,3

   CONTROL                  "Text",IDC_MM_TEXT,"Button",BS_AUTOR

                                          8

   CONTROL                       "Low Metric",IDC_MM_LOMETRIC,"
```

```
                                104,24,56,8
CONTROL              High Metric",IDC_MM_HIMETRIC,"Butto
                                BS_AUTORADIOBUTTON,104
CONTROL              "Low English",IDC_MM_LOENGLISH
                                BS_AUTORADIOBUTTON,104
CONTROL      "       High English",IDC_MM_HIENGLISH,"But
                               BS_AUTORADIOBUTTON,104,5
CONTROL              "Twips",IDC_MM_TWIPS,"Button",BS_AU
                                56,8
CONTROL              "Logical Twips",IDC_MM_LOGTWIP
                                BS_AUTORADIOBUTTON,104
CONTROL              "Italic",IDC_LF_ITALIC,"Button",BS_AUT
                                WS_TABSTOP,8,90,48,12
CONTROL            "Underline",IDC_LF_UNDER,"Button",BS_AU
                                WS_TABSTOP,8,104,48,12
CONTROL              "Strike Out",IDC_LF_STRIKE,"Butto
                                WS_TABSTOP,8,118,48,12
CONTROL              "Match Aspect",IDC_MATCH_ASPE
```

```
                                    WS_TABSTOP,60,104,62,8
CONTROL                    "Adv Grfx Mode",IDC_ADV_GRAPH
                                    BS_AUTOCHECKBOX | WS_TA
LTEXT                        "Character Set:",IDC_STATIC,8,1
EDITTEXT                   IDC_LF_CHARSET,58,135,24,12,ES_
PUSHBUTTON          "?",IDC_CHARSET_HELP,90,135,14,14
GROUPBOX                    "Quality",IDC_STATIC,132,98,62,4
CONTROL                    "Default",IDC_DEFAULT_QUALITY,"
                                    BS_AUTORADIOBUTTON,136
CONTROL                  "Draft",IDC_DRAFT_QUALITY,"Button",
                                    136,122,40,8
CONTROL                  "Proof",IDC_PROOF_QUALITY,"Button",
                                    136,134,40,8
LTEXT                        "Face Name:",IDC_STATIC,8,154
EDITTEXT                   IDC_LF_FACENAME,58,152,136,12,
GROUPBOX                    "Output Precision",IDC_STATIC,8,
CONTROL                    "OUT_DEFAULT_PRECIS",IDC_OUT_
                                    BS_AUTORADIOBUTTON,12,1
CONTROL                    "OUT_STRING_PRECIS",IDC_OUT_S
```

```
                                BS_AUTORADIOBUTTON,12,1
CONTROL                 "OUT_CHARACTER_PRECIS",IDC_OUT_C
                                BS_AUTORADIOBUTTON,12,2
CONTROL                   "OUT_STROKE_PRECIS",IDC_OUT_
                                BS_AUTORADIOBUTTON,12,2
CONTROL                 "OUT_TT_PRECIS",IDC_OUT_TT,"Buttor
                                12,230,112,8
CONTROL                  "OUT_DEVICE_PRECIS",IDC_OUT_D
                                BS_AUTORADIOBUTTON,12,2
CONTROL                 "OUT_RASTER_PRECIS",IDC_OUT_RAST
                                BS_AUTORADIOBUTTON,12,2
CONTROL                  "OUT_TT_ONLY_PRECIS",IDC_OUT_
                                BS_AUTORADIOBUTTON,12,2
CONTROL                  "OUT_OUTLINE_PRECIS",IDC_OUT_
                                BS_AUTORADIOBUTTON,12,2
GROUPBOX                  "Pitch",IDC_STATIC,132,166,62,5
CONTROL                "Default",IDC_DEFAULT_PITCH,"Button"
                                137,176,52,8
```

```
CONTROL                 "Fixed",IDC_FIXED_PITCH,"Button",BS_A
                                        189,52,8
CONTROL                 "Variable",IDC_VARIABLE_PITCH,"B
                                        BS_AUTORADIOBUTTON,137
GROUPBOX                "Family",IDC_STATIC,132,218,62,
CONTROL                 "Don't Care",IDC_FF_DONTCARE,"Button"
                                        137,229,52,8
CONTROL                 "Roman",IDC_FF_ROMAN,"Button",BS_AU
                                        52,8
CONTROL                 "Swiss",IDC_FF_SWISS,"Button",BS_AUTO
                                        52,8
CONTROL                 "Modern",IDC_FF_MODERN,"Button"
                                        265,52,8
CONTROL                 "Script",IDC_FF_SCRIPT,"Button",BS_A
                                        277,52,8
CONTROL                 "Decorative",IDC_FF_DECOR
                                        BS_AUTORADIOBUTTON,1
DEFPUSHBUTTON       "OK",IDOK,247,286,50,14
GROUPBOX                "Text Metrics",IDC_STATIC,20
```

```
LTEXT                    "Height:",IDC_STATIC,207,
LTEXT                    "0",IDC_TM_HEIGHT,281,1
LTEXT                    "Ascent:",IDC_STATIC,207,
LTEXT                    "0",IDC_TM_ASCENT,281,2
LTEXT                  "Descent:",IDC_STATIC,207,3
LTEXT                    "0",IDC_TM_DESCENT,281
LTEXT                    "Internal Leading:",IDC_ST
LTEXT                    "0",IDC_TM_INTLEAD,281,4
LTEXT                  "External Leading:",IDC_ST
LTEXT                    "0",IDC_TM_EXTLEAD,281,
LTEXT                    "Ave Char Width:",IDC_STA
LTEXT                "0",IDC_TM_AVECHAR,281,62,44
LTEXT                  "Max Char Width:",IDC_STA
LTEXT                    "0",IDC_TM_MAXCHAR,281
LTEXT                    "Weight:",IDC_STATIC,207,
LTEXT                    "0",IDC_TM_WEIGHT,281,8
LTEXT                  "Overhang:",IDC_STATIC,207
LTEXT                    "0",IDC_TM_OVERHANG,28
```

```
LTEXT                    "Digitized Aspect X:",IDC_S
LTEXT                    "0",IDC_TM_DIGASPX,281,
LTEXT                    "Digitized Aspect Y:",IDC_S
LTEXT                    "0",IDC_TM_DIGASPY,281,
LTEXT                  "First Char:",IDC_STATIC,207,122
LTEXT                  "0",IDC_TM_FIRSTCHAR,281,
LTEXT                    "Last Char:",IDC_STATIC,20
LTEXT                    "0",IDC_TM_LASTCHAR,28
LTEXT                    "Default Char:",IDC_STATIC
LTEXT                    "0",IDC_TM_DEFCHAR,281
LTEXT                    "Break Char:",IDC_STATIC,
LTEXT                  "0",IDC_TM_BREAKCHAR,281,
LTEXT                   "Italic?",IDC_STATIC,207,162
LTEXT                  "0",IDC_TM_ITALIC,281,162,44,
LTEXT                  "Underlined?",IDC_STATIC,207,1
LTEXT                  "0",IDC_TM_UNDER,281,172,44
LTEXT                  "Struck Out?",IDC_STATIC,207,1
LTEXT                  "0",IDC_TM_STRUCK,281,182,4
LTEXT                   "Variable Pitch?",IDC_STATIC,20
```

```
    LTEXT                                  "0",IDC_TM_VARIABLE,281,192,4
    LTEXT                                   "Vector Font?",IDC_STATIC,207,
    LTEXT                                   "0",IDC_TM_VECTOR,281,202,4
    LTEXT                                   "TrueType Font?",IDC_STATIC,20
    LTEXT                                   "0",IDC_TM_TRUETYPE,281,212
    LTEXT                                  "Device Font?",IDC_STATIC,207,
    LTEXT                                   "0",IDC_TM_DEVICE,281,222,44
    LTEXT                                   "Family:",IDC_STATIC,207,232,6
    LTEXT                                   "0",IDC_TM_FAMILY,281,232,44,
    LTEXT                                   "Character Set:",IDC_STATIC,20
    LTEXT                                   "0",IDC_TM_CHARSET,281,242,
    LTEXT                                   "0",IDC_TM_FACENAME,207,262
END

////////////////////////////////////////////////////////////////////////////
// Menu

PICKFONT MENU DISCARDABLE
BEGIN
  POPUP "&Device"
```

```
  BEGIN
      MENUITEM "&Screen",             IDM_DEVICE_SCREEN, CHEC
      MENUITEM "&Printer",                IDM_DEVICE_PRINTER
  END
END
```

RESOURCE.H

```
// Microsoft Developer Studio generated include file.
// Used by PickFont.rc
#define          IDC_LF_HEIGHT      1000
#define          IDC_LF_WIDTH       1001
#define          IDC_LF_ESCAPE      1002
#define          IDC_LF_ORIENT      1003
#define          IDC_LF_WEIGHT      1004
#define          IDC_MM_TEXT        1005
#define          IDC_MM_LOMETRIC    1006
#define          IDC_MM_HIMETRIC    1007
#define          IDC_MM_LOENGLISH   1008
#define          IDC_MM_HIENGLISH   1009
```

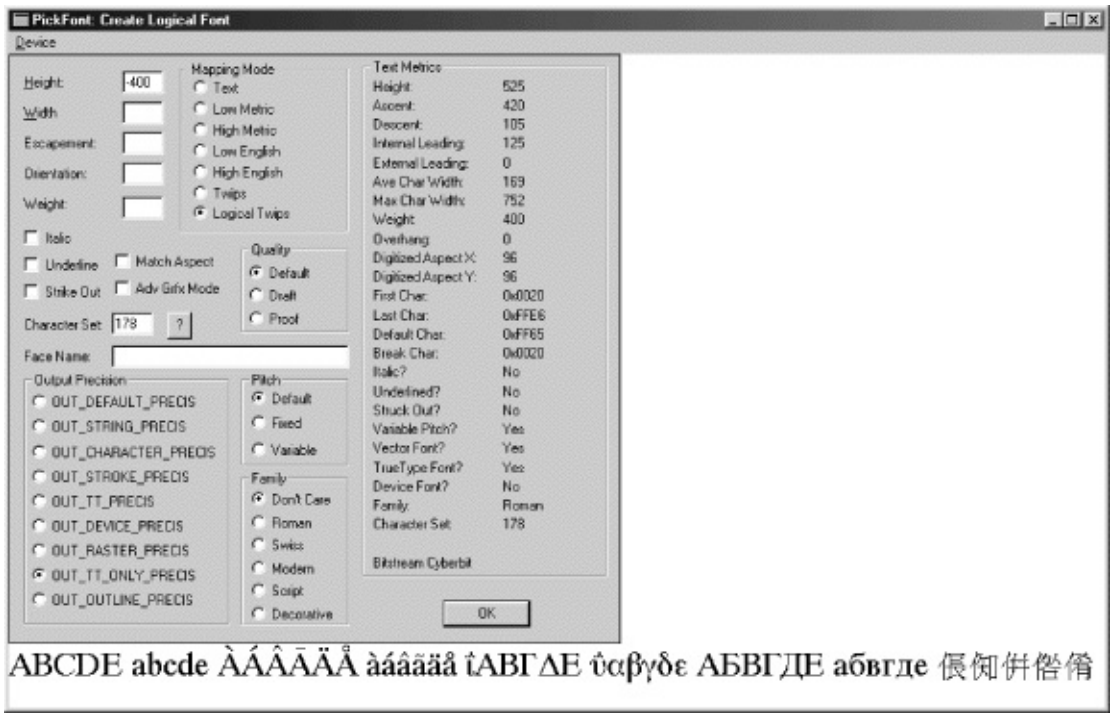
```
#define         IDC_MM_TWIPS      1010
#define         IDC_MM_LOGTWIPS    1011
#define         IDC_LF_ITALIC      1012
#define         IDC_LF_UNDER      1013
#define         IDC_LF_STRIKE      1014
#define         IDC_MATCH_ASPECT   1015
#define         IDC_ADV_GRAPHICS   1016
#define         IDC_LF_CHARSET     1017
#define         IDC_CHARSET_HELP   1018
#define         IDC_DEFAULT_QUALITY     1019
#define         IDC_DRAFT_QUALITY      1020
#define         IDC_PROOF_QUALITY      1021
#define         IDC_LF_FACENAME        1022
#define         IDC_OUT_DEFAULT        1023
#define         IDC_OUT_STRING         1024
#define         IDC_OUT_CHARACTER      1025
#define         IDC_OUT_STROKE         1026
#define         IDC_OUT_TT             1027
```

```
#define         IDC_OUT_DEVICE         1028
#define         IDC_OUT_RASTER         1029
#define         IDC_OUT_TT_ONLY        1030
#define         IDC_OUT_OUTLINE        1031
#define         IDC_DEFAULT_PITCH      1032
#define         IDC_FIXED_PITCH        1033
#define         IDC_VARIABLE_PITCH     1034
#define         IDC_FF_DONTCARE        1035
#define         IDC_FF_ROMAN           1036
#define         IDC_FF_SWISS           1037
#define         IDC_FF_MODERN          1038
#define         IDC_FF_SCRIPT          1039
#define         IDC_FF_DECORATIVE      1040
#define         IDC_TM_HEIGHT          1041
#define         IDC_TM_ASCENT          1042
#define         IDC_TM_DESCENT         1043
#define         IDC_TM_INTLEAD         1044
#define         IDC_TM_EXTLEAD         1045
#define         IDC_TM_AVECHAR         1046
```

```
#define         IDC_TM_MAXCHAR        1047
#define         IDC_TM_WEIGHT         1048
#define         IDC_TM_OVERHANG       1049
#define         IDC_TM_DIGASPX        1050
#define         IDC_TM_DIGASPY        1051
#define         IDC_TM_FIRSTCHAR      1052
#define         IDC_TM_LASTCHAR       1053
#define         IDC_TM_DEFCHAR        1054
#define         IDC_TM_BREAKCHAR      1055
#define         IDC_TM_ITALIC         1056
#define         IDC_TM_UNDER          1057
#define         IDC_TM_STRUCK         1058
#define         IDC_TM_VARIABLE       1059
#define         IDC_TM_VECTOR         1060
#define         IDC_TM_TRUETYPE       1061
#define         IDC_TM_DEVICE         1062
#define         IDC_TM_FAMILY         1063
#define         IDC_TM_CHARSET        1064
```

```
#define         IDC_TM_FACENAME       1065

#define         IDM_DEVICE_SCREEN     40001

#define         IDM_DEVICE_PRINTER     40002
```

17-1PICKFONTPICKFONTGetTextMetrics
1024×768



17-1　PICKFONTWindows NTUnicode

Logical　　TwipsMatch
Windows　　NT

DevicePICKFONTTEXTMETRIC
TEXTMETRIC

PICKFONT

CreateFont14LOGFONT

```
LOGFONT lf ;
```

CreateFontIndirect

```
hFont = CreatFontIndirect (&lf) ;
```

LOGFONT00CreateFontIndirect
LOGFONTWindows

LOGFONTPICKFONTEnterOK

LOGFONT

- **lfHeight**lfHeight0lfHeightlfHeight
  lfHeightWindowslfHeightlfHeight
  TEXTMETRICtmHeightlfHeighttmInternalLeading
  TEXTMETRICtmHeight

- **lfWidth**0WindowsTrueTypeTEXTMETRIC
  tmAveCharWidthlfWidthlfWidthLOGFONT0
  GetTextMetricstmAveCharWidthlfWidthtmAveCharWidth

lfEscapementlfOrientationTureType
Windows NTCM_ADVANCEDSetGraphicsModeAdv
PICKFONT

PICKFONT0-60030003600

- **lfEscapement**17-1

17-1

-

0	
900	
1800	
2700	

- Windows 98TrueTypeWindows NT SetGraphicsMode
- **lfOrientation**17-2

17-2

-

0	
900	90
1800	
2700	90

- Windows NTTrueTypeGM_ADVANCED,
- 10

- **lfWeight**WINGDI.H17-3

17-3

-

0	FW_DONTCARE
100	FW_THIN
200	FW_EXTRALIGHTFW_ULTRALIGHT
300	FW_LIGHT

400	FW_NORMALFW_REGULAR
500	FW_MEDIUM
600	FW_SEMIBOLDFW_DEMIBOLD
700	FW_BOLD
800	FW_EXTRABOLDFW_ULTRABOLD
900	FW_HEAVYFW_BLACK

- 0400700
- **lfItalic**WindowsGDIWindowsTrueTypeWindows

- **lfUnderline**GDIWindows
- **lfStrikeOut**GDI
- **lfCharSet**UnicodePICKFONT

lfCharSetANSI_CHARSETANSIDEFAULT_CHARSET1

- **lfOutPrecision**WindowsLOGFONT
  OUT_TT_ONLY_PRECISTrueType
- **lfClipPrecision**PICKFONT

- **lfQuality**WindowsTrueTypeDRAFT_QUALITYGDI PROOF_QUALITYPROOF_QUALITY DEFAULT_QUALITY0

- **lfPitchAndFamily**17-4

17-4

-

0	DEFAULT_PITCH
1	FIXED_PITCH
2	VARIABLE_PITCH

- 17-5

17-5

-

0x00	FW_DONTCARE
0x10	FF_ROMANserifs
0x20	FF_SWISSserifs
0x30	FF_MODERN
0x40	FF_SCRIPT
0x50	FF_DECORATIVE

- **lfFaceName**CourierArialTimes New RomanLF_FACES 32TrueTypelfFaceNameTimes Times New RomanlfItalic

CreateFontIndirectSelectObjectWindows

PICKFONT

- lfCharSetOEM_CHARSET(255)OEM TrueTypeBig Fonts TrueType TrueTypeOEM SYMBOL_CHARSET(2) SymbolWingdings

- lfPitchAndFamilyFIXED_PITCHWindows

- lfFaceNamelfFaceNameNULLlfPitchAndFamily FF_DONTCARE

- WindowslfHeightTrueTypeWindows
- lfQualityPROOF_QUALITYWindowsWindows
- lfHeightlfWeightWindowsTrueTypeMatch AspectPICKFONTPICKFONTTRUESetMapperFlags

PICKFONTGetTextMetricsPICKFONTDevice PICKFONTGetTextFace

WindowsTEXTMETRICTEXTMETRIC

- **tmHeight**LOGFONTlfHeightLOGFONTlfHeight tmHeighttmInternalLeadinglfHeight
- **tmAscent**
- **tmDescent**
- **tmInternalLeading**tmHeighttmHeighttmInternalLeading
- **tmExternalLeading tmHeight**
- **tmAveCharWidth**
- **tmMaxCharWidth**tmAveCharWidth
- **tmWeight**0999400700
- **tmOverhang**WindowstmAveCharWidthWindows tmAveCharWidthtmOverhangtmAveCharWidth
- **tmDigitizedAspectXtmDigitizedAspectY**LOGPIXELSX LOGPIXELSYGetDeviceCaps

- **tmFirstChar**
- **tmLastChar**TEXTMETRICGetTextMetricsW255
- **tmDefaultChar**Windows
- **tmBreakChar**WindowsEBCDIC32
- **tmItalic**
- **tmUnderlined**
- **tmStruckOut**
- **tmPitchAndFamily**WINGDI.H17-6

17-6

-

0x01	TMPF_FIXED_PITCH
0x02	TMPF_VECTOR
0x04	TMPF_TRUETYPE
0x08	TMPF_DEVICE

- TMPF_FIXED_PITCH1TMPF_VECTORTrueType PostScript1TMPF_DEVICE)GDI
- LOGFONTlfPitchAndFamily

- **tmCharSet**

Unicode

WindowsLOGFONTTEXTMETRIC0255WINGDI.H

```
#define         ANSI_CHARSET                0
#define         DEFAULT_CHARSET             1
#define         SYMBOL_CHARSET              2
#define         MAC_CHARSET                 77
#define         SHIFTJIS_CHARSET            128
#define         HANGEUL_CHARSET             129
#define         HANGUL_CHARSET              129
#define         JOHAB_CHARSET               130
#define         GB2312_CHARSET              134
#define         CHINESEBIG5_CHARSET         136
#define         GREEK_CHARSET               161
#define         TURKISH_CHARSET             162
#define         VIETNAMESE_CHARSET          163
```

```
#define     HEBREW_CHARSET          177
#define     ARABIC_CHARSET          178
#define     BALTIC_CHARSET          186
#define     RUSSIAN_CHARSET         204
#define     THAI_CHARSET            222
#define     EASTEUROPE_CHARSET      238
#define     OEM_CHARSET             255
```

Windows255

UNICODEPICKFONTUNICODEDEBUGRELEASE

UnicodePICKFONT0x400x450x600x65
SYMBOL_CHARSETAEae

PICKFONTUnicode120xC00xC50xE00xE5ANSI_CHARSETA
GREEK_CHARSETRUSSIAN_CHARSETTrueType
TrueTypeBig fontsWindows

Windows NTPICKFONTUnicode0xC00xC50xE00xE5A
SYMBOL_CHARSETUnicode0x03900x03950x03B00x03B5
Unicode0x04100x04150x04300x0435
GREEK_CHARSETRUSSIAN_CHARSETLOGFONTIDUnicode
ID

HEBREW_CHARSET177WindowsBig

PICKFONT0x50000x5004WindowsLucida
UnicodeBitstream CyberBit
http://www.bitstream.com/products/world/cyberbitsLucida Sans Unicoc

300KBitstream CyberBit13MLucida Sans SHIFTJIS_CHARSET ()HANGUL_CHARSET JOHAB_CHARSETGB2312_CHARSET CHINESEBIG5_CHARSET

Unicode

EZFONT

TrueTypeWindowsWindowsChooseFont

TrueTypeTrueTypeTrueType
13EZFONT17-2

17-2 EZFONT

```
EZFONT.H

/*----------------------------------------------------------------------

  EZFONT.H header file

----------------------------------------------------------------------*/

HFONT EzCreateFont (HDC hdc, TCHAR * szFaceName, int iDeciP
                    int iDeciPtWidth, int iAttributes, BOOL fLogRes) ;

#define          EZ_ATTR_BOLD                                  1

#define          EZ_ATTR_ITALIC                               2

#define          EZ_ATTR_UNDERLINE                             4

#define          EZ_ATTR_STRIKEOUT                             8
```

EZFONT.C

```
/*------------------------------------------------------------------------
  EZFONT.C --     Easy Font Creation
                                        (c) Charles Petzold, 1998
------------------------------------------------------------------------*/
#include <windows.h>
#include <math.h>
#include "ezfont.h"

HFONT EzCreateFont (HDC hdc, TCHAR * szFaceName, int iDeciP
                int iDeciPtWidth, int iAttributes, BOOL fLogRes)
{
          FLOAT                cxDpi, cyDpi ;
          HFONT                 hFont ;
          LOGFONT             lf ;
          POINT                pt ;
          TEXTMETRIC          tm ;
```

```
SaveDC (hdc) ;
  SetGraphicsMode (hdc, GM_ADVANCED) ;
       ModifyWorldTransform         (hdc, NULL, MWT_IDENTITY) ;
        SetViewportOrgEx                  (hdc, 0, 0, NULL) ;
        SetWindowOrgEx                          (hdc, 0, 0, NULL) ;

        if (fLogRes)
        {
             cxDpi = (FLOAT) GetDeviceCaps (hdc, LOGPIXELSX) ;
             cyDpi = (FLOAT) GetDeviceCaps (hdc, LOGPIXELSY) ;
        }
        else
        {
             cxDpi = (FLOAT) (25.4 * GetDeviceCaps (hdc, HORZR
                        GetDeviceCaps (hdc, HORZSIZE)) ;
             cyDpi = (FLOAT) (25.4 * GetDeviceCaps (hdc, VERTRI
                        GetDeviceCaps (hdc, VERTSIZE)) ;
  }
```

```
            pt.x = (int) (iDeciPtWidth    * cxDpi / 72) ;
            pt.y = (int) (iDeciPtHeight   * cyDpi / 72) ;

            DPtoLP (hdc, &pt, 1) ;
            lf.lfHeight                 = - (int) (fabs (pt.y) / 10.0 -
            lf.lfWidth                  = 0 ;
            lf.lfEscapement               = 0 ;
            lf.lfOrientation              = 0 ;
            lf.lfWeight                 = iAttributes & EZ_ATTR_B
      lf.lfItalic                 = iAttributes & EZ_ATTR_ITALIC
      lf.lfUnderline                    = iAttributes & EZ_ATTR_
      lf.lfStrikeOut                = iAttributes & EZ_ATTR_STRIK
            lf.lfCharSet                = DEFAULT_CHARSET ;
            lf.lfOutPrecision             = 0 ;
            lf.lfClipPrecision            = 0 ;
            lf.lfQuality                = 0 ;
lf.lfPitchAndFamily             = 0 ;
```

```
            lstrcpy (lf.lfFaceName, szFaceName) ;

            hFont = CreateFontIndirect (&lf) ;

    if (iDeciPtWidth != 0)
            {
                    hFont = (HFONT) SelectObject (hdc, hFont) ;
                    GetTextMetrics (hdc, &tm) ;
                    DeleteObject (SelectObject (hdc, hFont)) ;
                    lf.lfWidth = (int) (tm.tmAveCharWidth *
      fabs (pt.x) / fabs (pt.y) + 0.5) ;
                    hFont = CreateFontIndirect (&lf) ;
            }

            RestoreDC (hdc, -1) ;
            return hFont ;
}
```

EZFONT.CEzCreateFont

```
hFont = EzCreateFont (     hdc, szFaceName, iDeciPtHeight, iDec
                                                    iAttributes, fLogR
```

SelectObjectGetTextMetricsGetOutlineTextMetricsDeleteObject

szFaceNameTrueType

125

TrueTypeem-widthMem-
square

iAttributesEZFONT.H

```
EZ_ATTR_BOLD

EZ_ATTR_ITALIC

EZ_ATTR_UNDERLINE

EZ_ATTR_STRIKEOUT
```

EZ_ATTR_BOLDEZ_ATTR_ITALICTrueType

fLogResTRUEGetDeviceCapsLOGPIXELSX
LOGPIXELSYHORZRESHORZSIZEVERTRESVERTSIZE
Windows NT

EzCreateFontWindows NTSetGraphicsModeModifyWorldT
Windows 98Windows NT

EzCreateFontLOGFONTCreateFontIndirectCreateFontIndirect
EzCreateFontLOGFONTlfHeightGetDeviceCaps
DPtoLPDPtoLPEzCreateFont

17-3EZTESTEZFONTEZTESTFONTDEMO

17-3　EZTEST

```
EZTEST.C

/*------------------------------------------------------------------------
  EZTEST.C --   Test of EZFONT
                                                (c) Charles Petzold, 1998
------------------------------------------------------------------------*/

#include <windows.h>
#include "ezfont.h"

TCHAR szAppName    [] = TEXT ("EZTest") ;
TCHAR szTitle      [] = TEXT ("EZTest: Test of EZFONT") ;

void PaintRoutine (HWND hwnd, HDC hdc, int cxArea, int cyArea
{
          HFONT                hFont ;
          int                  y, iPointSize ;
          LOGFONT              lf ;
          TCHAR                szBuffer [100] ;
```

```
     TEXTMETRIC          tm ;

          // Set Logical Twips mapping mode

     SetMapMode (hdc, MM_ANISOTROPIC) ;
     SetWindowExtEx (hdc, 1440, 1440, NULL) ;
     SetViewportExtEx (hdc,       GetDeviceCaps (hdc, LOGPIX
GetDeviceCaps      (hdc, LOGPIXELSY), NULL) ;

          // Try some fonts

     y = 0 ;
     for (iPointSize = 80 ; iPointSize <= 120 ; iPointSize++)
     {
          hFont = EzCreateFont (hdc, TEXT ("Times New Roma
                                iPointSize, 0, 0, TRUE) ;

          GetObject (hFont, sizeof (LOGFONT), &lf) ;

          SelectObject (hdc, hFont) ;
          GetTextMetrics (hdc, &tm) ;
          TextOut (hdc, 0, y, szBuffer,
```

```
                    wsprintf (    szBuffer,
            TEXT ("Times New Roman font of %i.%i points, ")
           TEXT ("lf.lfHeight = %i, tm.tmHeight = %i"),
            iPointSize / 10, iPointSize % 10,
            lf.lfHeight, tm.tmHeight)) ;

              DeleteObject (SelectObject (hdc, GetStockObject (SY
              y += tm.tmHeight ;
        }
}
```

FONTDEMO.C

```
/*------------------------------------------------------------------------
  FONTDEMO.C --       Font Demonstration Shell Program
                                                    (c) Charles Petzold, 199
------------------------------------------------------------------------*/
#include <windows.h>
#include "..\\EZTest\\EzFont.h"
#include "..\\EZTest\\resource.h"
```

```
extern      void                    PaintRoutine (HWND, HDC, int, int
LRESULT     CALLBACK                WndProc (HWND, UINT, WPARAM,

HINSTANCE hInst ;

extern TCHAR szAppName [] ;
extern TCHAR szTitle [] ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                                PSTR szCmdLine, i
{
        TCHAR                   szResource [] = TEXT ("FontDe
        HWND                    hwnd ;
        MSG                     msg ;
        WNDCLASS                wndclass ;

        hInst = hInstance ;
        wndclass.style                  = CS_HREDRAW | CS_VR
        wndclass.lpfnWndProc                = WndProc ;
        wndclass.cbClsExtra                 = 0 ;
```

```
    wndclass.cbWndExtra                    = 0 ;
    wndclass.hInstance                     = hInstance ;
    wndclass.hIcon                         = LoadIcon (NULL, IDI_
    wndclass.hCursor                       = LoadCursor (NULL,
    wndclass.hbrBackground                 = (HBRUSH) GetStockO
    wndclass.lpszMenuName                  = szResource ;
    wndclass.lpszClassName                 = szAppName ;

    if (!RegisterClass (&wndclass))
    {
          MessageBox (  NULL, TEXT ("This program requi
                                          szAppName, M
          return 0 ;
    }

    hwnd = CreateWindow ( szAppName, szTitle,
          WS_OVERLAPPEDWINDOW,
          CW_USEDEFAULT, CW_USEDEFAULT,
```

```
                    CW_USEDEFAULT, CW_USEDEFAULT,
                    NULL, NULL, hInstance, NULL) ;

          ShowWindow (hwnd, iCmdShow) ;
          UpdateWindow (hwnd) ;

          while (GetMessage (&msg, NULL, 0, 0))
          {
               TranslateMessage (&msg) ;
               DispatchMessage (&msg) ;
          }
     return msg.wParam ;
}

LRESULT CALLBACK WndProc ( HWND hwnd, UINT message, WPA
{
          static DOCINFO              di = { sizeof (DOCINFO), TEXT (
          static int                  cxClient, cyClient ;
          static PRINTDLG       pd     = { sizeof (PRINTDLG) } ;
```

```
     BOOL                                    fSuccess ;
     HDC                                     hdc, hdcPrn ;
     int                                     cxPage, cyPage ;
     PAINTSTRUCT             ps ;

     switch (message)
     {
     case   WM_COMMAND:
               switch (wParam)
          {
               case   IDM_PRINT:
                                        // Get printer DC
                                   pd.hwndOwner = hwnd ;
                                   pd.Flags     = PD_RETURNDC | PD
                                   if (!  PrintDlg (&pd))
                                             return 0 ;
                                   if (NULL == (hdcPrn = pd.hDC))
```

```
                    {
MessageBox(    hwnd, TEXT ("Cannot obtain Printer I
          szAppName, MB_ICONEXCLAMATION | MB_OK
                    return 0 ;
                    }
                    // Get size of printable area o

                    cxPage = GetDeviceCaps (hdcPrn
                    cyPage = GetDeviceCaps (hdcPrn

                    fSuccess = FALSE ;

                    // Do the printer page

                   SetCursor (LoadCursor (NULL, IDC_
                    ShowCursor (TRUE) ;

                    if ((StartDoc (hdcPrn, &di) > 0) &&
                {
     PaintRoutine (hwnd, hdcPrn, cxPage, cyPage) ;
```

```
                    if (EndPage (hdcPrn) > 0)
                    {
                                        fSuccess = TRUE ;
                                        EndDoc (hdcPrn) ;
                    }
               }
               DeleteDC (hdcPrn) ;

               ShowCursor (FALSE) ;
               SetCursor (LoadCursor (NULL, IDC_ARR

               if (!fSuccess)
                              MessageBox (hwnd,
     TEXT ("Error encountered during printing"),
      szAppName, MB_ICONEXCLAMATION | MB_OK) ;
               return 0 ;

          case   IDM_ABOUT:
               MessageBox (  hwnd, TEXT ("Font Dem
       TEXT ("(c) Charles Petzold, 1998"),
```

```
        szAppName, MB_ICONINFORMATION | MB_OK) ;
                        return 0 ;
        }
        break ;

    case   WM_SIZE:
                cxClient = LOWORD (lParam) ;
                cyClient = HIWORD (lParam) ;
                return 0 ;

    case   WM_PAINT:
                hdc = BeginPaint (hwnd, &ps) ;

                PaintRoutine (hwnd, hdc, cxClient, cyClient) ;

                EndPaint (hwnd, &ps) ;
                return 0 ;
```

```
        case   WM_DESTROY :
                    PostQuitMessage (0) ;
                    return 0 ;
   }
        return DefWindowProc (hwnd, message, wParam, lParam)
}
```

```
FONTDEMO.RC
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
//////////////////////////////////////////////////////////////////////////////
// Menu
FONTDEMO MENU DISCARDABLE
BEGIN
   POPUP "&File"
   BEGIN
              MENUITEM "&Print...",                                    IDM
   END
```

```
  POPUP "&Help"
  BEGIN
            MENUITEM "&About...",                                   IDM
  END
END
```

```
RESOURCE.H
// Microsoft Developer Studio generated include file.
// Used by FontDemo.rc
#define IDM_PRINT       40001
#define IDM_ABOUT       40002
```

EZTEST.CPaintRoutineLogical Twips8120.1Times New TEXTMETRICFONTDEMO

PICKFONTLOGFONTlfOrientationlfEscapementTrueTypeGDI

EzCreateFontFONTROT17-4FONTROT.C EZFONTFONTDEMO

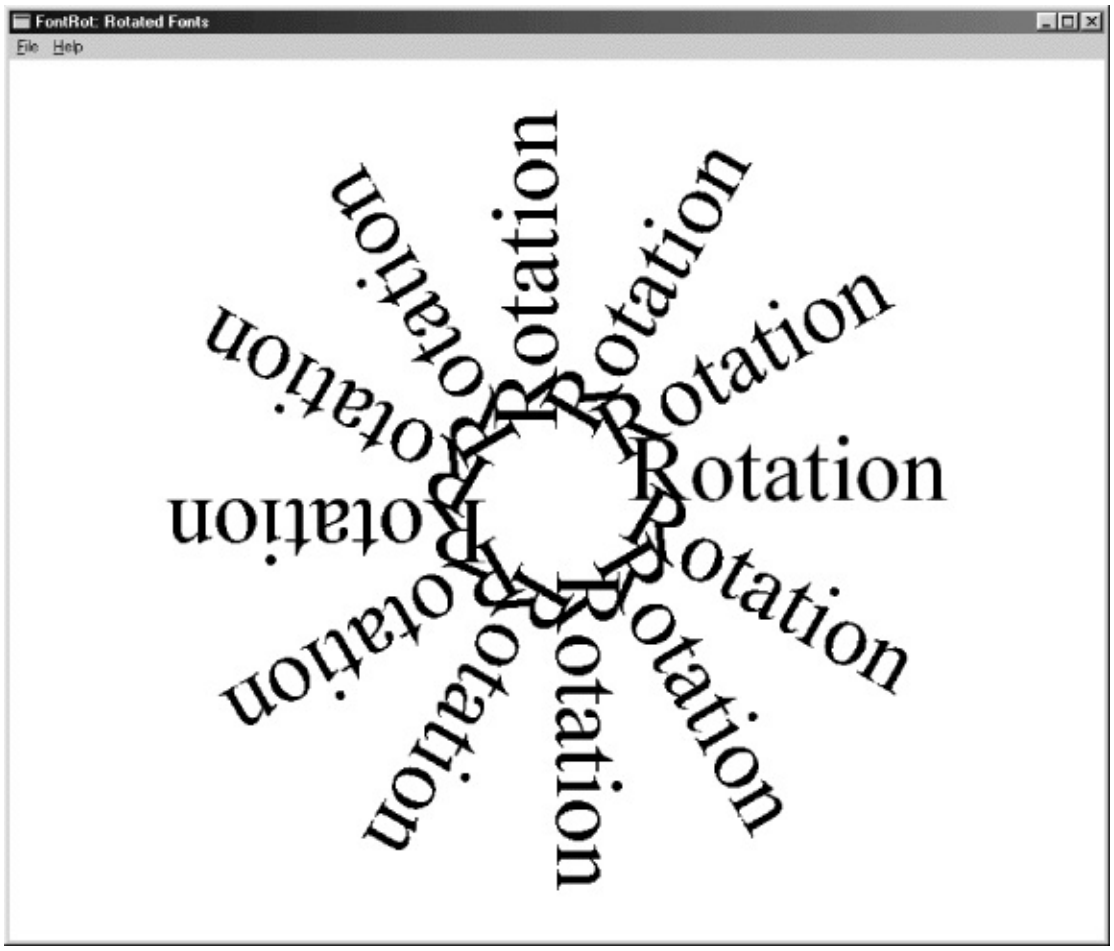
17-4 FONTROT

```
FONTROT.C
```

```
/*------------------------------------------------------------------------
  FONTROT.C --  Rotated Fonts
                                                (c) Charles Petzold, 1998
------------------------------------------------------------------------*/
#include <windows.h>
#include "..\\eztest\\ezfont.h"
TCHAR szAppName    [] = TEXT ("FontRot") ;
TCHAR szTitle      [] = TEXT ("FontRot: Rotated Fonts") ;

void PaintRoutine (HWND hwnd, HDC hdc, int cxArea, int cyArea
{
         static TCHAR szString [] = TEXT ("   Rotation") ;
          HFONT                          hFont ;
          int                            i ;
          LOGFONT                        lf ;

          hFont = EzCreateFont (hdc, TEXT ("Times New Roman"), 
          GetObject (hFont, sizeof (LOGFONT), &lf) ;
  DeleteObject (hFont) ;
```

```
        SetBkMode (hdc, TRANSPARENT) ;
        SetTextAlign (hdc, TA_BASELINE) ;
        SetViewportOrgEx (hdc, cxArea / 2, cyArea / 2, NULL) ;

        for (i = 0 ; i < 12 ; i ++)
        {
                lf.lfEscapement = lf.lfOrientation = i * 300 ;
                SelectObject (hdc, CreateFontIndirect (&lf)) ;

                TextOut (hdc, 0, 0, szString, lstrlen (szString)) ;
                DeleteObject (SelectObject (hdc, GetStockObjec
        }
}
```

FONTROTEzCreateFont54Times New RomanLOGFONTfor
17-2



17-2 FONTROT

Windows NTX

GDIChooseFontChooseFont

WindowsEnumFonts

```
EnumFonts (hdc, szTypeFace, EnumProc, pData) ;
```

NULLcallbackGDIcallbackLOGFONTTEXTMETRIC

EnumFontFamiliesWindows 3.1TrueType

```
EnumFontFamilies (hdc, szFaceName, EnumProc, pData) ;
```

EnumFontFamiliesNULLTimes
callbackEnumFontFamiliesGDITimes
TrueTypecallbackENUMLOGFONTLOGFONT
ItalicBoldTEXTMETRICTrueTypeNEWTEXTMETRIC
NEWTEXTMETRICTEXTMETRIC

EnumFontFamiliesExWindows32

```
EnumFontFamiliesEx (hdc, &logfont, EnumProc, pData, dwFlags)
```

LOGFONTlfCharSetlfFaceNameCallbackENUMLOGFONTEX
NEWTEXTMETRICEX

ChooseFont

ChooseFontChooseFontChooseFontCHOOSEFONT
ChooseFontLOGFONTCHOOSEFONT

17-5CHOSFONTChooseFontLOGFONTPICKFONT

17-5 CHOSFONT

```
CHOSFONT.C

/*------------------------------------------------------------------------

  CHOSFONT.C -- ChooseFont Demo

                                        (c) Charles Petzold, 1998
```

```
------------------------------------------------------------------------*/
#include <windows.h>
#include "resource.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                          PSTR szCmdLine, int iCm
{
   static TCHAR szAppName[] = TEXT ("ChosFont") ;
         HWND                        hwnd ;
         MSG                        msg ;
         WNDCLASS                   wndclass ;

         wndclass.style                    = CS_HREDRAW | CS_V
         wndclass.lpfnWndProc                = WndProc ;
         wndclass.cbClsExtra                 = 0 ;
         wndclass.cbWndExtra                 = 0 ;
         wndclass.hInstance                  = hInstance ;
         wndclass.hIcon                      = LoadIcon (NULL, IDI_
```

```
    wndclass.hCursor                        = LoadCursor (NULL,
    wndclass.hbrBackground                  = (HBRUSH) GetStockO
    wndclass.lpszMenuName                   = szAppName ;
    wndclass.lpszClassName                  = szAppName ;

    if (!RegisterClass (&wndclass))
    {
          MessageBox (  NULL, TEXT ("This program requires W
                                          szAppName, MB_ICONE
          return 0 ;
    }

    hwnd = CreateWindow ( szAppName, TEXT ("ChooseFont
                WS_OVERLAPPEDWINDOW,
                CW_USEDEFAULT, CW_USEDEFAULT,
                CW_USEDEFAULT, CW_USEDEFAULT,
               NULL, NULL, hInstance, NULL) ;
```

```
        ShowWindow (hwnd, iCmdShow) ;
        UpdateWindow (hwnd) ;
        while (GetMessage (&msg, NULL, 0, 0))
        {
             TranslateMessage (&msg) ;
             DispatchMessage (&msg) ;
   }
  return msg.wParam ;
}

LRESULT CALLBACK WndProc ( HWND hwnd, UINT message, WPA
{
        static CHOOSEFONT           cf ;
        static int                  cyChar ;
        static LOGFONT              if ;
        static TCHAR  szText[] = TEXT ("\x41\x42\x43\x44\x45 ")
                        TEXT ("\x61\x62\x63\x64\x65 ")
                         TEXT ("\xC0\xC1\xC2\xC3\xC4\xC5 ")
                         TEXT ("\xE0\xE1\xE2\xE3\xE4\xE5 ")
```

```
#ifdef UNICODE
     TEXT ("\x0390\x0391\x0392\x0393\x0394\x0395 ")
    TEXT ("\x03B0\x03B1\x03B2\x03B3\x03B4\x03B5 ")
   TEXT ("\x0410\x0411\x0412\x0413\x0414\x0415 ")
     TEXT ("\x0430\x0431\x0432\x0433\x0434\x0435 ")
     TEXT ("\x5000\x5001\x5002\x5003\x5004")
#endif
                                ;
          HDC                                hdc ;
          int                               y ;
          PAINTSTRUCT                             ps ;
          TCHAR                              szBuffer [64] ;
          TEXTMETRIC                              tm ;

          switch (message)
          {
          case WM_CREATE:
                                // Get text height
                    cyChar = HIWORD (GetDialogBaseUnits ()) ;
```

```
                    // Initialize the LOGFONT structure
          GetObject (GetStockObject (SYSTEM_FONT), siz
                    // Initialize the CHOOSEFONT structure
     cf.lStructSize                = sizeof (CHOOSEFONT)
          cf.hwndOwner         = hwnd ;
          cf.hDC               = NULL ;
          cf.lpLogFont         = &lf ;
          cf.iPointSize        = 0 ;
          cf.Flags             = CF_INITTOLOGFONTSTRUCT |
                               CF_SCREENFONTS | CF_EFFECTS
          cf.rgbColors         = 0 ;
          cf.lCustData         = 0 ;
          cf.lpfnHook          = NULL ;
          cf.lpTemplateName    = NULL ;
          cf.hInstance         = NULL ;
          cf.lpszStyle         = NULL ;
          cf.nFontType         = 0 ;
          cf.nSizeMin          = 0 ;
```

```
                cf.nSizeMax          = 0 ;
                return 0 ;

        case   WM_COMMAND:
                switch (LOWORD (wParam))
                {
                case   IDM_FONT:
                                if (ChooseFont (&cf))
                      InvalidateRect (hwnd, NULL, TRUE) ;
                                 return 0 ;
                }
                return 0 ;

        case   WM_PAINT:
                hdc = BeginPaint (hwnd, &ps) ;

                                // Display sample text using selec

                SelectObject (hdc, CreateFontIndirect (&lf)) ;
                GetTextMetrics (hdc, &tm) ;
                SetTextColor (hdc, cf.rgbColors) ;
```

```
                TextOut (hdc, 0, y = tm.tmExternalLeading, szTe

                                // Display LOGFONT structure field

                DeleteObject         (SelectObject (hdc, GetStock

                SetTextColor (hdc, 0) ;


                TextOut (hdc, 0, y += tm.tmHeight, szBuffer,
wsprintf (szBuffer, TEXT ("lfHeight = %i"), lf.lfHeight)) ;

                TextOut (hdc, 0, y += cyChar, szBuffer,
wsprintf (szBuffer, TEXT ("lfWidth = %i"), lf.lfWidth)) ;


                TextOut (hdc, 0, y += cyChar, szBuffer,
wsprintf ( szBuffer, TEXT ("lfEscapement = %i"),
                        lf.lfEscapement)) ;


                TextOut (hdc, 0, y += cyChar, szBuffer,
wsprintf ( szBuffer, TEXT ("lfOrientation = %i"),
                                lf.lfOrientation)) ;
```

```
                TextOut (hdc, 0, y += cyChar, szBuffer,
wsprintf (szBuffer, TEXT ("lfWeight = %i"),lf.lfWeight)) ;

                TextOut (hdc, 0, y += cyChar, szBuffer,
wsprintf (szBuffer, TEXT ("lfItalic = %i"),lf.lfItalic)) ;

                TextOut (hdc, 0, y += cyChar, szBuffer,
wsprintf (szBuffer, TEXT ("lfUnderline = %i"),lf.lfUnderline))

                TextOut (hdc, 0, y += cyChar, szBuffer,
wsprintf (szBuffer, TEXT ("lfStrikeOut = %i"),lf.lfStrikeOut)) ;

                TextOut (hdc, 0, y += cyChar, szBuffer,
wsprintf (szBuffer, TEXT ("lfCharSet = %i"),lf.lfCharSet)) ;

                TextOut (hdc, 0, y += cyChar, szBuffer,
wsprintf ( szBuffer, TEXT ("lfOutPrecision = %i"),
          lf.lfOutPrecision)) ;

                TextOut (hdc, 0, y += cyChar, szBuffer,
wsprintf (szBuffer, TEXT ("lfClipPrecision = %i"),
```

```
            lf.lfClipPrecision)) ;

                    TextOut (hdc, 0, y += cyChar, szBuffer,
      wsprintf ( szBuffer, TEXT ("lfQuality = %i"),lf.lfQuality)) ;

                    TextOut (hdc, 0, y += cyChar, szBuffer,
      wsprintf ( szBuffer, TEXT ("lfPitchAndFamily = 0x%02X"),
                lf.lfPitchAndFamily)) ;

                    TextOut (hdc, 0, y += cyChar, szBuffer,
      wsprintf ( szBuffer, TEXT ("lfFaceName = %s"),lf.lfFaceName

                    EndPaint (hwnd, &ps) ;
                    return 0 ;
         case   WM_DESTROY:
                    PostQuitMessage (0) ;
                    return 0 ;
         }
         return DefWindowProc (hwnd, message, wParam, lParam)
}
```

```
CHOSFONT.RC

//Microsoft Developer Studio generated resource script.

#include "resource.h"

#include "afxres.h"

//////////////////////////////////////////////////////////////////////////////

// Menu

CHOSFONT MENU DISCARDABLE

BEGIN

   MENUITEM "&Font!",                                                    IDM

END
```

```
RESOURCE.H

// Microsoft Developer Studio generated include file.

// Used by ChosFont.rc

#define IDM_FONT     40001
```

CHOOSEFONTFlagsCHOSFONTCF_INITLOGFONTSTRUCT
WindowsChooseFontLOGFONTTrueTypeCF_TTONLY
CF_FIXEDPITCHONLYCF_SCRIPTSONLY
CF_SCREENFONTSCF_PRINTERFONTSCF_BOTH
CHOOSEFONThDCCHOSFONTCF_SCREENFONTS

CF_EFFECTSCHOSFONT

FontChooseFontScriptIDLOGFONT

ChooseFont…fHeightGetDeviceCapsLOGPIXELSY
96ChooseFont72Times Roman1…hooseFontLOGFONTlfHeigl
-96961… &#65533;/p>

:

- Windows NTLogical
- MM_TEXT
- ChooseFontLOGFONTlfHeightlfHeightChooseFont
  CHOOSEFONThDClfHeight

CHOOSEFONTiPointSizelfHeightEZFONT.C

ChooseFontUNICHARSLucida
UNICHARSTextOutWWindows NTW

17-6 UNICHARS

```
UNICHARS.C

/*------------------------------------------------------------------------
  UNICHARS.C -- Displays 16-bit character codes
                                        (c) Charles Petzold, 1998
------------------------------------------------------------------------*/

#include <windows.h>

#include "resource.h"
```

```
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                          PSTR szCmdLine, int iCr
{
          static TCHAR szAppName[] = TEXT ("UniChars") ;
          HWND                         hwnd ;
          MSG                          msg ;
          WNDCLASS                     wndclass ;

          wndclass.style                    = CS_HREDRAW | CS_V
          wndclass.lpfnWndProc                = WndProc ;
          wndclass.cbClsExtra                 = 0 ;
          wndclass.cbWndExtra                 = 0 ;
          wndclass.hInstance                  = hInstance ;
          wndclass.hIcon                    = LoadIcon (NULL, IDI_
          wndclass.hCursor                   = LoadCursor (NULL,
          wndclass.hbrBackground          = (HBRUSH) GetStockO
          wndclass.lpszMenuName            = szAppName ;
```

```
        wndclass.lpszClassName              = szAppName ;

        if (!RegisterClass (&wndclass))
        {
             MessageBox (  NULL, TEXT ("This program requies W
                                                  szAppName, MB_IC
             return 0 ;
}

        hwnd = CreateWindow ( szAppName, TEXT ("Unicode Cha
                   WS_OVERLAPPEDWINDOW | WS_VSCROLL,
                   CW_USEDEFAULT, CW_USEDEFAULT,
                   CW_USEDEFAULT, CW_USEDEFAULT,
                  NULL, NULL, hInstance, NULL) ;

        ShowWindow (hwnd, iCmdShow) ;
        UpdateWindow (hwnd) ;
        while (GetMessage (&msg, NULL, 0, 0))
 {
```

```
            TranslateMessage (&msg) ;
            DispatchMessage (&msg) ;
   }
        return msg.wParam ;
}

LRESULT CALLBACK WndProc ( HWND hwnd, UINT message, WPA
{
        static CHOOSEFONT           cf ;
        static int                      iPage ;
        static LOGFONT                  lf ;
        HDC                             hdc ;
        int                         cxChar, cyChar, x, y, i, cxLab
        PAINTSTRUCT                         ps ;
        SIZE                            size ;
        TCHAR                           szBuffer [8] ;
        TEXTMETRIC                          tm ;
        WCHAR                           ch ;

        switch (message)
```

```
{
 case   WM_CREATE:
            hdc = GetDC (hwnd) ;
            lf.lfHeight = - GetDeviceCaps (hdc, LOGPIXELSY
            lstrcpy (lf.lfFaceName, TEXT ("Lucida Sans Unico
            ReleaseDC (hwnd, hdc) ;

           cf.lStructSize = sizeof (CHOOSEFONT) ;
            cf.hwndOwner         = hwnd ;
            cf.lpLogFont         = &lf ;
            cf.Flags             = CF_INITTOLOGFONTSTRUCT |

            SetScrollRange       (hwnd, SB_VERT, 0, 255, FA
           SetScrollPos         (hwnd, SB_VERT, iPage,  TRUE
            return 0 ;

 case   WM_COMMAND:
            switch (LOWORD (wParam))
            {
            case   IDM_FONT:
```

```
                    if (   ChooseFont (&cf))
                    InvalidateRect (hwnd, NULL, TRUE
                    return 0 ;
          }
              return 0 ;
     case   WM_VSCROLL:
               switch (LOWORD (wParam))
                         {
                         case SB_LINEUP:    iPage -=  1      ;  br
                        case SB_LINEDOWN:  iPage +=  1      ;
                         case SB_PAGEUP:    iPage -= 16      ;  b
                         case SB_PAGEDOWN:  iPage += 16
                         case SB_THUMBPOSITION:iPage= HIW
                         default:
                               return 0 ;
                         }
               iPage = max (0, min (iPage, 255)) ;
```

```
                SetScrollPos (hwnd, SB_VERT, iPage, TRUE) ;
                InvalidateRect (hwnd, NULL, TRUE) ;
                return 0 ;

        case   WM_PAINT:
                hdc = BeginPaint (hwnd, &ps) ;

                SelectObject (hdc, CreateFontIndirect (&lf)) ;

                GetTextMetrics (hdc, &tm) ;
                cxChar = tm.tmMaxCharWidth ;
                cyChar = tm.tmHeight + tm.tmExternalLeading

                cxLabels = 0 ;

                for (i = 0 ; i < 16 ; i++)
                {
                        wsprintf (szBuffer, TEXT (" 000%1X: "), i) ;
                        GetTextExtentPoint (hdc, szBuffer, 7, &size

                            cxLabels = max (cxLabels, size.cx) ;
```

```
                }

                for (y = 0 ; y < 16 ; y++)
                {
    wsprintf (szBuffer, TEXT (" %03X_: "), 16 * iPage + y) ;
                                TextOut (hdc, 0, y * cyChar, szBuf

                               for (x = 0 ; x < 16 ; x++)
                                {
                             ch = (WCHAR) (256 * iPage + 16 * y +
                              TextOutW (hdc, x * cxChar + cxLabels,
                                              y * cyChar, &ch, 1) ;
    }
                }

  DeleteObject (SelectObject (hdc, GetStockObject (SYSTEM_
                EndPaint (hwnd, &ps) ;
                return 0 ;

   case   WM_DESTROY:
                PostQuitMessage (0) ;
```

```
                    return 0 ;
  }
        return DefWindowProc (hwnd, message, wParam, lParam)
}
```

UNICHARS.RC

```
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
//////////////////////////////////////////////////////////////////////////////
// Menu
UNICHARS MENU DISCARDABLE
BEGIN
  MENUITEM "&Font!",                                    IDM_
END
```

RESOURCE.H

```
// Microsoft Developer Studio generated include file.
// Used by Unichars.rc
```

```
#define IDM_FONT            40001
```

DT_WORDBREAKDrawText
DrawTextDrawTextTextOut

GetTextExtentPoint32Windows

```
GetTextExtentPoint32 (hdc, pString, iCount, &size) ;
```

SIZEcxcy

```
TCHAR * szText [] = TEXT ("Hello, how are you?") ;
```

yStartxLeftxRightxStart

```
GetTextExtentPoint32 (hdc, szText, lstrlen (szText), &size) ;
```

size.cxxRightxLeft

xStartxLeft

```
TextOut (hdc, xStart, yStart, szText, lstrlen (szText)) ;
```

size.cyyStart

xStart

```
xStart = xRight - size.cx ;
```

```
xStart = (xLeft + xRight - size.cx) / 2 ;
```

xRightxLeftsize.cx

```
xRight - xLeft - size.cx
```

```
SetTextJustification (hdc, xRight - xLeft - size.cx, 3)
```

3xStartxLeftTextOut

```
TextOut (hdc, xStart, yStart, szText, lstrlen (szText)) ;
```

xLeftxRight

SetTextJustificationGetTextExtentPoint32

```
SetTextJustification (hdc, 0, 0) ;
```

GetTextExtentPoint32
SetTextJustificationTextOut

17-7JUSTIFY1Mark TwainThe Adventures of Huckleberry F
17-3JUSTIFY1

17-7 JUSTIFY1

```
JUSTIFY1.C

/*------------------------------------------------------------------------
  JUSTIFY1.C -- Justified Type Program #1
                                                (c) Charles Petzold, 199
------------------------------------------------------------------------*/

#include <windows.h>
#include "resource.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

TCHAR szAppName[] = TEXT ("Justify1") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                                PSTR szCmdLine, int iCr
{
        HWND                hwnd ;
        MSG                 msg ;
        WNDCLASS            wndclass ;
```

```
        wndclass.style                  = CS_HREDRAW | CS_V
        wndclass.lpfnWndProc               = WndProc ;
        wndclass.cbClsExtra                = 0 ;
        wndclass.cbWndExtra                = 0 ;
        wndclass.hInstance                 = hInstance ;
        wndclass.hIcon                  = LoadIcon (NULL, IDI_
        wndclass.hCursor                = LoadCursor (NULL,
        wndclass.hbrBackground          = (HBRUSH) GetStockO
        wndclass.lpszMenuName           = szAppName ;
        wndclass.lpszClassName          = szAppName ;

        if (!RegisterClass (&wndclass))
        {
                MessageBox (  NULL, TEXT ("This program requi
                                        szAppName, MB_IC
                return 0 ;
}
```

```
        hwnd = CreateWindow (szAppName, TEXT ("Justified Type
                    WS_OVERLAPPEDWINDOW,
                     CW_USEDEFAULT, CW_USEDEFAULT,
                     CW_USEDEFAULT, CW_USEDEFAULT,
                     NULL, NULL, hInstance, NULL) ;

        ShowWindow (hwnd, iCmdShow) ;
        UpdateWindow (hwnd) ;

       while (GetMessage (&msg, NULL, 0, 0))
        {
                     TranslateMessage (&msg) ;
                     DispatchMessage (&msg) ;
   }
        return msg.wParam ;
}

void DrawRuler (HDC hdc, RECT * prc)
```

```
{
        static int iRuleSize [16] = {360, 72,144, 72,216, 72,144,7
288, 72,144, 72,216, 72,144,72 } ;
        int                         i, j ;
        POINT                       ptClient ;

        SaveDC (hdc) ;
                        // Set Logical Twips mapping mode
        SetMapMode (hdc, MM_ANISOTROPIC) ;
        SetWindowExtEx (hdc, 1440, 1440, NULL) ;
        SetViewportExtEx (hdc,GetDeviceCaps (hdc, LOGPIXELSX
                        GetDeviceCaps (hdc, LOGPIXELSY), NULL) ;

                        // Move the origin to a half inch from upper

        SetWindowOrgEx (hdc, -720, -720, NULL) ;
                        // Find the right margin (quarter inch from
        ptClient.x = prc->right ;
        ptClient.y = prc->bottom ;
```

```
        DPtoLP (hdc, &ptClient, 1) ;
        ptClient.x -= 360 ;

                        // Draw the rulers
MoveToEx            (hdc, 0,         -360, NULL) ;
        LineTo              (hdc, ptClient.x, -360) ;
        MoveToEx            (hdc, -360,         0, NULL) ;
        LineTo              (hdc, -360, ptClient.y) ;

        for (i = 0, j = 0 ; i <= ptClient.x ; i += 1440 / 16, j++)
{
            MoveToEx            (hdc, i, -360, NULL) ;
            LineTo              (hdc, i, -360 - iRuleSize [j % 16]) ;
        }

        for (i = 0, j = 0 ; i <= ptClient.y ; i += 1440 / 16, j++)
{
            MoveToEx            (hdc, -360, i, NULL) ;
```

```
            LineTo              (hdc, -360 - iRuleSize [j % 16], i) ;
    }


        RestoreDC (hdc, -1) ;
}
void Justify (HDC hdc, PTSTR pText, RECT * prc, int iAlign)
{
        int          xStart, yStart, cSpaceChars ;
        PTSTR        pBegin, pEnd ;
        SIZE         size ;


        yStart = prc->top ;
        do                          // for each text line
    {
        cSpaceChars = 0 ;           // initialize number of spaces ir

             while (*     pText == '')         // skip over leading spa
                                    pText++ ;

         pBegin = pText ;                              // set pointer to c
```

```
        do                              // until the line is known
        {
                                pEnd =pText ;// set pointer to cha

                                        // skip to next space

                                while (*pText != '\0' && *pText++

                                if (*pText == '\0')
                                        break ;

                                         // after each space enco

                                cSpaceChars++ ;
                                GetTextExtentPoint32(hdc, pBegi
}
        while (size.cx < (prc->right - prc->left)) ;
        cSpaceChars-- ;                         // discount last spa
```

```
            while (*(pEnd - 1) == ' ')    // eliminate trailing spaces
            {
                        pEnd-- ;
                        cSpaceChars-- ;
    }

            // if end of text and no space characters, set pEnd to
            if (*  pText == '\0' || cSpaceChars <= 0)
                                    pEnd = pText ;

            GetTextExtentPoint32 (hdc, pBegin, pEnd - pBegin, &


            switch (iAlign)      // use alignment for xStart
            {
            case   IDM_ALIGN_LEFT:
                        xStart = prc->left ;
                        break ;


            case   IDM_ALIGN_RIGHT:
                        xStart = prc->right - size.cx ;
```

```
                        break ;

                case   IDM_ALIGN_CENTER:
                        xStart = (prc->right + prc->left - size.
                        break ;

                case   IDM_ALIGN_JUSTIFIED:
                        if (*  pText != '\0' && cSpaceChars > 0
SetTextJustification (hdc, prc->right-prc->left - size.cx, cSpaceCh
                        xStart = prc->left ;
                        break ;
                }
                        // display the text

                TextOut (hdc, xStart, yStart, pBegin, pEnd - pBegin) ;

                        // prepare for next line

                SetTextJustification (hdc, 0, 0) ;
```

```
            yStart += size.cy ;
            pText = pEnd ;
    }
        while (*pText && yStart < prc->bottom - size.cy) ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
        static CHOOSEFONT     cf ;
        static DOCINFO                di = { sizeof (DOCINFO), TE
        static int                      iAlign = IDM_ALIGN_LEFT ;
        static LOGFONT              lf ;
        static PRINTDLG             pd ;
        static TCHAR                    szText[] = {
    TEXT ("You don't know about me, without you ")
    TEXT ("have read a book by the name of \"The ")
    TEXT ("Adventures of Tom Sawyer,\" but that ")
    TEXT ("ain't no matter. That book was made by ")
    TEXT ("Mr. Mark Twain, and he told the truth, ")
```

```
TEXT ("mainly. There was things which he ")
TEXT ("stretched, but mainly he told the truth. ")
TEXT ("That is nothing. I never seen anybody ")
TEXT ("but lied, one time or another, without ")
TEXT ("it was Aunt Polly, or the widow, or ")
TEXT ("maybe Mary. Aunt Polly -- Tom's Aunt ")
TEXT ("Polly, she is -- and Mary, and the Widow ")
TEXT ("Douglas, is all told about in that book ")
TEXT ("-- which is mostly a true book; with ")
TEXT ("some stretchers, as I said before.") } ;
     BOOL                        fSuccess ;
     HDC                         hdc, hdcPrn ;
     HMENU                       hMenu ;
     int                         iSavePointSize ;
     PAINTSTRUCT                 ps ;
     RECT                        rect ;

     switch (message)
     {
```

```
        case   WM_CREATE:
                                    // Initialize the CHOOSEFONT stru

                GetObject (GetStockObject (SYSTEM_FONT), siz

                cf.lStructSize                  = sizeof (CHOOSEFO
                cf.hwndOwner                    = hwnd ;
                cf.hDC                          = NULL ;
                cf.lpLogFont                    = &lf ;
                cf.iPointSize                   = 0 ;
                cf.Flags                        = CF_INITTOLOGFONTS
              CF_EFFECTS ;
                cf.rgbColors                    = 0 ;
                cf.lCustData                    = 0 ;
                cf.lpfnHook                     = NULL ;
                cf.lpTemplateName               = NULL ;
                cf.hInstance                    = NULL ;
                cf.lpszStyle                    = NULL ;
                cf.nFontType                    = 0 ;
```

```
                cf.nSizeMin                     = 0 ;
                cf.nSizeMax                     = 0 ;

                return 0 ;

        case   WM_COMMAND:
                hMenu = GetMenu (hwnd) ;


                switch (LOWORD (wParam))
                {
                case IDM_FILE_PRINT:
                                // Get printer DC

                                pd.lStructSize       = sizeof (PRINT
                                pd.hwndOwner         = hwnd ;
                                pd.Flags             = PD_RETURNDC

                                if (!PrintDlg (&pd))
                                        return 0 ;

                                if (NULL == (hdcPrn = pd.hDC))
```

```
                                    {
            MessageBox (    hwnd, TEXT ("Cannot obtain Printer I
                        szAppName, MB_ICONEXCLAMATION | MB_OK
                          return 0 ;
                                    }
                                          // Set margins of 1 inch

            rect.left     =      GetDeviceCaps (hdcPrn, LOGPIXELSX
                                GetDeviceCaps (hdcPrn, PHYSICALOFF

          rect.top  = GetDeviceCaps (hdcPrn, LOGPIXELSY)
                        GetDeviceCaps (hdcPrn, PHYSICALOFFSETY) ;
rect.right =       GetDeviceCaps (hdcPrn, PHYSICALWIDTH) -
                        GetDeviceCaps (hdcPrn, LOGPIXELSX) -
                        GetDeviceCaps (hdcPrn, PHYSICALOFFSETX) ;
rect.bottom=       GetDeviceCaps (hdcPrn, PHYSICALHEIGHT) -
                        GetDeviceCaps (hdcPrn, LOGPIXELSY) -
                        GetDeviceCaps (hdcPrn, PHYSICALOFFSETY) ;

                                          // Display text on printer
```

```
                                    SetCursor (LoadCursor (NULL, IDC
                                    ShowCursor (TRUE) ;

                                    fSuccess = FALSE ;
if ((StartDoc (hdcPrn, &di) > 0) && (StartPage (hdcPrn) > 0))
                                    {
                                  // Select font using adjusted lfHeight

              iSavePointSize = lf.lfHeight ;
              lf.lfHeight = -(GetDeviceCaps (hdcPrn, LOGPIXELSY)
                                cf.iPointSize) / 720 ;

              SelectObject (hdcPrn, CreateFontIndirect (&lf)) ;
              lf.lfHeight = iSavePointSize ;

                 // Set text color

              SetTextColor (hdcPrn, cf.rgbColors) ;

               // Display text
```

```
        Justify (hdcPrn, szText, &rect, iAlign) ;

        if (EndPage (hdcPrn) > 0)
                                        {
                                        fSuccess = TRUE ;
                                       EndDoc (hdcPrn) ;
                                        }
                        }
                        ShowCursor (FALSE) ;
                        SetCursor (LoadCursor (NULL, IDC_ARR

                        DeleteDC (hdcPrn) ;

                        if (!fSuccess)
      MessageBox (hwnd, TEXT ("Could not print text"),
                  szAppName, MB_ICONEXCLAMATION | MB_OK
             return 0 ;

case   IDM_FONT:
                        if (ChooseFont (&cf))
```

```
                                    InvalidateRect (hwnd, NULL,
                    return 0 ;

          case   IDM_ALIGN_LEFT:
          case   IDM_ALIGN_RIGHT:
          case   IDM_ALIGN_CENTER:
          case   IDM_ALIGN_JUSTIFIED:
                    CheckMenuItem (hMenu, iAlign, MF_UN
                    iAlign = LOWORD (wParam) ;
                    CheckMenuItem (hMenu, iAlign, MF_CH
                    InvalidateRect (hwnd, NULL, TRUE) ;
                    return 0 ;
               }
              return 0 ;

     case   WM_PAINT:
               hdc = BeginPaint (hwnd, &ps) ;

               GetClientRect (hwnd, &rect) ;
```

```
                DrawRuler (hdc, &rect) ;

                rect.left     +=     GetDeviceCaps (hdc, LOGPIXE
                rect.top      +=     GetDeviceCaps (hdc, LOGPIXE
                rect.right           -=     GetDeviceCaps (hdc, LOG

                SelectObject (hdc, CreateFontIndirect (&lf)) ;
                SetTextColor (hdc, cf.rgbColors) ;

               Justify (hdc, szText, &rect, iAlign) ;

                DeleteObject (SelectObject (hdc, GetStockObjec
                EndPaint (hwnd, &ps) ;
                return 0 ;

      case   WM_DESTROY:
                PostQuitMessage (0) ;
                return 0 ;
}
```

```
        return DefWindowProc (hwnd, message, wParam, lParam)
}
```

JUSTIFY1.RC

```
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
//////////////////////////////////////////////////////////////////////////////
// Menu
JUSTIFY1 MENU DISCARDABLE BEGIN POPUP "&File"
  BEGIN
            MENUITEM "&Print",                                IDM_FILE
  END
  POPUP "&Font"
  BEGIN
        MENUITEM "&Font...",      IDM_FONT
  END
  POPUP "&Align"
  BEGIN
```

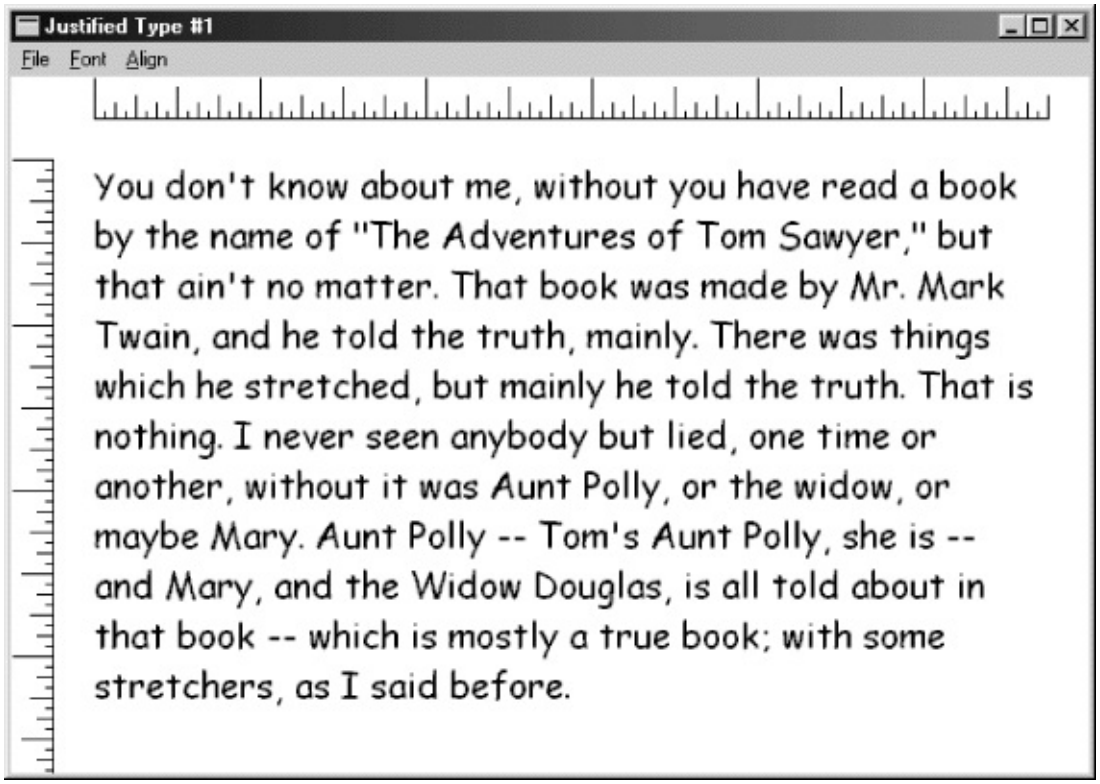
```
    MENUITEM "&Left",                       IDM_ALIGN_LEFT, CHECK
    MENUITEM "&Right",                      IDM_ALIGN_RIGHT
    MENUITEM "&Centered",                   IDM_ALIGN_CENTER
    MENUITEM "&Justified",                  IDM_ALIGN_JUSTIFIED
  END
END
```

```
RESOURCE.H
// Microsoft Developer Studio generated include file.
// Used by Justify1.rc
#define         IDM_FILE_PRINT   40001
#define         IDM_FONT        40002
#define         IDM_ALIGN_LEFT   40003
#define         IDM_ALIGN_RIGHT  40004
#define         IDM_ALIGN_CENTER 40005
#define         IDM_ALIGN_JUSTIFIED  40006
```

JUSTIFY1… DrawRuler

JustifyGetTextExtentPoint32JUSTIFY1linefeediAlign

JUSTIFY1Windows



17-3
JUSTIFY1

TrueTypeWYSIWYG

JUSTIFY1Print1…
Printrect.rightWM_PAINT

```
rect.right = rect.left + 6 * GetDeviceCaps (hdc, LOGPIXELSX) ;
```

Print

```
rect.right = rect.left + 6 * GetDeviceCaps (hdcPrn, LOGPIXELSX)
```

TrueTypelinefeed

linefeed

17-8JUSTIFY2JUSTIFY2MicrosoftDavid
JustifyJUSTIFY1Herman

17-8　JUSTIFY2

```
JUSTIFY2.C

/*------------------------------------------------------------------------

  JUSTIFY2.C --         Justified Type Program #2

                                                (c) Charles Petzold, 199

------------------------------------------------------------------------*/

#include <windows.h>

#include "resource.h"

#define OUTWIDTH 6                      // Width of formatted output

#define LASTCHAR 127                     // Last character code used

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
```

```
TCHAR szAppName[] = TEXT ("Justify2") ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                                PSTR szCmdLine, int iCr
{
   HWND                  hwnd ;
        MSG                   msg ;
        WNDCLASS wndclass ;
        wndclass.style                    = CS_HREDRAW | CS_V
        wndclass.lpfnWndProc                = WndProc ;
        wndclass.cbClsExtra                 = 0 ;
        wndclass.cbWndExtra                 = 0 ;
        wndclass.hInstance                  = hInstance ;
        wndclass.hIcon                    = LoadIcon (NULL, IDI_
        wndclass.hCursor                    = LoadCursor (NULL,
        wndclass.hbrBackground            = (HBRUSH) GetStockO
        wndclass.lpszMenuName             = szAppName ;
        wndclass.lpszClassName            = szAppName ;
        if (!RegisterClass (&wndclass))
```

```
{
          MessageBox (NULL, TEXT ("This program requires Wi
                                          szAppName, MB_ICONE
          return 0 ;
}

     hwnd = CreateWindow (szAppName, TEXT ("Justified Type
               WS_OVERLAPPEDWINDOW,
               CW_USEDEFAULT, CW_USEDEFAULT,
               CW_USEDEFAULT, CW_USEDEFAULT,
               NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
```

```
        }
        return msg.wParam ;
}

void DrawRuler (HDC hdc, RECT * prc)
{
        static int iRuleSize [16] = {360,72,144, 72,216,72,144,72
72,216,72,144, 72 } ;
        int                     i, j ;
        POINT               ptClient ;

        SaveDC (hdc) ;
                    // Set Logical Twips mapping mode
        SetMapMode (hdc, MM_ANISOTROPIC) ;
        SetWindowExtEx (hdc, 1440, 1440, NULL) ;
        SetViewportExtEx (hdc,       GetDeviceCaps (hdc, LOGPIX
                    GetDeviceCaps (hdc, LOGPIXELSY), NULL) ;

              // Move the origin to a half inch from upper left
```

```
SetWindowOrgEx (hdc, -720, -720, NULL) ;
          // Find the right margin (quarter inch from right)
ptClient.x = prc->right ;
ptClient.y = prc->bottom ;
DPtoLP (hdc, &ptClient, 1) ;
ptClient.x -= 360 ;

          // Draw the rulers
MoveToEx          (hdc, 0,    -360, NULL) ;
LineTo            (hdc, OUTWIDTH * 1440,-36    0) ;
MoveToEx          (hdc, -360,   0, NULL) ;
LineTo            (hdc, -360,      ptClient.y) ;

for (i = 0, j = 0 ;   i <= ptClient.x && i <= OUTWIDTH * 1
  i += 1440 / 16, j++)
{
          MoveToEx          (hdc, i, -360, NULL) ;
```

```
                LineTo          (hdc, i, -360 - iRuleSize [j % 16]
        }

    for (i = 0, j = 0 ; i <= ptClient.y ; i += 1440 / 16, j++)
        {
                MoveToEx        (hdc, -360, i, NULL) ;
                LineTo          (hdc, -360 - iRuleSize [j % 16], i
    }

        RestoreDC (hdc, -1) ;
}

/*------------------------------------------------------------------------
  GetCharDesignWidths:  Gets character widths for font as large
                                                        original desig
------------------------------------------------------------------------*/

UINT GetCharDesignWidths (HDC hdc, UINT uFirst, UINT uLast, in
{
        HFONT                                           hFont, hFontDes
```

```
LOGFONT                                    lf ;
OUTLINETEXTMETRIC            otm ;

hFont = GetCurrentObject (hdc, OBJ_FONT) ;
GetObject (hFont, sizeof (LOGFONT), &lf) ;

     // Get outline text metrics (we'll only be using a field
     //           independent of the DC the font is selected i

otm.otmSize = sizeof (OUTLINETEXTMETRIC) ;
GetOutlineTextMetrics (hdc, sizeof (OUTLINETEXTMETRIC)

     // Create a new font based on the design size
lf.lfHeight   = - (int) otm.otmEMSquare ;
lf.lfWidth         = 0 ;
hFontDesign   = CreateFontIndirect (&lf) ;

     // Select the font into the DC and get the character w

SaveDC (hdc) ;
SetMapMode (hdc, MM_TEXT) ;
```

```
        SelectObject (hdc, hFontDesign) ;

        GetCharWidth (hdc, uFirst, uLast, piWidths) ;
        SelectObject (hdc, hFont) ;
        RestoreDC (hdc, -1) ;

                // Clean up
        DeleteObject (hFontDesign) ;
        return otm.otmEMSquare ;
}

/*-------------------------------------------------------------------------
  GetScaledWidths:      Gets floating point character widths for se
                                                        font size
-------------------------------------------------------------------------*/

void GetScaledWidths (HDC hdc, double * pdWidths)
{
        double              dScale ;
        HFONT                   hFont ;
        int                     aiDesignWidths [LASTCHAR + 1] ;
```

```
        int                     i ;
        LOGFONT             lf ;
        UINT                    uEMSquare ;

                // Call function above

        uEMSquare = GetCharDesignWidths (hdc, 0, LASTCHAR, a
                // Get LOGFONT for current font in device contex
        hFont = GetCurrentObject (hdc, OBJ_FONT) ;
        GetObject (hFont, sizeof (LOGFONT), &lf) ;
                // Scale the widths and store as floating point va
        dScale = (double) -lf.lfHeight / (double) uEMSquare ;
        for (  i = 0 ; i <= LASTCHAR ; i++)
                        pdWidths[i] = dScale * aiDesignWidths[i] ;
}

/*-------------------------------------------------------------------------
  GetTextExtentFloat:  Calculates text width in floating point
-------------------------------------------------------------------------*/
```

```
double GetTextExtentFloat (double * pdWidths, PTSTR psText, int
{
        double dWidth = 0 ;
        int         i ;

        for (  i = 0 ; i < iCount ; i++)
                        dWidth += pdWidths [psText[i]] ;

   return dWidth ;
}

/*------------------------------------------------------------------------
  Justify: Based on design units for screen/printer compatibility
------------------------------------------------------------------------*/

void Justify (HDC hdc, PTSTR pText, RECT * prc, int iAlign)
{
        double dWidth, adWidths[LASTCHAR + 1] ;
        int         xStart, yStart, cSpaceChars ;
        PTSTR       pBegin, pEnd ;
        SIZE        size ;
```

```
            // Fill the adWidths array with floating point character
        GetScaledWidths (hdc, adWidths) ;
        yStart = prc->top ;
        do                          // for each text line
{
            cSpaceChars = 0 ;       // initialize number of spaces
            while (*pText == ' ')// skip over leading spaces
                                pText++ ;
        pBegin = pText ;            // set pointer to char at beginnin
        do                          // until the line is known
            {
                        pEnd = pText ;      // set pointer to char at
                                            // skip to next space
                        while (*pText != '\0' && *pText++ != ' ') ;
```

```
        if (*pText == '\0')
                        break ;

        // after each space encountered, calculate exter

            cSpaceChars++ ;
            dWidth = GetTextExtentFloat (adWidths,
                        pText - pBegin - 1) ;
    }
    while (dWidth < (double) (prc->right - prc->left)) ;

    cSpaceChars-- ;        // discount last space at end of l

    while (*(pEnd - 1) == ' ')    // eliminate trailing spaces
    {
            pEnd-- ;
            cSpaceChars-- ;
    }
```

```
                    // if end of text and no space characters, se


          if (*pText == '\0' || cSpaceChars <= 0)
                              pEnd = pText ;

                    // Now get integer extents

          GetTextExtentPoint32(hdc, pBegin, pEnd - pBegin, &s


          switch (iAlign)              // use alignment for xStart
          {
          case   IDM_ALIGN_LEFT:
                              xStart = prc->left ;
                              break ;

          case   IDM_ALIGN_RIGHT:
                              xStart = prc->right - size.cx ;
                              break ;
          case   IDM_ALIGN_CENTER:
                              xStart = (prc->right + prc->left - size.
```

```
                        break ;

                case   IDM_ALIGN_JUSTIFIED:
                        if (*pText != '\0' && cSpaceChars > 0)
                                SetTextJustification (hdc,
                          prc->right - prc->left - size.cx,
                            cSpaceChars) ;
                        xStart = prc->left ;
                        break ;
                }
                        // display the text

                TextOut (hdc, xStart, yStart, pBegin, pEnd - pBegin) ;

                        // prepare for next line

                SetTextJustification (hdc, 0, 0) ;
                yStart += size.cy ;
                pText = pEnd ;
        }
```

```
        while (*pText && yStart < prc->bottom - size.cy) ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
        static CHOOSEFONT    cf ;
        static       DOCINFO                    di = { sizeof (DOCINFO
        static int                       iAlign = IDM_ALIGN_LEFT ;
        static LOGFONT             lf ;
        static PRINTDLG            pd ;
        static TCHAR                     szText[] = {
  TEXT ("Call me Ishmael. Some years ago -- never ")
  TEXT ("mind how long precisely -- having little ")
  TEXT ("or no money in my purse, and nothing ")
  TEXT ("particular to interest me on shore, I ")
  TEXT ("thought I would sail about a little and ")
  TEXT ("see the watery part of the world. It is ")
  TEXT ("a way I have of driving off the spleen, ")
  TEXT ("and regulating the circulation. Whenever ")
```

```
TEXT ("I find myself growing grim about the ")
TEXT ("mouth; whenever it is a damp, drizzly ")
TEXT ("November in my soul; whenever I find ")
TEXT ("myself involuntarily pausing before ")
TEXT ("coffin warehouses, and bringing up the ")
TEXT ("rear of every funeral I meet; and ")
TEXT ("especially whenever my hypos get such an ")
TEXT ("upper hand of me, that it requires a ")
TEXT ("strong moral principle to prevent me ")
TEXT ("from deliberately stepping into the ")
TEXT ("street, and methodically knocking ")
TEXT ("people's hats off -- then, I account it ")
TEXT ("high time to get to sea as soon as I ")
TEXT ("can. This is my substitute for pistol ")
TEXT ("and ball. With a philosophical flourish ")
TEXT ("Cato throws himself upon his sword; I ")
TEXT ("quietly take to the ship. There is ")
TEXT ("nothing surprising in this. If they but ")
```

```
TEXT ("knew it, almost all men in their degree, ")
TEXT ("some time or other, cherish very nearly ")
TEXT ("the same feelings towards the ocean with ")
TEXT ("me.") } ;
        BOOL                        fSuccess ;
        HDC                         hdc, hdcPrn ;
        HMENU                       hMenu ;
        int                         iSavePointSize ;
        PAINTSTRUCT                 ps ;
        RECT                        rect ;

        switch (message)
        {
        case   WM_CREATE:
                                // Initialize the CHOOSEFONT structure

                hdc = GetDC (hwnd) ;
                lf.lfHeight = - GetDeviceCaps (hdc, LOGPIXELSY
                lf.lfOutPrecision = OUT_TT_ONLY_PRECIS ;
```

```
lstrcpy (lf.lfFaceName, TEXT ("Times New Roman
ReleaseDC (hwnd, hdc) ;

cf.lStructSize    = sizeof (CHOOSEFO
cf.hwndOwner      = hwnd ;
cf.hDC            = NULL ;
cf.lpLogFont      = &lf ;
cf.iPointSize     = 120 ;

                  // Set flags for TrueType only!

cf.Flags    = CF_INITTOLOGFONTSTRUCT | CF_SC
                  CF_TTONLY | CF_EFFECTS ;
cf.rgbColors      = 0 ;
cf.lCustData      = 0 ;
cf.lpfnHook       = NULL ;
cf.lpTemplateName      = NULL ;
cf.hInstance      = NULL ;
cf.lpszStyle      = NULL ;
cf.nFontType      = 0 ;
```

```
                cf.nSizeMin                 = 0 ;
                cf.nSizeMax                  = 0 ;

                return 0 ;

        case   WM_COMMAND:
                hMenu = GetMenu (hwnd) ;


                switch (LOWORD (wParam))
                {
                case   IDM_FILE_PRINT:
                                // Get printer DC

                                 pd.lStructSize = sizeof (PRINTDLG
                                pd.hwndOwner   = hwnd ;
          pd.Flags            = PD_RETURNDC | PD_NOPAGENUM

                        if (!PrintDlg (&pd))
                                        return 0 ;

                                if (NULL == (hdcPrn = pd.hDC))
```

```
                    {
MessageBox(hwnd, TEXT ("Cannot obtain Printer DC"),
          szAppName, MB_ICONEXCLAMATION | MB_OK)
        return 0 ;
                    }
                              // Set margins for OUTWIDTH

                rect.left = (GetDeviceCaps (hdcPrn, PH
 GetDeviceCaps (hdcPrn, LOGPIXELSX)*OUTWID
-      GetDeviceCaps (hdcPrn, PHYSICALOFFSETX


              rect.right = rect.left +
 GetDeviceCaps (hdcPrn, LOGPIXELSX) * OUTWI

// Set margins of 1 inch at top and bottom

               rect.top  = GetDeviceCaps (hdcPrn, LO
GetDeviceCaps (hdcPrn, PHYSICALOFFSETY) ;

          rect.bottom =GetDeviceCaps (hdcPrn, PHYS
GetDeviceCaps (hdcPrn, LOGPIXELSY) -
```

```
        GetDeviceCaps (hdcPrn, PHYSICALOFFSETY) ;

                                // Display text on printer

                        SetCursor (LoadCursor (NULL, IDC_WA
                        ShowCursor (TRUE) ;

                        fSuccess = FALSE ;

                        if ((StartDoc (hdcPrn, &di) > 0) && (St
                        {
          // Select font using adjusted lfHeight

    iSavePointSize = lf.lfHeight ;
    lf.lfHeight = -(GetDeviceCaps (hdcPrn, LOGPIXELSY)
                        cf.iPointSize) / 720 ;

    SelectObject (hdcPrn, CreateFontIndirect (&lf)) ;
                                lf.lfHeight = iSavePointSize ;

                                        // Set text color
```

```
                    SetTextColor (hdcPrn, cf.rgbColors

                                // Display text

        Justify (hdcPrn, szText, &rect, iAlign) ;

                                if (EndPage (hdcPrn) > 0)
                                {
                                        fSuccess = TRUE ;
                                         EndDoc (hdcPrn) ;
                                }
                            }
                        ShowCursor (FALSE) ;
                        SetCursor (LoadCursor (NULL, IDC_ARROW

                        DeleteDC (hdcPrn) ;

                        if (!fSuccess)
                  MessageBox (hwnd, TEXT ("Could not print text"),
                           szAppName, MB_ICONEXCLAMATION | MB_OK
                        return 0 ;
```

```
                case   IDM_FONT:
                              if (ChooseFont (&cf))
                                InvalidateRect (hwnd, NULL, TRUE
                                return 0 ;

                case   IDM_ALIGN_LEFT:
                case   IDM_ALIGN_RIGHT:
                case   IDM_ALIGN_CENTER:
                case   IDM_ALIGN_JUSTIFIED:
                              CheckMenuItem (hMenu, iAlign, MF
                                iAlign = LOWORD (wParam) ;
                                CheckMenuItem (hMenu, iAlign, MF
                                InvalidateRect (hwnd, NULL, TRUE
                                return 0 ;
               }
                return 0 ;

      case   WM_PAINT:
                hdc = BeginPaint (hwnd, &ps) ;
```

```
                GetClientRect (hwnd, &rect) ;
                DrawRuler (hdc, &rect) ;

                rect.left     += GetDeviceCaps (hdc, LOGPIXELSX
                rect.top      += GetDeviceCaps (hdc, LOGPIXELS
                rect.right    = rect.left + OUTWIDTH * GetDevice

                SelectObject (hdc, CreateFontIndirect (&lf)) ;
                SetTextColor (hdc, cf.rgbColors) ;

                Justify (hdc, szText, &rect, iAlign) ;

                DeleteObject (SelectObject (hdc, GetStockObjec
                EndPaint (hwnd, &ps) ;
               return 0 ;

        case   WM_DESTROY:
                PostQuitMessage (0) ;
```

```
                    return 0 ;
   }
        return DefWindowProc (hwnd, message, wParam, lParam)
}
```

JUSTIFY2.RC

```
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
//////////////////////////////////////////////////////////////////////////////
// Menu
JUSTIFY2 MENU DISCARDABLE BEGIN POPUP "&File"
  BEGIN
            MENUITEM "&Print",          IDM_FILE_PRINT
  END
  POPUP "&Font"
  BEGIN
            MENUITEM "&Font...",  IDM_FONT
  END
```

```
  POPUP "&Align"
  BEGIN
            MENUITEM "&Left",                IDM_ALIGN_LEFT, CHI
            MENUITEM "&Right",    IDM_ALIGN_RIGHT
            MENUITEM "&Centered",        IDM_ALIGN_CENTER
           MENUITEM "&Justified",       IDM_ALIGN_JUSTIFIED
  END
END
```

RESOURCE.H

```
// Microsoft Developer Studio generated include file.
// Used by Justify2.rc
#define          IDM_FILE_PRINT      40001
#define          IDM_FONT           40002
#define          IDM_ALIGN_LEFT      40003
#define          IDM_ALIGN_RIGHT     40004
#define          IDM_ALIGN_CENTER    40005
#define          IDM_ALIGN_JUSTIFIED 40006
```

JUSTIFY2TrueTypeGetCharDesignWidthsGetOutlineTextMetrics
OUTLINETEXTMETRICotmEMSquare

TrueTypeem-squareemMTrueType
OUTLINETEXTMETRICotmEMSquareTrueTypeotmEMSquare
20482048×2048

TrueTypeLOGFONTlfHeightotmEMSquareGetCharWidth
otmEMSquare

GetCharDesignWidthsJUSTIFY2ASCIIGetScaledWidths
GetTextExtentFloatJustify

GDI

GDIWindows32

```
BeginPath (hdc) ;
```

GDILineToPolylineToBezierTo
MoveToEx/

CloseFigureCloseFigure

```
EndPath (hdc) ;
```

```
StrokePath (hdc) ;
```

```
FillPath (hdc) ;

StrokeAndFillPath (hdc) ;

hRgn = PathToRegion (hdc) ;

SelectClipPath (hdc, iCombine) ;
```

StrokePath

FillPathStrokeAndFillPathiCombineCombineRgnRGN_

Windows

StrokePathCreatePenWindowsExtCreatePen
ExtCreatePen

```
hPen = ExtCreatePen (iStyle, iWidth, &lBrush, 0, NULL) ;
```

Windows 98Windows

ExtCreatePen CreatePenPS_GEOMETRICiWidth
PS_COSMETICiWidth1Windows 98PS_COSMETICWindow

CreatePenExtCreatePenPS_GEOMETRIC

CreatePenExtCreatePen
Windows NTExtCreatePen

```
PS_ENDCAP_ROUND
```

```
PS_ENDCAP_SQUARE
PS_ENDCAP_FLAT
```

squareflat

```
PS_JOIN_ROUND
PS_JOIN_BEVEL
PS_JOIN_MITER
```

bevelmiter17-9ENDJOIN

17-9　ENDJOIN

```
ENDJOIN.C
/*------------------------------------------------------------------------
  ENDJOIN.C --  Ends and Joins Demo
                                          (c) Charles Petzold, 1998
------------------------------------------------------------------------*/
#include <windows.h>
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
```

```
                                        PSTR szCmdLine, int iCm
{
     static TCHAR szAppName[] = TEXT ("EndJoin") ;
     HWND                 hwnd ;
     MSG                  msg ;
     WNDCLASS             wndclass ;

     wndclass.style                = CS_HREDRAW | CS_V
     wndclass.lpfnWndProc            = WndProc ;
     wndclass.cbClsExtra             = 0 ;
     wndclass.cbWndExtra             = 0 ;
     wndclass.hInstance              = hInstance ;
     wndclass.hIcon                  = LoadIcon (NULL, IDI_
     wndclass.hCursor                = LoadCursor (NULL, I
     wndclass.hbrBackground          = (HBRUSH) GetStockO
     wndclass.lpszMenuName           = NULL ;
     wndclass.lpszClassName          = szAppName ;

     if (!RegisterClass (&wndclass))
```

```
{
          MessageBox (  NULL, TEXT ("This program requires W
                                                  szAppName, MB_IC
          return 0 ;
}


     hwnd = CreateWindow ( szAppName, TEXT ("Ends and Joi
                         WS_OVERLAPPEDWINDOW,
                        CW_USEDEFAULT, CW_USEDEFAULT,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                        NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
```

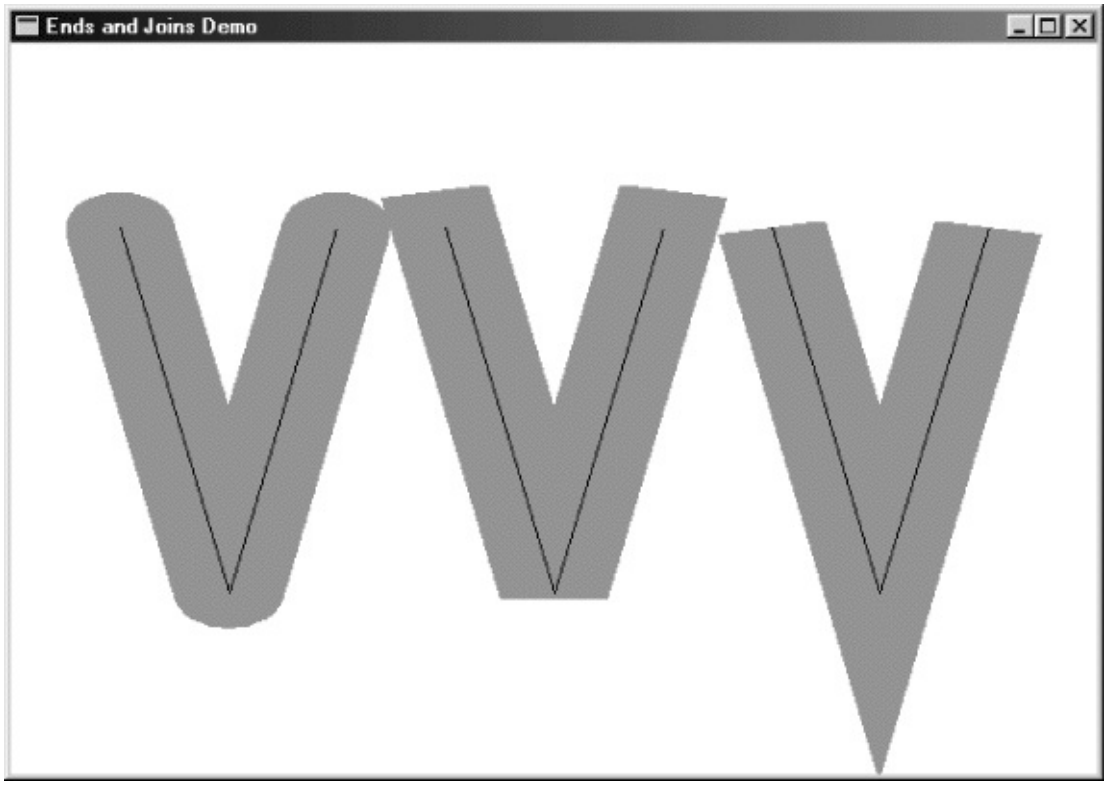
```
            DispatchMessage (&msg) ;
        }
        return msg.wParam ;
}
LRESULT CALLBACK WndProc ( HWND hwnd, UINT iMsg, WPARAM
{
        static int iEnd[] =   {PS_ENDCAP_ROUND,PS_ENDCAP_SQ
        static int iJoin[]=   {PS_JOIN_ROUND,PS_JOIN_BEVEL,PS_JC
        static int    cxClient, cyClient ;
        HDC                hdc ;
        int                i ;
        LOGBRUSH           ib ;
        PAINTSTRUCT        ps ;

        switch (iMsg)
        {
        case   WM_SIZE:
                cxClient = LOWORD (lParam) ;
                cyClient = HIWORD (lParam) ;
```

```
                return 0 ;

        case   WM_PAINT:
                hdc = BeginPaint (hwnd, &ps) ;

                SetMapMode (hdc, MM_ANISOTROPIC) ;
                SetWindowExtEx (hdc, 100, 100, NULL) ;
                SetViewportExtEx (hdc, cxClient, cyClient, NULL)

                lb.lbStyle = BS_SOLID ;
                lb.lbColor = RGB (128, 128, 128) ;
                lb.lbHatch = 0 ;

                for (i = 0 ; i < 3 ; i++)
                {
                SelectObject (hdc, ExtCreatePen (PS_SOLID | PS_
                      iEnd [i] | iJoin [i], 10, &lb, 0, NULL)) ;
                                BeginPath (hdc) ;
```

```
                                MoveToEx          (hdc, 10 + 30 *
                               LineTo             (hdc, 20 + 30 * i, 7
                                LineTo             (hdc, 30 + 30 * i,

                                EndPath (hdc) ;
                                StrokePath (hdc) ;

                                DeleteObject (
          SelectObject (hdc,GetStockObject (BLACK_PEN))) ;

                                MoveToEx     (hdc, 10 + 30 * i, 25
                                LineTo       (hdc, 20 + 30 * i, 75)
                                LineTo       (hdc, 30 + 30 * i, 25)
               }
          EndPaint (hwnd, &ps) ;
          return 0 ;

     case   WM_DESTROY:
               PostQuitMessage (0) ;
```

```
                return 0 ;

        }

  return DefWindowProc (hwnd, iMsg, wParam, lParam) ;

}
```

V17-4



17-4 ENDJOIN

WindowsStrokePathGDIGDI

17-10FONTOUT1

17-10　FONTOUT1

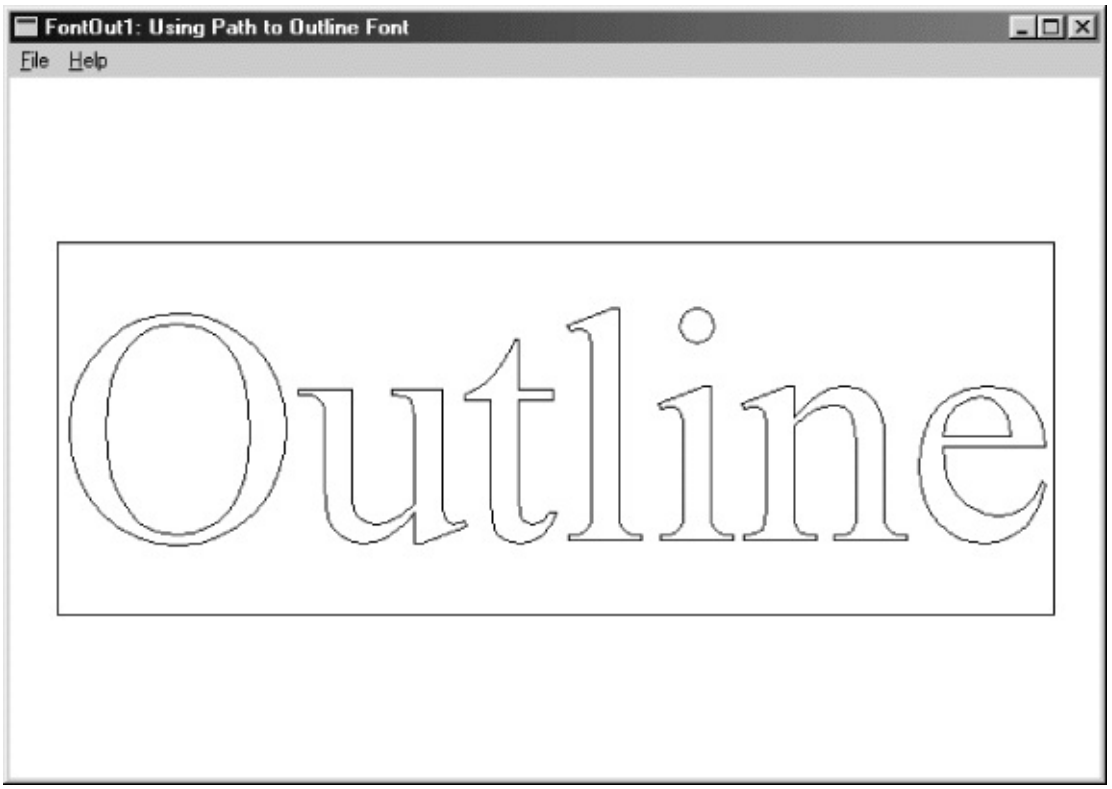
```
FONTOUT1.C

/*------------------------------------------------------------------------
  FONTOUT1.C --         Using Path to Outline Font

                                                (c) Charles Petzold, 199
------------------------------------------------------------------------*/

#include <windows.h>

#include "..\\eztest\\ezfont.h"

TCHAR szAppName [] = TEXT ("FontOut1") ;

TCHAR szTitle [] = TEXT ("FontOut1: Using Path to Outline Font")

void PaintRoutine (HWND hwnd, HDC hdc, int cxArea, int cyArea

{

          static TCHAR szString [] = TEXT ("Outline") ;

          HFONT                          hFont ;
```

```
        SIZE                    size ;

        hFont = EzCreateFont (hdc, TEXT ("Times New Roman"), 1
        SelectObject (hdc, hFont) ;
        GetTextExtentPoint32 (hdc, szString, lstrlen (szString), &s
       BeginPath (hdc) ;
        TextOut (hdc, (cxArea - size.cx) / 2, (cyArea - size.cy) / 2,
                                        szString, lstrlen (szStrin
       EndPath (hdc) ;
        StrokePath (hdc) ;
        SelectObject (hdc, GetStockObject (SYSTEM_FONT)) ;
        DeleteObject (hFont) ;
}
```

EZFONTFONTDEMO

144TrueTypeGetTextExtentPoint32TextOutTextOutBeginPath EndPathGDI

FONTOUT1StrokePathGDI17-5



17-5 FONTOUT1

OPAQUETRANSPARENTOPAQUEGDI

ExtCreatePen17-11FONTOUT2

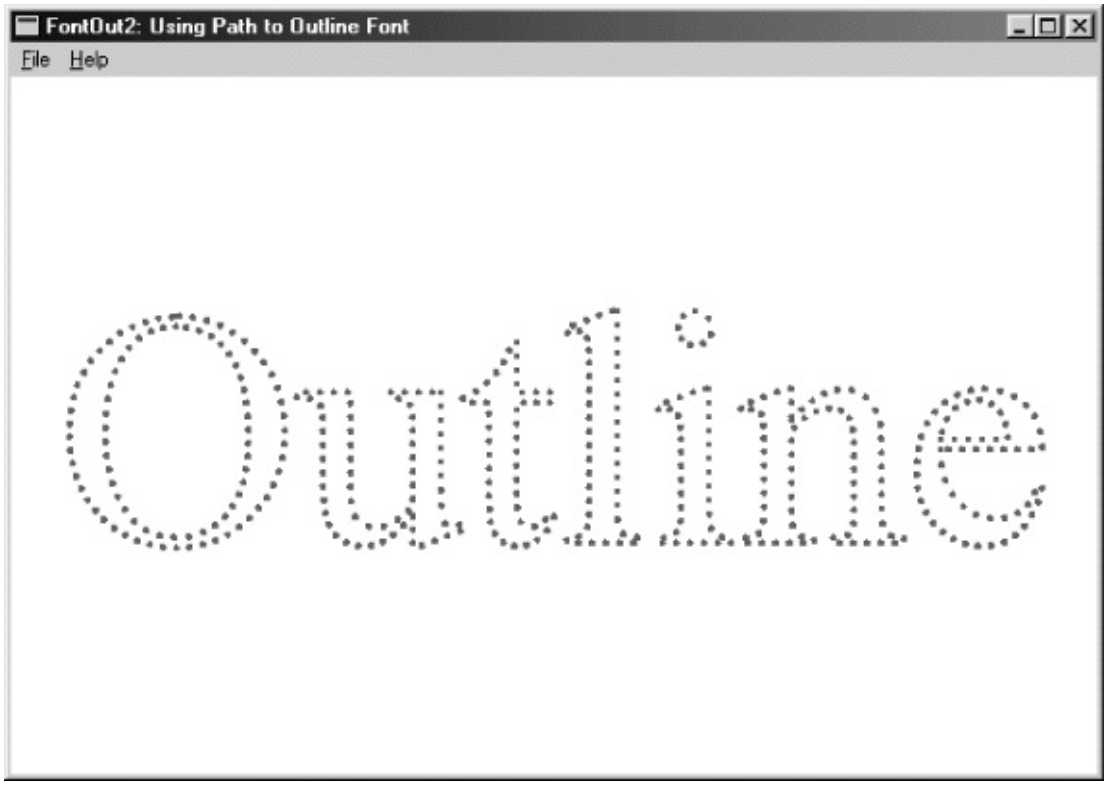
17-11 FONTOUT2

```
FONTOUT2.C

/*------------------------------------------------------------------------

  FONTOUT2.C --         Using Path to Outline Font

                                                (c) Charles Petzold, 199
```

```
------------------------------------------------------------------------*/

#include <windows.h>

#include "..\\eztest\\ezfont.h"

TCHAR szAppName [] = TEXT ("FontOut2") ;

TCHAR szTitle [] = TEXT ("FontOut2: Using Path to Outline Font")

void PaintRoutine (HWND hwnd, HDC hdc, int cxArea, int cyArea

{

          static TCHAR szString [] = TEXT ("Outline") ;

          HFONT                          hFont ;

          LOGBRUSH                   lb ;

          SIZE                            size ;

          hFont = EzCreateFont (hdc, TEXT ("Times New Roman"), 1

          SelectObject (hdc, hFont) ;

          SetBkMode (hdc, TRANSPARENT) ;

          GetTextExtentPoint32 (hdc, szString, lstrlen (szString), &s

          BeginPath (hdc) ;
```

```
        TextOut (hdc, (cxArea - size.cx) / 2, (cyArea - size.cy) / 2,
              szString, lstrlen (szString)) ;
        EndPath (hdc) ;
        lb.lbStyle    =      BS_SOLID ;
        lb.lbColor    =      RGB (255, 0, 0) ;
        lb.lbHatch    =      0 ;

        SelectObject (hdc, ExtCreatePen (PS_GEOMETRIC | PS_DO
                GetDeviceCaps (hdc, LOGPIXELSX) / 24, &lb, 0, NUI
        StrokePath (hdc) ;
        DeleteObject (SelectObject (hdc, GetStockObject (BLACK_
        SelectObject (hdc, GetStockObject (SYSTEM_FONT)) ;
        DeleteObject (hFont) ;
}
```

StrokePath31/24Windows
Windows 98



17-6 FONTOUT2

FillPathStrokeAndFillPath

StrokeAndFillPath17-12 FONTFILL

17-12 FONTFILL

```
FONTFILL.C

/*------------------------------------------------------------------------

  FONTFILL.C --         Using Path to Fill Font

                                                (c) Charles Petzold, 199

------------------------------------------------------------------------*/
```

```
#include <windows.h>
#include "..\\eztest\\ezfont.h"

TCHAR szAppName [] = TEXT ("FontFill") ;
TCHAR szTitle [] = TEXT ("FontFill: Using Path to Fill Font") ;
void PaintRoutine (HWND hwnd, HDC hdc, int cxArea, int cyArea
{
          static TCHAR szString [] = TEXT ("Filling") ;
          HFONT                          hFont ;
          SIZE                           size ;

          hFont = EzCreateFont (hdc, TEXT ("Times New Roman"), 1
          SelectObject (hdc, hFont) ;
          SetBkMode (hdc, TRANSPARENT) ;

          GetTextExtentPoint32 (hdc, szString, lstrlen (szString), &s
          BeginPath (hdc) ;
          TextOut (hdc, (cxArea - size.cx) / 2, (cyArea - size.cy) / 2,
                                                  szString, lstrlen (szString
          EndPath (hdc) ;
```

```
        SelectObject (hdc, CreateHatchBrush (HS_DIAGCROSS, RG
        SetBkColor (hdc, RGB (0, 0, 255)) ;
        SetBkMode (hdc, OPAQUE) ;

        StrokeAndFillPath (hdc) ;
        DeleteObject (SelectObject (hdc, GetStockObject (WHITE_
        SelectObject (hdc, GetStockObject (SYSTEM_FONT)) ;
        DeleteObject (hFont) ;
}
```

FONTFILLHS_DIAGCROSSTRANSPARENTOPAQUE17-7

SetBkMode

ALTERNATEWINDINGTrueType
ALTERNATE

17-7 FONTFILL

TrueType17-13 F

17-13 FONTCLIP

```
FONTCLIP.C

/*------------------------------------------------------------------------

  FONTCLIP.C --         Using Path for Clipping on Font

                                                (c) Charles Petzold, 199

------------------------------------------------------------------------*/

#include <windows.h>
```

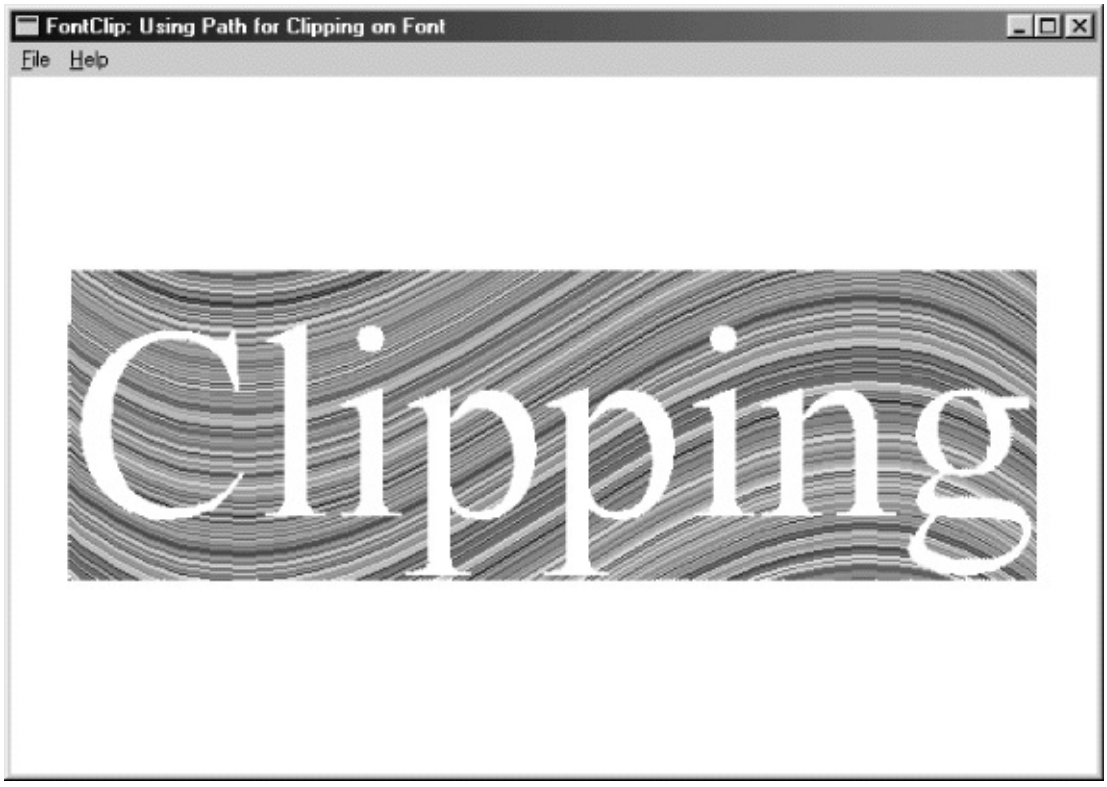
```
#include "..\\eztest\\ezfont.h"

TCHAR szAppName [] = TEXT ("FontClip") ;
TCHAR szTitle      [] = TEXT ("FontClip: Using Path for Clipping or

void PaintRoutine (HWND hwnd, HDC hdc, int cxArea, int cyArea)
{
          static TCHAR szString [] = TEXT ("Clipping") ;
          HFONT                          hFont ;
          int                           y, iOffset ;
    POINT                      pt [4] ;
          SIZE                        size ;

          hFont = EzCreateFont (hdc, TEXT ("Times New Roman"), 1
          SelectObject (hdc, hFont) ;

         GetTextExtentPoint32 (hdc, szString, lstrlen (szString), &si
          BeginPath (hdc) ;
          TextOut (hdc, (cxArea - size.cx) / 2, (cyArea - size.cy) / 2,
                                                szString, lstrlen (szString)) ;
```

```
        EndPath (hdc) ;

                                // Set clipping area

        SelectClipPath (hdc, RGN_COPY) ;

                                // Draw Bezier splines

iOffset = (cxArea + cyArea) / 4 ;

        for (y = -iOffset ; y < cyArea + iOffset ; y++)

        {

              pt[0].x = 0 ;

              pt[0].y = y ;


              pt[1].x = cxArea / 3 ;

              pt[1].y = y + iOffset ;


             pt[2].x = 2 * cxArea / 3 ;

              pt[2].y = y - iOffset ;


              pt[3].x = cxArea ;

              pt[3].y = y ;


              SelectObject (hdc, CreatePen (PS_SOLID, 1,

                                RGB (rand () % 256, rand () % 256, rar
```

```
            PolyBezier (hdc, pt, 4) ;
             DeleteObject (SelectObject (hdc, GetStockObject (BL
   }

   DeleteObject (SelectObject (hdc, GetStockObject (WHITE_BRU
  SelectObject (hdc, GetStockObject (SYSTEM_FONT)) ;
   DeleteObject (hFont) ;
}
```

SetBkModeSelectClipPath

FONTCLIPTRANSPARENTSetBkModeOPAQUE17-8



17-8 FONTCLIP

FONTCLIPSetBkModeTRANSPARENT

FONTDEMO

MetaFile

MetaFileMetaFileMetaFile

paint(draw)MetaFile

MetaFile

MetaFileMetaFileMetaFileMetaFile

MetaFileMetaFile

Microsoft WindowsMetaFileWindows
WindowsMetaFileMetaFileMetaFile

MetaFile

MetaFileWindowsMetaFileI/O

MetaFile

CreateMetaFileMetaFileWindowsMetaFileGDIMetaFileGDI
MetaFileMetaFileMetaFileMetaFileMetaFileGDI

CreateMetaFileNULLNULLMetaFile.WMFWindows
MetaFileMetaFile

18-1MetaFileWM_CREATEMetaFileWM_PAINT100

18-1　　MetaFile

MetaFile.C

```
/*------------------------------------------------------------------------
```

```
  MetaFile.C --        MetaFile Demonstration Program

                                                (c) Charles Petzold, 199
----------------------------------------------------------------------*/

#include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                                PSTR szCmdLine, int iCr
{
          static TCHAR szAppName [] = TEXT ("MetaFile") ;
          HWND                                          hwnd ;
          MSG                                           msg ;
          WNDCLASS                                      wndclass ;

          wndclass.style                                     = CS_HREDRA
          wndclass.lpfnWndProc                          = WndProc ;
          wndclass.cbClsExtra                           = 0 ;
          wndclass.cbWndExtra                           = 0 ;
          wndclass.hInstance                            = hInstance ;
          wndclass.hIcon                                     = LoadIcon (
```

```
     wndclass.hCursor                        = LoadCursor (N
     wndclass.hbrBackground          = (HBRUSH) GetStockO
     wndclass.lpszMenuName           = NULL ;
     wndclass.lpszClassName          = szAppName ;

     if (!RegisterClass (&wndclass))
{
          MessageBox (  NULL, TEXT ("This program requires W
                                        szAppName, MB_IC
          return 0 ;
}

     hwnd = CreateWindow (szAppName, TEXT ("MetaFile Dem
               WS_OVERLAPPEDWINDOW,
               CW_USEDEFAULT, CW_USEDEFAULT,
               CW_USEDEFAULT, CW_USEDEFAULT,
               NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
```

```
        UpdateWindow (hwnd) ;

        while (GetMessage (&msg, NULL, 0, 0))
   {
             TranslateMessage (&msg) ;
             DispatchMessage (&msg) ;
        }
        return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
        static HMetaFile            hmf ;
        static int                        cxClient, cyClient ;
        HBRUSH                                hBrush ;
        HDC                                     hdc, hdcMeta ;
        int                                     x, y ;
        PAINTSTRUCT                       ps ;
```

```
switch (message)
{
case   WM_CREATE:
            hdcMeta=      CreateMetaFile (NULL) ;
           hBrush=      CreateSolidBrush (RGB (0, 0, 255)) ;
            Rectangle     (hdcMeta, 0, 0, 100, 100) ;

            MoveToEx            (hdcMeta,     0,     0,     NULL)
            LineTo              (hdcMeta,     100,   100)   ;
            MoveToEx            (hdcMeta,     0,     100,   NULL
           LineTo               (hdcMeta,     100,   0)     ;

            SelectObject (hdcMeta, hBrush) ;
            Ellipse (hdcMeta, 20, 20, 80, 80) ;

            hmf = CloseMetaFile (hdcMeta) ;

            DeleteObject (hBrush) ;
```

```
                return 0 ;

        case   WM_SIZE:
                cxClient = LOWORD (lParam) ;
                cyClient = HIWORD (lParam) ;
                return 0 ;

        case   WM_PAINT:
                hdc = BeginPaint (hwnd, &ps) ;

                SetMapMode (hdc, MM_ANISOTROPIC) ;
                SetWindowExtEx (hdc, 1000, 1000, NULL) ;
                SetViewportExtEx (hdc, cxClient, cyClient, NULL)

                for (x = 0 ; x < 10 ; x++)
                for (y = 0 ; y < 10 ; y++)
                {
                                SetWindowOrgEx (hdc, -100 * x, -
                                PlayMetaFile (hdc, hmf) ;
```

```
                }
                EndPaint (hwnd, &ps) ;
                return 0 ;

        case   WM_DESTROY:
                DeleteMetaFile (hmf) ;
                PostQuitMessage (0) ;
                return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam)
}
```

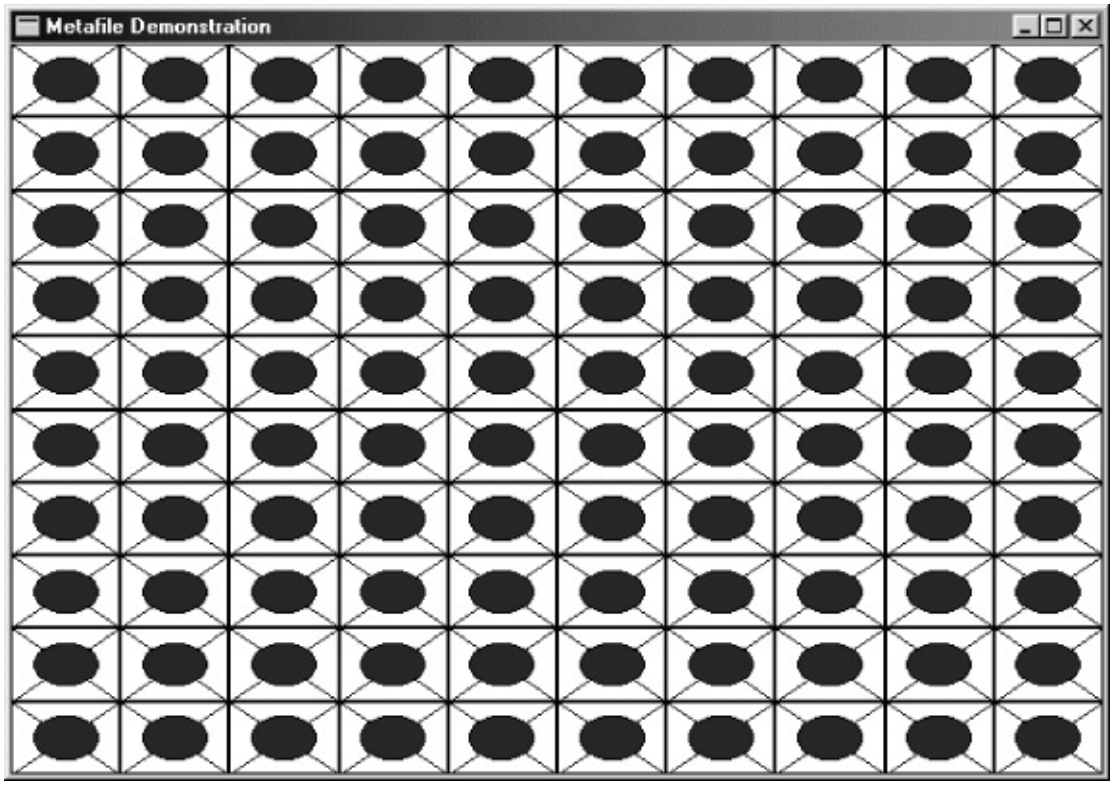
MetaFile4MetaFileCreateMetaFileWM_CREATENULLMetaFile
MetaFileMetaFileDCMetaFileCloseMetaFileMetaFileMetaFile

MetaFileGDIMoveToExLineToSelectObjectEllipse
MetaFile

WM_PAINTMetaFilePlayMetaFile100MetaFilePlayMetaFile
WM_CREATEMetaFileCreateMetaFileCloseMetaFile

GDIMetaFileWM_DESTROYDeleteMetaFile

MetaFile18-1



18-1 MetaFile

MetaFile

CreateMetaFileNULLMetaFileMetaFileMetaFileMetaFile

MetaFileMetaFileCreateMetaFileNULLWM_CREATEMetaFile
DeleteMetaFile

WM_PAINTGetMetaFileMetaFile

```
hmf = GetMetaFile (szFileName) ;
```

MetaFileWM_PAINTMetaFile

```
DeleteMetaFile (hmf) ;
```

WM_DESTROYMetaFileWM_CREATEWM_PAINT

```
DeleteFile (szFileName) ;
```

MetaFileMetaFile

```
hmf = SetMetaFileBitsEx (iSize, pData) ;
```

MetaFileSetMetaFileBitsExGetMetaFileBitsExMetaFile

MetaFile

MetaFileMetaFileMetaFileMetaFile

MetaFileMetaFileMM_ISOTROPICMM_ANISOTROPICMetaFile
MetaFileMetaFileMM_ISOTROPICMM_ANISOTROPIC
MetaFileMetaFileGDIMetaFileGDI

MetaFileMetaFileMetaFilePICTMetaFileMetaFile

MetaFilePICT16

```
typedef struct tagMetaFilePICT
{
        LONG mm ;                               // mapping mode
        LONG xExt ;                             // width of the MetaFile i
```

```
        LONG yExt ;                             // height of the MetaFile
        LONG hMF ;                              // handle to the MetaFil
}
MetaFilePICT ;
```

MM_ISOTROPICMM_ANISOTROPICxExtyExtmmMetaFile MetaFileMetaFileMetaFileGDIxy

MM_ISOTROPICMM_ANISOTROPICxExtyExt MM_ISOTROPICMM_ANISOTROPICMM_ISOTROPIC MM_ANISOTROPICMM_ISOTROPICMM_ANISOTROPIC SetWindowExtExSetViewportExtExSetWindowExtExSetViewportExtEx

MM_ISOTROPICMM_ANISOTROPICMetaFileMetaFileSetViewportExtEx MetaFileMetaFileMetaFilexExtyExtMetaFileMM_ISOTROPIC MM_ANISOTROPICMetaFileMetaFileGDI

MetaFileMetaFile

- MetaFilePICTmm
- MM_ISOTROPICMM_ANISOTROPICxExtyExtmm MM_ISOTROPICMM_ANISOTROPICMetaFile MM_ANISOTROPICxExtyExtxExtyExt0.01mm MM_HIMETRICMM_ISOTROPICxExtyExt
- MM_ISOTROPICMM_ANISOTROPICMetaFileSetWindowExtEx SetWindowOrgExMetaFileMetaFileMetaFileSetMapMode SetViewportExtExSetViewportOrgEx
- MetaFileMetaFileMetaFile

MetaFileMetaFileMM_ISOTROPICMM_ANISOTROPICMetaFile

```
hdcMeta = CreateMetaFile (NULL) ;

SetWindowExtEx (hdcMeta, ...) ;

SetWindowOrgEx (hdcMeta, ...) ;
```

MetaFileGDIMetaFileMetaFileMetaFile

```
hmf = CloseMetaFile (hdcMeta) ;
```

MetaFilePICT

```
GLOBALHANDLE              hGlobal ;

LPMetaFilePICT            pMFP ;


hGlobal= GlobalAlloc (GHND | GMEM_SHARE, sizeof (MetaFilePIC

pMFP = (LPMetaFilePICT) GlobalLock (hGlobal) ;
```

4

```
pMFP->mm   = MM_... ;

pMFP->xExt = ...  ;

pMFP->yExt = ...  ;

pMFP->hMF  = hmf  ;
```

```
GlobalUnlock (hGlobal) ;
```

MetaFile

```
OpenClipboard (hwnd) ;

EmptyClipboard () ;

SetClipboardData (CF_MetaFilePICT, hGlobal) ;

CloseClipboard () ;
```

hGlobalMetaFilehmfMetaFile

MetaFile

1. MetaFilemm
2. MM_ISOTROPICMM_ANISOTROPICxExtyExt
   MM_ISOTROPICMM_ANISOTROPICxExtyExt
3. MetaFile

MetaFile

```
OpenClipboard (hwnd) ;

hGlobal = GetClipboardData (CF_MetaFilePICT) ;

pMFP = (LPMetaFilePICT) GlobalLock (hGlobal) ;
```

mm

```
SaveDC (hdc) ;
```

```
SetMappingMode (pMFP->mm) ;
```

MM_ISOTROPICMM_ANISOTROPICxExtyExtLPtoDP

MM_ISOTROPICMM_ANISOTROPICxExtyExtxExtyExt
cxClientcyClientMetaFile

```
void PrepareMetaFile (     HDC hdc, LPMetaFilePICT pmfp,
                           int cxClient, int cyClient)
{
        int xScale, yScale, iScale ;
        SetMapMode (hdc, pmfp->mm) ;
        if (pmfp->mm == MM_ISOTROPIC || pmfp->mm == MM_
        {
                if (pmfp->xExt == 0)
                        SetViewportExtEx (hdc, cxClient, cyClie
                else if (pmfp->xExt > 0)
                         SetViewportExtEx (hdc,
                        pmfp->xExt *  GetDeviceCaps (hdc, HORZR
                    GetDeviceCaps (hdc, HORZSIZE) / 100),
                        pmfp->yExt *  GetDeviceCaps (hdc, VERTR
                    GetDeviceCaps (hdc, VERTSIZE) / 100), NULL
```

```
                    else if (pmfp->xExt < 0)
                    {
                        xScale = 100 *cxClient * GetDeviceCaps (h
                GetDeviceCaps (hdc, HORZRES) / -pmfp->xExt ;
                        lScale = 100 *cyClient * GetDeviceCaps (h
                GetDeviceCaps (hdc, VERTRES) / -pmfp->yExt ;
                        iScale = min (xScale, yScale) ;
  SetViewportExtEx (hdc, -pmfp->xExt * iScale * GetDeviceCaps
              GetDeviceCaps (hdc, HORZSIZE) / 100, -pmfp->yEx
* GetDeviceCaps (hdc, VERTRES) / GetDeviceCaps (hdc, VERTSIZ
            }
        }
}
```

xExtyExtMetaFilexExtyExt0.01mm GetDeviceCaps0.01mmMetaFilexExtyExtiScalecxClientcyClient

MetaFile

```
PlayMetaFile (pMFP->hMF) ;
RestoreDC (hdc, -1) ;
```

```
GlobalUnlock (hGlobal) ;

CloseClipboard () ;
```

MetaFileWindowsMetaFileMetaFile

MetaFile

MetaFile32Windows.EMF

MetaFileMetaFile

MetaFileMetaFile(EMF)MetaFileWindows
MetaFileGDI32

18-2EMF1MetaFile

18-2　　EMF1

```
EMF1.C

/*------------------------------------------------------------------------

  EMF1.C --     Enhanced MetaFile Demo #1

                                        (c) Charles Petzold, 1998

------------------------------------------------------------------------*/

#include <windows.h>
```

```
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                        LPSTR lpszCmdLine, int
{
        static TCHAR szAppName[] = TEXT ("EMF1") ;
        HWND                 hwnd ;
        MSG                  msg ;
        WNDCLASS             wndclass ;

        wndclass.style                  = CS_HREDRA
        wndclass.lpfnWndProc              = WndProc ;
        wndclass.cbClsExtra               = 0 ;
        wndclass.cbWndExtra               = 0 ;
        wndclass.hInstance                = hInstance ;
        wndclass.hIcon                  = LoadIcon (NULL, IDI_
        wndclass.hCursor              = LoadCursor (NULL, IDC_A
  wndclass.hbrBackground          = GetStockObject (WHITE_BI
        wndclass.lpszMenuName           = NULL ;
```

```
        wndclass.lpszClassName          = szAppName ;

        if (!RegisterClass (&wndclass))
        {
                MessageBox (  NULL, TEXT ("This program requi
                                                szAppName, M
                return 0 ;
}

        hwnd = CreateWindow (szAppName, TEXT ("Enhanced Me
                        WS_OVERLAPPEDWINDOW,
                       CW_USEDEFAULT, CW_USEDEFAULT,
                        CW_USEDEFAULT, CW_USEDEFAULT,
                        NULL, NULL, hInstance, NULL) ;

        ShowWindow (hwnd, nCmdShow) ;
        UpdateWindow (hwnd) ;

        while (GetMessage (&msg, NULL, 0, 0))
```

```
        {
                    TranslateMessage (&msg) ;
                    DispatchMessage (&msg) ;
    }
        return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
        static HENHMetaFile   hemf ;
        HDC                                     hdc, hdcEM
        PAINTSTRUCT                         ps ;
        RECT                                    rect ;

        switch (message)
    {
        case   WM_CREATE:
                    hdcEMF = CreateEnhMetaFile (NULL, NULL, NUL

                    Rectangle    (hdcEMF, 100, 100, 200, 200) ;
```

```
                MoveToEx            (hdcEMF, 100, 100, NULL) ;
                LineTo              (hdcEMF, 200, 200) ;

                MoveToEx            (hdcEMF, 200, 100, NULL) ;
                LineTo              (hdcEMF, 100, 200) ;

                hemf = CloseEnhMetaFile (hdcEMF) ;
                return 0 ;

        case   WM_PAINT:
                hdc = BeginPaint (hwnd, &ps) ;

                GetClientRect (hwnd, &rect) ;

                rect.left           =                   rect.right       / 4
                rect.right           = 3 * rect.right           / 4 ;
                rect.top             =                    rect.bottom
                rect.bottom          = 3 * rect.bottom          / 4 ;
```

```
                PlayEnhMetaFile (hdc, hemf, &rect) ;

                EndPaint (hwnd, &ps) ;
                return 0 ;

        case   WM_DESTROY:
                DeleteEnhMetaFile (hemf) ;

                PostQuitMessage (0) ;
                return 0 ;
   }
        return DefWindowProc (hwnd, message, wParam, lParam)
}
```

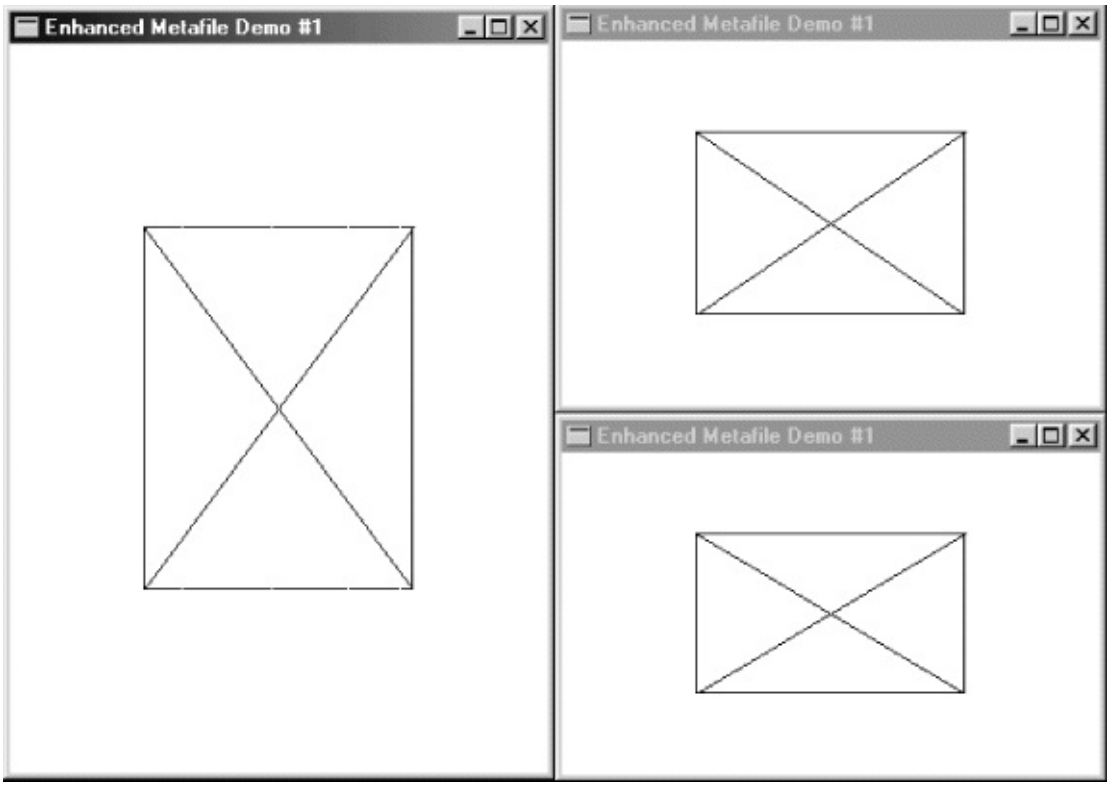
EMF1WM_CREATECreateEnhMetaFileMetaFile4NULL4NULL

CreateMetaFileCreateEnhMetaFileMetaFile

CloseEnhMetaFileMetaFileHENHMetaFile

WM_PAINTEMF1RECT4EMF1PlayEnhMetaFileMetaFile
RECT

MetaFileGDIMetaFile100MetaFileGDIPlayEnhMetaFile
EMF1Windows18-2



18-2 EMF1

WM_DESTROYEMF1DeleteEnhMetaFileMetaFile

EMF1

MetaFile

PlayEnhMetaFile18-2MetaFile

…

MetaFileWindowsMetaFile

MetaFileMetaFileMetaFile18-3EMF2MetaFile

18-3　EMF2

```
EMF2.C
/*------------------------------------------------------------------------
  EMF2.C --     Enhanced MetaFile Demo #2
                                        (c) Charles Petzold, 1998
------------------------------------------------------------------------*/
#include <windows.h>
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                                LPSTR lpszCmdLine, int
{
          static TCHAR         szAppName[] = TEXT ("EMF2") ;
          HWND                                  hwnd ;
          MSG                                  msg ;
          WNDCLASS                            wndclass ;
```

```
     wndclass.style                    = CS_HREDRAW |
     wndclass.lpfnWndProc              = WndProc ;
     wndclass.cbClsExtra               = 0 ;
     wndclass.cbWndExtra               = 0 ;
     wndclass.hInstance                = hInstance ;
     wndclass.hIcon                    = LoadIcon (NULL, IDI_
     wndclass.hCursor                  = LoadCursor (NULL, IDC_A
     wndclass.hbrBackground            = GetStockObject (WH
     wndclass.lpszMenuName             = NULL ;
     wndclass.lpszClassName            = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (  NULL, TEXT ("This program requires W
                                    szAppName, MB_IC
          return 0 ;
     }
```

```
        hwnd = CreateWindow (szAppName, TEXT ("Enhanced Me
                    WS_OVERLAPPEDWINDOW,
                    CW_USEDEFAULT, CW_USEDEFAULT,
                    CW_USEDEFAULT, CW_USEDEFAULT,
                   NULL, NULL, hInstance, NULL) ;

       ShowWindow (hwnd, nCmdShow) ;
        UpdateWindow (hwnd) ;

        while (GetMessage (&msg, NULL, 0, 0))
        {
             TranslateMessage (&msg) ;
             DispatchMessage (&msg) ;
        }
        return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
```

```
HDC                         hdc, hdcEMF ;
HENHMetaFile hemf ;
PAINTSTRUCT         ps ;
RECT                        rect ;

switch (message)
{
case   WM_CREATE:
hdcEMF = CreateEnhMetaFile (NULL, TEXT ("emf2.emf"),
TEXT ("EMF2\0EMF Demo #2\0")) ;
            if (!hdcEMF)
                  return 0 ;
            Rectangle    (hdcEMF, 100, 100, 200, 200) ;
            MoveToEx           (hdcEMF, 100, 100, NULL) ;
            LineTo             (hdcEMF, 200, 200) ;

            MoveToEx           (hdcEMF, 200, 100, NULL) ;
```

```
                LineTo              (hdcEMF, 100, 200) ;

                hemf = CloseEnhMetaFile (hdcEMF) ;

                DeleteEnhMetaFile (hemf) ;
                return 0 ;

        case   WM_PAINT:
                hdc = BeginPaint (hwnd, &ps) ;

                GetClientRect (hwnd, &rect) ;

                rect.left           =        rect.right     / 4 ;
                rect.right    = 3 * rect.right          / 4 ;
                rect.top      =       rect.bottom         / 4 ;
                rect.bottom   = 3 *  rect.bottom          / 4 ;

                if (hemf = GetEnhMetaFile (TEXT ("emf2.emf")))
                {
```

```
                        PlayEnhMetaFile (hdc, hemf, &rect) ;

                        DeleteEnhMetaFile (hemf) ;

                }

                EndPaint (hwnd, &ps) ;

                return 0 ;


        case   WM_DESTROY:

                PostQuitMessage (0) ;

                return 0 ;

       }

        return DefWindowProc (hwnd, message, wParam, lParam)

}
```

EMF1CreateEnhMetaFileNULLEMF2NULLGDIMetaFile
NULLGDI

CreateEnhMetaFileNULLEMF1NULLEMF2NULLMetaFile
EMF2EMF2.EMFMetaFile

RECT0.01mmMetaFileMetaFileWindows
GDINULLGDI

MetaFileNULLNULLC\0NULLLoonyCad
V6.4\0Flying Frogs\0\0CNULLEMF2\0

MetaFileEMF1EMF2CreateEnhMetaFileGDICloseEnhMetaFile
MetaFile

WM_CREATEEMF2EMF1MetaFileDeleteEnhMetaFile
MetaFileMetaFileDeleteFileMetaFileEMF1

MetaFileEMF2WM_PAINTGetEnhMetaFileMetaFileMetaFile
EMF1EMF2PlayEnhMetaFileMetaFilePlayEnhMetaFileEMF1
EMF2WM_PAINTMetaFileWM_PAINTEMF2MetaFile

MetaFileMetaFileMetaFile

EMF2MetaFile18-3EMF2.EMF

0000	01 00 00 00 88 00 00 00 64 00 00 00 64 00 00 00	
0010	C8 00 00 00 C8 00 00 00 35 0C 00 00 35 0C 00 00	
0020	6A 18 00 00 6A 18 00 00 20 45 4D 46 00 00 01 00	
0030	F4 00 00 00 07 00 00 00 01 00 00 00 12 00 00 00	.
0040	64 00 00 00 00 00 00 00 00 00 04 00 00 00 03 00 00	d
0050	40 01 00 00 F0 00 00 00 00 00 00 00 00 00 00 00	@
0060	00 00 00 00 45 00 4D 00 46 00 32 00 00 00 45 00	
0070	4D 00 46 00 20 00 44 00 65 00 6D 00 6F 00 20 00	
0080	23 00 32 00 00 00 00 00 2B 00 00 00 18 00 00 00	
0090	63 00 00 00 63 00 00 00 C6 00 00 00 C6 00 00 00	
00A0	1B 00 00 00 10 00 00 00 64 00 00 00 64 00 00 00	
00B0	36 00 00 00 10 00 00 00 C8 00 00 00 C8 00 00 00	

```
00C0       1B 00 00 00 10 00 00 00 C8 00 00 00 64 00 00 00
00D0       36 00 00 00 10 00 00 00 64 00 00 00 C8 00 00 00
00E0       0E 00 00 00 14 00 00 00 00 00 00 00 10 00 00 00
00F0       14 00 00 00                                     ....
```

18-3 EMF2.EMF

18-3MetaFileEMF2Microsoft Windows NT 41024×768Windows MetaFile12MetaFile

MetaFileMetaFileMetaFileENHMETARECORDWINGDI.H

```
typedef struct tagENHMETARECORD
{
        DWORD iType ;                       // record type
         DWORD nSize ;                       // record size
         DWORD dParm [1] ;                  // parameters
}
ENHMETARECORD ;
```

iTypeWINGDI.HEMR_nSizeiTypenSizedParm

18-30x000000010x000000881361EMR_HEADER0x0088

5EMF2MetaFile5GDI0x00880x0000002BEMR_RECTANGLE RectangleMetaFile0x00000018 24432Rectangle5 MetaFileEMF2(100100)(200200)420x00000063 0x000000C6 (198)EMF2Windows 98MetaFile0x00000064 0x000000C7(199)RectangleMetaFileWindows

4162MoveToEx (0x0000001BEMR_MOVETOEX)LineTo EMR_LINETO)MetaFile

MetaFile200x0000000EEMR_EOFend of file

MetaFileENHMETAHEADER

```
typedef struct tagENHMETAHEADER
{
        DWORD iType ;       // EMR_HEADER = 1
        DWORD nSize ;       // structure size
        RECTL rclBounds ;    // bounding rectangle in pixels
        RECTL rclFrame ;     // size of image in 0.01 millimeters
        DWORD dSignature ;  // ENHMETA_SIGNATURE = " EMF"
        DWORD nVersion ;    // 0x00010000
        DWORD nBytes ;     // file size in bytes
        DWORD nRecords ;   // total number of records
       WORD  nHandles ;   // number of handles in handle table
        WORD  sReserved ;
```

```
        DWORD nDescription ;    // character length of description
        DWORD offDescription ;   // offset of description string in
        DWORD nPalEntries ;     // number of entries in palette
        SIZEL szlDevice ;        // device resolution in pixels
        SIZEL szlMillimeters ;          // device resolution in millimet
        DWORD cbPixelFormat ;  // size of pixel format
        DWORD offPixelFormat ;  // offset of pixel format
        DWORD bOpenGL ;      // FALSE if no OpenGL records
}
ENHMETAHEADER ;
```

MetaFileWindows MetaFileMetaFileI/OM
GetEnhMetaFileHeader

```
GetEnhMetaFileHeader (hemf, cbSize, &emh) ;
```

MetaFileENHMETAHEADERGetEnh-MetaFileDescription

ENHMETAHEADER100MF2.EMFMetaFile0x88136
Windows 98MetaFileENHMETAHEADER312

rclBoundsRECT(100,100)(200,200)

rclFrame0.01(0x0C35,0x0C35)(0x186A,0x186A)(3125,3125)
(6250,6250)

dSignatureENHMETA_SIGNATURE0x464D4520IntelASCII

" EMF"dVersion0x00010000

nBytes0x000000F4MetaFilenRecords0x000000075GDI

nHandles0x0001MetaFileGDIMetaFile

0x00000012180x00000064MetaFile

nPalEntriesMetaFile

SIZEL32cxcyszlDeviceMetaFile0x0040szlMillimeters
0x0050MetaFilereference
NULLGDIEMF2MetaFileWindows

GDIGetDeviceCapsEMF2.EMFszlDevice0x0400×0x03001024×768
HORZRESVERTRESGetDeviceCapsszlMillimeters0x140×0xF0
320×240HORZSIZEVERTSIZEGetDeviceCaps

0.3125mm0.3125mmGDIrclFrame

MetaFileENHMETAHEADERCreateEnhMetaFileNULL
EMF2NULLEMF Demo #218Unicode36MetaFileWi
NTWindows 98Unicode

MetaFileGDI

GDIMetaFileGDI18-4
GetVersionWindows 98Windows NT

18-4 EMF3

EMF3.C

```
/*------------------------------------------------------------------

  EMF3.C --     Enhanced MetaFile Demo #3

                                (c) Charles Petzold, 1998
```

```
------------------------------------------------------------------------*/
#include <windows.h>
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                          PSTR szCmdLine, int iCr
{
     static TCHAR szAppName[] = TEXT ("EMF3") ;
     HWND                                 hwnd ;
     MSG                                  msg ;
     WNDCLASS                             wndclass ;

     wndclass.style                           = CS_HREDRA
     wndclass.lpfnWndProc                  = WndProc ;
     wndclass.cbClsExtra                   = 0 ;
     wndclass.cbWndExtra                   = 0 ;
     wndclass.hInstance                    = hInstance ;
     wndclass.hIcon                             = LoadIcon (
     wndclass.hCursor                        = LoadCursor (N
```

```
        wndclass.hbrBackground      = GetStockObject (WHITE_BI
        wndclass.lpszMenuName       = NULL ;
        wndclass.lpszClassName      = szAppName ;

        if (!RegisterClass (&wndclass))
        {
             MessageBox (  NULL, TEXT ("This program requires W
                                          szAppName, MB_IC
             return 0 ;
}

hwnd = CreateWindow (szAppName, TEXT ("Enhanced MetaFi
                    WS_OVERLAPPEDWINDOW,
                    CW_USEDEFAULT, CW_USEDEFAULT,
                    CW_USEDEFAULT, CW_USEDEFAULT,
                   NULL, NULL, hInstance, NULL) ;

        ShowWindow (hwnd, iCmdShow) ;
        UpdateWindow (hwnd) ;
```

```
        while (GetMessage (&msg, NULL, 0, 0))
        {
             TranslateMessage (&msg) ;
            DispatchMessage (&msg) ;
        }
   return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
        LOGBRUSH                    lb ;
        HDC                           hdc, hdcEMF ;
        HENHMetaFile         hemf ;
        PAINTSTRUCT                ps ;
        RECT                             rect ;
        switch (message)
   {
        case   WM_CREATE:
```

```
hdcEMF = CreateEnhMetaFile (NULL, TEXT ("emf3.emf"),
TEXT ("EMF3\0EMF Demo #3\0")) ;

          SelectObject (hdcEMF, CreateSolidBrush (RGB (

          lb.lbStyle = BS_SOLID ;
          lb.lbColor = RGB (255, 0, 0) ;
          lb.lbHatch = 0 ;

          SelectObject (hdcEMF,
          ExtCreatePen (PS_SOLID | PS_GEOMETRIC, 5, &l

          if (GetVersion () & 0x80000000) // Windows 98
          Rectangle (hdcEMF, 100, 100, 201, 201) ;
          else
            // Windows NT
                        Rectangle (hdcEMF, 101, 101, 20

          MoveToEx          (hdcEMF, 100, 100, NULL) ;
```

```
                LineTo              (hdcEMF, 200, 200) ;

                MoveToEx            (hdcEMF, 200, 100, NULL) ;
                LineTo              (hdcEMF, 100, 200) ;

                DeleteObject (SelectObject (hdcEMF, GetStockO
                DeleteObject (SelectObject (hdcEMF, GetStockO

                hemf = CloseEnhMetaFile (hdcEMF) ;

                        DeleteEnhMetaFile (hemf) ;
                        return 0 ;

                case   WM_PAINT:
                        hdc = BeginPaint (hwnd, &ps) ;

                        GetClientRect (hwnd, &rect) ;
```

```
                    rect.left           =     rect.right       / 4
                    rect.right          = 3 * rect.right       /
                    rect.top            =     rect.bottom      /
                    rect.bottom         = 3 * rect.bottom

                    hemf = GetEnhMetaFile (TEXT ("emf3.

                    PlayEnhMetaFile (hdc, hemf, &rect) ;
                   DeleteEnhMetaFile (hemf) ;
                    EndPaint (hwnd, &ps) ;
                    return 0 ;

               case   WM_DESTROY:
                    PostQuitMessage (0) ;
                    return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam)
}
```

CreateEnhMetaFileGDIMetaFileGDIGDIGDI

EMF3CreateSolidBrushExtCreatePen`GDIMetaFileGDI
MetaFile

SelectObjectGDIMetaFileGDIGDIMetaFileSelectObject
EMF3.EMF18-4

```
0000    01 00 00 00 88 00 00 00 60 00 00 00 60 00 00 00
0010    CC 00 00 00 CC 00 00 00 B8 0B 00 00 B8 0B 00 00
0020    E7 18 00 00 E7 18 00 00 20 45 4D 46 00 00 01 00
0030    88 01 00 00 0F 00 00 00 03 00 00 00 12 00 00 00
0040    64 00 00 00 00 00 00 00 00 00 04 00 00 00 03 00 00
0050    40 01 00 00 F0 00 00 00 00 00 00 00 00 00 00 00
0060    00 00 00 00 45 00 4D 00 46 00 33 00 00 00 45 00
0070    4D 00 46 00 20 00 44 00 65 00 6D 00 6F 00 20 00
0080    23 00 33 00 00 00 00 00 27 00 00 00 18 00 00 00
0090    01 00 00 00 00 00 00 00 00 00 FF 00 00 00 00 00
00A0    25 00 00 00 0C 00 00 00 01 00 00 00 5F 00 00 00
00B0    34 00 00 00 02 00 00 00 34 00 00 00 00 00 00 00
00C0    34 00 00 00 00 00 00 00 00 00 01 00 05 00 00 00
00D0    00 00 00 00 FF 00 00 00 00 00 00 00 00 00 00 00
```

00E0	25 00 00 00 0C 00 00 00 02 00 00 00 2B 00 00 00
00F0	18 00 00 00 63 00 00 00 63 00 00 00 C6 00 00 00
0100	C6 00 00 00 1B 00 00 00 10 00 00 00 64 00 00 00
0110	64 00 00 00 36 00 00 00 10 00 00 00 C8 00 00 00
0120	C8 00 00 00 1B 00 00 00 10 00 00 00 C8 00 00 00
0130	64 00 00 00 36 00 00 00 10 00 00 00 64 00 00 00
0140	C8 00 00 00 25 00 00 00 0C 00 00 00 07 00 00 80
0150	28 00 00 00 0C 00 00 00 02 00 00 00 25 00 00 00
0160	0C 00 00 00 00 00 00 80 28 00 00 00 0C 00 00 00
0170	01 00 00 00 0E 00 00 00 14 00 00 00 00 00 00 00
0180	10 00 00 00 14 00 00 00

18-4 EMF3.EMF

MetaFileEMF2.EMFEMF3.EMFrclBoundsEMF2.EMF(0x64,0x64)
(0xC8,0xC8)EMF3.EMF(0x60,0x60)(0xCC,0xCC)rclFrame
0.01mm

EMF2.EMFnBytes0x0030MetaFile0xFAEMF3.EMF0x0188
EMF2.EMFMetaFile75GDIEMF3.EMF1584SelectObject
DeleteObject

nHandles0x0038GDIMetaFilePlatform
EMF2.EMF1EMF3.EMF3

0x00880x27EMR_CREATE-BRUSHINDIRECTMetaFile
CreateBrushIndirectLOGBRUSH0x1824

MetaFileGDI14MetaFile0x009034LOGBRUSH3
0x00000000BS_SOLIDlbStyle0x00FF0000lbColor
0x00000000lbHatch

EMF3.EMF0x00A00x25EMR_SELECTOBJECTSelectObjectMetaFile
0x0C120x01GDI

EMF3.EMF0x00AC0x5FEMR_EXTCREATEPEN0x34524
0x02MetaFileGDI

EMR_EXTCREATEPEN400x340x000x340x000x00010000
PS_SOLID (0x00000000)PS_GEOMETRIC (0x00010000)5
ExtCreatePen30

EMR_EXTCREATEPEN52LOGBRUSH30EMF3.EMF

EMF3.EMF12SelectObject5EMF2.EMF0x2B(EMR_RECTANGLE)
0x1B (EMR_MOVETOEX)0x36 (EMR_LINETO)

0x25(EMR_SELECTOBJECT)0x28 (EMR_DELETEO
0x800000070x800000000x07BLACK_PEN0x00
(WHITE_BRUSH)

DeleteObject21MetaFileDeleteObjectGDIMetaFile

MetaFile0x0EEMF_EOF

GDIMetaFileGDIEMR_CREATEBRUSHINDIRECT
EMR_EXTCREATEPEN1EMR_SELECTOBJECTMetaFile
EMR_SELECTOBJECT

MetaFile

MetaFile18-5 EMF4

```
EMF4.C

/*-------------------------------------------------------------------------
  EMF4.C --     Enhanced MetaFile Demo #4
                                        (c) Charles Petzold, 1998
-------------------------------------------------------------------------*/

#define OEMRESOURCE

#include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                                PSTR szCmdLine, int iCr
{
          static TCHAR szAppName[] = TEXT ("EMF4") ;
         HWND                         hwnd ;
          MSG                         msg ;
          WNDCLASS                    wndclass ;

          wndclass.style                                    = CS_HREDRA
```

```
     wndclass.lpfnWndProc                   = WndProc ;
     wndclass.cbClsExtra                    = 0 ;
     wndclass.cbWndExtra                    = 0 ;
    wndclass.hInstance                     = hInstance ;
     wndclass.hIcon                                 = LoadIcon (
     wndclass.hCursor                           = LoadCursor (N
     wndclass.hbrBackground             = GetStockObject (WH
     wndclass.lpszMenuName              = NULL ;
     wndclass.lpszClassName             = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (  NULL, TEXT ("This program requires W
                                          szAppName, MB_IC
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, TEXT ("Enhanced M
```

```
                    WS_OVERLAPPEDWINDOW,
                    CW_USEDEFAULT, CW_USEDEFAULT,
                   CW_USEDEFAULT, CW_USEDEFAULT,
                   NULL, NULL, hInstance, NULL) ;

          ShowWindow (hwnd, iCmdShow) ;
          UpdateWindow (hwnd) ;
          while (GetMessage (&msg, NULL, 0, 0))
          {
                TranslateMessage (&msg) ;
               DispatchMessage (&msg) ;
          }
    return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
          BITMAP                    bm ;
          HBITMAP                    hbm ;
```

```
HDC                     hdc, hdcEMF, hdcMem ;
HENHMetaFile            hemf ;
PAINTSTRUCT             ps ;
RECT                    rect ;

switch (message)
{
case   WM_CREATE:
hdcEMF = CreateEnhMetaFile (NULL, TEXT ("emf4.emf"),
TEXT ("EMF4\0EMF Demo #4\0")) ;

          hbm = LoadBitmap (NULL, MAKEINTRESOURCE

          GetObject (hbm, sizeof (BITMAP), &bm) ;

          hdcMem = CreateCompatibleDC (hdcEMF) ;

          SelectObject (hdcMem, hbm) ;
```

```
                StretchBlt (hdcEMF,100,100,100,100,
            hdcMem,0,0,bm.bmWidth, bm.bmHeight, SRCCOPY) ;

                DeleteDC (hdcMem) ;
             DeleteObject (hbm) ;

                hemf = CloseEnhMetaFile (hdcEMF) ;

                DeleteEnhMetaFile (hemf) ;
                return 0 ;

        case   WM_PAINT:
                hdc = BeginPaint (hwnd, &ps) ;

                GetClientRect (hwnd, &rect) ;
               rect.left            =     rect.right            / 4 ;
               rect.right           = 3 * rect.right            / 4 ;
                rect.top                  =     rect.bottom     / 4 ;
```

```
                rect.bottom         = 3 * rect.bottom         / 4


                hemf = GetEnhMetaFile (TEXT ("emf4.emf")) ;


                PlayEnhMetaFile (hdc, hemf, &rect) ;
                DeleteEnhMetaFile (hemf) ;
                EndPaint (hwnd, &ps) ;
                return 0 ;


        case   WM_DESTROY:
                PostQuitMessage (0) ;
                return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam)
}
```

EMF4OEM_CLOSECreateCompatibleDCMetaFile
SelectObjectBitBltStretchBlt

EMF4GetObjectSelectObject

MetaFileGDIStretchBltGDIMetaFileEMF4.EMF18-5

0000	01 00 00 00 88 00 00 00 64 00 00 00 64 00 00 00
0010	C7 00 00 00 C7 00 00 00 35 0C 00 00 35 0C 00 00
0020	4B 18 00 00 4B 18 00 00 20 45 4D 46 00 00 01 00
0030	F0 0E 00 00 03 00 00 00 01 00 00 00 12 00 00 00
0040	64 00 00 00 00 00 00 00 00 00 04 00 00 00 03 00 00
0050	40 01 00 00 F0 00 00 00 00 00 00 00 00 00 00 00
0060	00 00 00 00 45 00 4D 00 46 00 34 00 00 00 45 00
0070	4D 00 46 00 20 00 44 00 65 00 6D 00 6F 00 20 00
0080	23 00 34 00 00 00 00 00 4D 00 00 00 54 0E 00 00
0090	64 00 00 00 64 00 00 00 C7 00 00 00 C7 00 00 00
00A0	64 00 00 00 64 00 00 00 64 00 00 00 64 00 00 00
00B0	20 00 CC 00 00 00 00 00 00 00 00 00 00 00 80 3F
00C0	00 00 00 00 00 00 00 00 00 00 80 3F 00 00 00 00
00D0	00 00 00 00 FF FF FF 00 00 00 00 00 6C 00 00 00
00E0	28 00 00 00 94 00 00 00 C0 0D 00 00 28 00 00 00
00F0	16 00 00 00 28 00 00 00 28 00 00 00 16 00 00 00
0100	01 00 20 00 00 00 00 00 C0 0D 00 00 00 00 00 00
0110	00 00 00 00 00 00 00 00 00 00 00 00 C0 C0 C0 00
0120	C0 C0 C0 00 C0 C0 C0 00 C0 C0 C0 00 C0 C0 C0 00

```
. . . .
0ED0       C0 C0 C0 00 C0 C0 C0 00 C0 C0 C0 00 0E 00 00 00
0EE0       14 00 00 00 00 00 00 00 10 00 00 00 14 00 00 00
```

18-5 EMF4.EMF

MetaFile30x0E540x4DEMR_STRETCHBLT

GDIEMF4.CMetaFile

GDIDIBDIBMetaFileGDIStretchDIBitsStretchBltGDI CreateDIBitmapDIBStretchBlt

EMR_STRETCHBLTMetaFile0x0088DIBMetaFile0x00F40x0EDC DIBBITMAPINFOHEADER400x011C224032DIB4

MetaFile

MetaFileMetaFile18-6 E

18-6 EMF5

```
EMF5.C
/*------------------------------------------------------------------------
  EMF5.C --     Enhanced MetaFile Demo #5
                                        (c) Charles Petzold, 1998
------------------------------------------------------------------------*/
```

```
#include <windows.h>
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                        PSTR szCmdLine, int iCm
{
        static TCHAR szAppName[] = TEXT ("EMF5") ;
        HWND                         hwnd ;
        MSG                          msg ;
        WNDCLASS                     wndclass ;

        wndclass.style                              = CS_HREDRA
        wndclass.lpfnWndProc                    = WndProc ;
        wndclass.cbClsExtra                     = 0 ;
        wndclass.cbWndExtra                     = 0 ;
        wndclass.hInstance                      = hInstance ;
        wndclass.hIcon                                  = LoadIcon (
        wndclass.hCursor                            = LoadCursor (N
        wndclass.hbrBackground              = GetStockObject (WH
```

```
     wndclass.lpszMenuName        = NULL ;
     wndclass.lpszClassName       = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (  NULL, TEXT ("This program requires W
                                        szAppName, MB_IC
          return 0 ;
}

     hwnd = CreateWindow (szAppName, TEXT ("Enhanced Me
               WS_OVERLAPPEDWINDOW,
               CW_USEDEFAULT, CW_USEDEFAULT,
               CW_USEDEFAULT, CW_USEDEFAULT,
               NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;
```

```
        while (GetMessage (&msg, NULL, 0, 0))
        {
             TranslateMessage (&msg) ;
             DispatchMessage (&msg) ;
        }
        return msg.wParam ;
}
int CALLBACK EnhMetaFileProc (     HDC hdc, HANDLETABLE * pH
                        CONST ENHMETARECORD * pEmfRecord,
                          int iHandles, LPARAM pData)
{
        PlayEnhMetaFileRecord (hdc, pHandleTable, pEmfRecord,
        return TRUE ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
        HDC                 hdc ;
        HENHMetaFile        hemf ;
```

```
     PAINTSTRUCT          ps ;
     RECT                 rect ;

     switch (message)
{
     case   WM_PAINT:
               hdc = BeginPaint (hwnd, &ps) ;

               GetClientRect (hwnd, &rect) ;

               rect.left                =     rect.right            / 4 ;
               rect.right               = 3 * rect.right            / 4
               rect.top                 =     rect.bottom           /
               rect.bottom                 = 3 * rect.bottom

          hemf = GetEnhMetaFile (TEXT ("..\\emf3\\emf3.emf")

               EnumEnhMetaFile (hdc, hemf, EnhMetaFileProc,
```

```
                    DeleteEnhMetaFile (hemf) ;

                    EndPaint (hwnd, &ps) ;

                    return 0 ;


          case   WM_DESTROY:

                    PostQuitMessage (0) ;

                    return 0 ;

          }

          return DefWindowProc (hwnd, message, wParam, lParam)
}
```

EMF3EMF3.EMFEMF3Visual
PlayEnhMetaFileEMF5EnumEnhMetaFilePlayEnhMetaFile

```
PlayEnhMetaFile (hdc, hemf, &rect) ;
```

MetaFileMetaFileRECTMetaFile

EnumEnhMetaFile53PlayEnhMetaFileRECT

EnumEnhMetaFileEnhMetaFileProcNULL

EnumEnhMetaFileMetaFileGDIEnhMetaFileProcTRUE
FALSE

54PlayEnhMetaFileRecordGDI

EMF5EnumEnhMetaFilePlayEnhMetaFileRecordEMF3PlayEnhMetaFile EMF5MetaFileMetaFile

GDIEnumEnhMetaFilePlayEnhMetaFileRecord

ENHMETARECORDMetaFileMetaFile

PlayEnhMetaFileRecordEMF5.CPlayEnhMetaFileRecord

```
if (pEmfRecord->iType != EMR_LINETO)
```

```
if (pEmfRecord->iType != EMR_SELECTOBJECT)
```

GDIMetaFile

MetaFile18-7 EMF6

18-7 EMF6

```
EMF6.C

/*------------------------------------------------------------------------

  EMF6.C --     Enhanced MetaFile Demo #6

                                    (c) Charles Petzold, 1998

------------------------------------------------------------------------*/

#include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
```

```
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                        PSTR lpszCmdLine, int i
{
     static TCHAR szAppName[] = TEXT ("EMF6") ;
     HWND                 hwnd ;
     MSG                  msg ;
     WNDCLASS             wndclass ;

     wndclass.style                = CS_HREDRAW | CS_V
     wndclass.lpfnWndProc             = WndProc ;
     wndclass.cbClsExtra              = 0 ;
     wndclass.cbWndExtra              = 0 ;
     wndclass.hInstance               = hInstance ;
     wndclass.hIcon                = LoadIcon (NULL, IDI_
     wndclass.hCursor           = LoadCursor (NULL, IDC_A
     wndclass.hbrBackground         = GetStockObject (WH
     wndclass.lpszMenuName          = NULL ;
     wndclass.lpszClassName         = szAppName ;
```

```
     if (!RegisterClass (&wndclass))
     {
          MessageBox (  NULL, TEXT ("This program requires W
                                                  szAppName, MB_IC
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, TEXT ("Enhanced Me
               WS_OVERLAPPEDWINDOW,
               CW_USEDEFAULT, CW_USEDEFAULT,
               CW_USEDEFAULT, CW_USEDEFAULT,
               NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;
     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
```

```
            DispatchMessage (&msg) ;
        }
        return msg.wParam ;
}

int CALLBACK EnhMetaFileProc (     HDC hdc, HANDLETABLE * pH
                        CONST ENHMETARECORD * pEmfRecord,
                        int iHandles, LPARAM pData)
{
        ENHMETARECORD * pEmfr ;
        pEmfr = (ENHMETARECORD *) malloc (pEmfRecord->nSiz
        CopyMemory (pEmfr, pEmfRecord, pEmfRecord->nSize) ;
        if (pEmfr->iType == EMR_RECTANGLE)
                pEmfr->iType = EMR_ELLIPSE ;
        PlayEnhMetaFileRecord (hdc, pHandleTable, pEmfr, iHand
        free (pEmfr) ;
        return TRUE ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
```

```
{
        HDC                 hdc ;
        HENHMetaFile        hemf ;
        PAINTSTRUCT         ps ;
        RECT                rect ;

        switch (message)
        {
        case   WM_PAINT:
                    hdc = BeginPaint (hwnd, &ps) ;

                    GetClientRect (hwnd, &rect) ;

                    rect.left                =     rect.right      / 4 ;
                    rect.right                = 3 * rect.right      / 4 ;
                    rect.top                 =     rect.bottom     / 4 ;
                    rect.bottom                 = 3 * rect.bottom     / 4
```

```
        hemf = GetEnhMetaFile (TEXT ("..\\emf3\\emf3.emf")) ;
        EnumEnhMetaFile (     hdc, hemf, EnhMetaFileProc, NULL,
                                  DeleteEnhMetaFile (hemf) ;
                                  EndPaint (hwnd, &ps) ;
                                  return 0 ;

        case   WM_DESTROY:
                    PostQuitMessage (0) ;
                    return 0 ;
   }
        return DefWindowProc (hwnd, message, wParam, lParam)
}
```

EMF5EMF6EMF3EMF3.EMFMetaFileVisual C+

EMF6MetaFilemallocMetaFilepEmfRecordnSizepEmfr
pEmfrENHMETARECORD

CopyMemorypEmfRecordpEmfrEMR_RECTANGLE
EMR_ELLIPSEiTypePEmfrPlayEnhMetaFileRecord

RectangleEllipseMetaFile

EMF6.Cif

```
if (pEmfr->iType == EMR_RECTANGLE)
```

```
{
        PlayEnhMetaFileRecord (hdc, pHandleTable, pEmfr, nObje
        pEmfr->iType = EMR_ELLIPSE ;
}
```

RectangleEllipse

MetaFileGDI

MetaFileENHMETAHEADERnHandlesMetaFileGDIEMF5EMF6
MetaFile3

EMF5EMF6nHandles3

HANDLETABLEWINGDI.H

```
typedef struct tagHANDLETABLE
{
        HGDIOBJ objectHandle [1] ;
}
HANDLETABLE ;
```

HGDIOBJGDI32GDIobjectHandlenHandles3

GDI

```
pHandleTable->objectHandle[i]
```

3i012

MetaFile

0

MetaFileEMR_CREATEBRUSHINDIRECT1PlayEnhMetaFileRecord GDIobjectHandle1EMR_SELECTOBJECT PlayEnhMetaFileRecordGDI1SelectObjectMetaFileGDI objectHandle10

objectHandleGetObjectTypeGetObjectMetaFile

MetaFileMetaFileMetaFileMetaFileMetaFileMetaFileMetaFile EnumEnhMetaFileMetaFileMetaFileGDI

MetaFileEMR_HEADEREMF_EOFMetaFile18-8

18-8 EMF7

```
EMF7.C

/*------------------------------------------------------------------------

  EMF7.C -- Enhanced MetaFile Demo #7

                                        (c) Charles Petzold, 1998

------------------------------------------------------------------------*/

#include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
```

```
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                                PSTR lpszCmdLine,
{
     static TCHAR szAppName[] = TEXT ("EMF7") ;
     HWND                 hwnd ;
     MSG                  msg ;
     WNDCLASS             wndclass ;

  wndclass.style                  = CS_HREDRAW | CS_VR
     wndclass.lpfnWndProc            = WndProc ;
     wndclass.cbClsExtra             = 0 ;
     wndclass.cbWndExtra             = 0 ;
     wndclass.hInstance              = hInstance ;
     wndclass.hIcon                  = LoadIcon (NULL, IDI_
     wndclass.hCursor           = LoadCursor (NULL, IDC_A
     wndclass.hbrBackground          = GetStockObject (WH
     wndclass.lpszMenuName           = NULL ;
     wndclass.lpszClassName          = szAppName ;
```

```
if (!RegisterClass (&wndclass))
{
     MessageBox (  NULL, TEXT ("This program requires W
                                          szAppName, MB_IC
     return 0 ;
}

hwnd = CreateWindow (szAppName, TEXT ("Enhanced Me
           WS_OVERLAPPEDWINDOW,
          CW_USEDEFAULT, CW_USEDEFAULT,
           CW_USEDEFAULT, CW_USEDEFAULT,
           NULL, NULL, hInstance, NULL) ;

ShowWindow (hwnd, iCmdShow) ;
UpdateWindow (hwnd) ;

while (GetMessage (&msg, NULL, 0, 0))
{
```

```
            TranslateMessage (&msg) ;
            DispatchMessage (&msg) ;
        }
    return msg.wParam ;
}

int CALLBACK EnhMetaFileProc (     HDC hdc, HANDLETABLE * pH
                        CONST ENHMETARECORD * pEmfRecord,
                    int iHandles, LPARAM pData)
{
        HBRUSH                  hBrush ;
        HPEN                       hPen ;
        LOGBRUSH                lb ;

        if (pEmfRecord->iType != EMR_HEADER && pEmfRecord->
            PlayEnhMetaFileRecord (hdc, pHandleTable, pEmfReco
        if (pEmfRecord->iType == EMR_RECTANGLE)
        {
            hBrush = SelectObject (hdc, GetStockObject (NULL_B
```

```
                lb.lbStyle = BS_SOLID ;
                lb.lbColor = RGB (0, 255, 0) ;
                lb.lbHatch = 0 ;

                hPen = SelectObject (hdc,
                                        ExtCreatePen (PS_SOLID | PS_GEO
           Ellipse (hdc, 100, 100, 200, 200) ;
                DeleteObject (SelectObject (hdc, hPen)) ;
                SelectObject (hdc, hBrush) ;
          }
          return TRUE ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
          ENHMETAHEADER        emh ;
          HDC                              hdc, hdcEMF ;
          HENHMetaFile         hemfOld, hemf ;
          PAINTSTRUCT                ps ;
```

```
     RECT                                    rect ;

     switch (message)
     {
     case   WM_CREATE:
                                                  // Retrieve existing Meta

               hemfOld = GetEnhMetaFile (TEXT ("..\\emf3\\emf

              GetEnhMetaFileHeader (hemfOld, sizeof (ENHMET

                                                  // Create a new MetaFile

          hdcEMF = CreateEnhMetaFile (NULL, TEXT ("emf7.er
                         TEXT ("EMF7\0EMF Demo #7\0")) ;

                                                  // Enumerate the existin
```

```
                EnumEnhMetaFile (hdcEMF, hemfOld, EnhMetaF
                                (RECT *) & emh.rclBounds) ;

                                        // Clean up

        hemf = CloseEnhMetaFile (hdcEMF) ;

        DeleteEnhMetaFile (hemfOld) ;
             DeleteEnhMetaFile (hemf) ;
             return 0 ;

case   WM_PAINT:
             hdc = BeginPaint (hwnd, &ps) ;

             GetClientRect (hwnd, &rect) ;
             rect.left                  =     rect.right      / 4 ;
             rect.right                  = 3 * rect.right      / 4 ;
             rect.top                   =     rect.bottom   / 4 ;
```

```
                rect.bottom                = 3 * rect.bottom   / 4 ;

                hemf = GetEnhMetaFile (TEXT ("emf7.emf")) ;

                PlayEnhMetaFile (hdc, hemf, &rect) ;
                DeleteEnhMetaFile (hemf) ;
                EndPaint (hwnd, &ps) ;
                return 0 ;

        case   WM_DESTROY:
                PostQuitMessage (0) ;
                return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam)
}
```

EMF7EMF3EMF3.EMFEMF7EMF3MetaFile

EMF7WM_PAINTPlayEnhMetaFileEnumEnhMetaFileWM_CREATE

GetEnhMetaFileEMF3.EMFMetaFileGetEnhMetaFileHeaderMetaFile
EnumEnhMetaFilerclBounds

MetaFileEMF7.EMFCreateEnhMetaFileMetaFileEMF7.EMF
MetaFileEMF3.EMFMetaFileEnumEnhMetaFile

EnhMetaFileProcPlayEnhMetaFileRecordMetaFileMetaFile

RectangleMetaFilePlayEnhMetaFileMetaFile
MetaFile

WM_CREATECloseEnhMetaFileMetaFileMetaFileEMF3.EMF
EMF7.EMF

MetaFile

MetaFileCF_ENHMetaFileGetClipboardDataMetaFile
SetClipboardDataMetaFileMetaFileCopyEnhMetaFileMetaFile
WindowsMetaFileWindowsMetaFile

18-9　　EMFVIEWMetaFileMetaFile

18-9 EMFVIEW

EMFVIEW.C

```
/*------------------------------------------------------------------------
  EMFVIEW.C -- View Enhanced MetaFiles
                                          (c) Charles Petzold, 1998
------------------------------------------------------------------------*/

#include <windows.h>
#include <commdlg.h>
```

```
#include "resource.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

TCHAR szAppName[] = TEXT ("EmfView") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                          PSTR szCmdLine, int iCr
{
        HACCEL                  hAccel ;
        HWND                      hwnd ;
        MSG                      msg ;
   WNDCLASS wndclass ;
        wndclass.style                    = CS_HREDRAW | CS_V
        wndclass.lpfnWndProc                = WndProc ;
        wndclass.cbClsExtra                 = 0 ;
        wndclass.cbWndExtra                 = 0 ;
        wndclass.hInstance                  = hInstance ;
        wndclass.hIcon                = LoadIcon (NULL, IDI_APPL
        wndclass.hCursor               = LoadCursor (NULL, IDC_A
        wndclass.hbrBackground          = (HBRUSH) GetStockO
```

```
wndclass.lpszMenuName  = szAppName ;
wndclass.lpszClassName = szAppName ;

if (!RegisterClass (&wndclass))
{
     MessageBox (  NULL, TEXT ("This program requires W
                                        szAppName, MB_IC
     return 0 ;
}

hwnd = CreateWindow (szAppName, TEXT ("Enhanced Me
                    WS_OVERLAPPEDWINDOW,
                    CW_USEDEFAULT, CW_USEDEFAULT,
                   CW_USEDEFAULT, CW_USEDEFAULT,
                    NULL, NULL, hInstance, NULL) ;

ShowWindow (hwnd, iCmdShow) ;
UpdateWindow (hwnd) ;

hAccel = LoadAccelerators (hInstance, szAppName) ;
while (GetMessage (&msg, NULL, 0, 0))
```

```
        {
             if (!TranslateAccelerator (hwnd, hAccel, &msg))
             {
                        TranslateMessage (&msg) ;
                      DispatchMessage (&msg) ;
             }
        }
        return msg.wParam ;
}

HPALETTE CreatePaletteFromMetaFile (HENHMetaFile hemf)
{
        HPALETTE                hPalette ;
        int                     iNum ;
        LOGPALETTE *  plp ;

        if (!hemf)
                return NULL ;
        if (0 == (iNum = GetEnhMetaFilePaletteEntries (hemf, 0,
```

```
                    return NULL ;

          plp = malloc (sizeof (LOGPALETTE) + (iNum - 1) * sizeof (

          plp->palVersion    = 0x0300 ;

          plp->palNumEntries = iNum ;


          GetEnhMetaFilePaletteEntries (hemf, iNum, plp->palPalEr

          hPalette = CreatePalette (plp) ;

          free (plp) ;

          return hPalette ;

}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA

{

          static DOCINFO       di = { sizeof (DOCINFO), TEXT ("EmfV

          static HENHMetaFile hemf ;

          static OPENFILENAME ofn ;

          static PRINTDLG             printdlg = { sizeof (PRINTDLG) }

          static TCHAR                    szFileName [MAX_PATH], szT

          static TCHAR                    szFilter[] =

                                                              TEXT ("Enha
```

```
                                                    TEXT ("All F

     BOOL                                    bSuccess ;
     ENHMETAHEADER                  header ;
     HDC                                     hdc, hdcPrn ;
     HENHMetaFile                hemfCopy ;
     HMENU                                   hMenu ;
     HPALETTE                               hPalette ;
     int                                     i, iLength, iEnable
     PAINTSTRUCT                          ps ;
     RECT                                    rect ;
     PTSTR                                   pBuffer ;

     switch (message)
     {
     case WM_CREATE:
                    // Initialize OPENFILENAME structure
               ofn.lStructSize             = sizeof (OPENFILEN
               ofn.hwndOwner               = hwnd ;
               ofn.hInstance               = NULL ;
```

```
    ofn.lpstrFilter       = szFilter ;
    ofn.lpstrCustomFilter = NULL ;
    ofn.nMaxCustFilter    = 0 ;
    ofn.nFilterIndex      = 0 ;
    ofn.lpstrFile         = szFileName ;
    ofn.nMaxFile          = MAX_PATH ;
    ofn.lpstrFileTitle    = szTitleName ;
    ofn.nMaxFileTitle     = MAX_PATH ;
    ofn.lpstrInitialDir   = NULL ;
    ofn.lpstrTitle        = NULL ;
    ofn.Flags             = 0 ;
    ofn.nFileOffset       = 0 ;
    ofn.nFileExtension    = 0 ;
    ofn.lpstrDefExt       = TEXT ("emf") ;
    ofn.lCustData         = 0 ;
    ofn.lpfnHook          = NULL ;
    ofn.lpTemplateName    = NULL ;
    return 0 ;
```

```
case   WM_INITMENUPOPUP:
                hMenu = GetMenu (hwnd) ;

               iEnable = hemf ? MF_ENABLED : MF_GRAYED ;

     EnableMenuItem (hMenu, IDM_FILE_SAVE_AS,          iEnab
      EnableMenuItem (hMenu, IDM_FILE_PRINT,           iEnab
      EnableMenuItem (hMenu, IDM_FILE_PROPERTIES,        iE
      EnableMenuItem (hMenu, IDM_EDIT_CUT,             iEnab
           EnableMenuItem (hMenu, IDM_EDIT_COPY,
           EnableMenuItem (hMenu, IDM_EDIT_DELETE,
           EnableMenuItem (hMenu, IDM_EDIT_PASTE,
            IsClipboardFormatAvailable (CF_ENHMetaFile) ?
            MF_ENABLED : MF_GRAYED) ;
           return 0 ;

      case   WM_COMMAND:
           switch (LOWORD (wParam))
                     {
                case   IDM_FILE_OPEN:
```

```
                // Show the File Open dialog box

                ofn.Flags = 0 ;


                if (!GetOpenFileName (&ofn))
                        return 0 ;


        // If there's an existing EMF, get rid of it.

                if (hemf)
                {
                        DeleteEnhMetaFile (hem
                       hemf = NULL ;
                }
                        // Load the EMF into me

            SetCursor (LoadCursor (NULL, IDC_WA
            ShowCursor (TRUE) ;

            hemf = GetEnhMetaFile (szFileName) ;
```

```
                    ShowCursor (FALSE) ;
                    SetCursor (LoadCursor (NULL, IDC_ARR

          // Invalidate the client area for later update

                    InvalidateRect (hwnd, NULL, TRUE) ;

                   if (hemf == NULL)
                    {
          MessageBox (   hwnd, TEXT ("Cannot load MetaFile")
                        szAppName, MB_ICONEXCLAMATION | MB_
                    }
                    return 0 ;

          case   IDM_FILE_SAVE_AS:
                    if (!hemf)
                              return 0 ;

                              // Show the File Save dialog

                    ofn.Flags = OFN_OVERWRITEPROMPT ;
```

```
                    if (!GetSaveFileName (&ofn))
                                return 0 ;

                                // Save the EMF to disk file

                    SetCursor (LoadCursor (NULL, IDC_WA
                    ShowCursor (TRUE) ;

                    hemfCopy = CopyEnhMetaFile (hemf, 

                    ShowCursor (FALSE) ;
                    SetCursor (LoadCursor (NULL, IDC_ARR
                    if (hemfCopy)
                    {
                                DeleteEnhMetaFile (hemf) ;
                               hemf = hemfCopy ;
                    }
                    else
          MessageBox (   hwnd, TEXT ("Cannot save MetaFile"
```

```
                    szAppName, MB_ICONEXCLAMATION | MB_(
                    return 0 ;

          case   IDM_FILE_PRINT:
                // Show the Print dialog box and get printer DC

          printdlg.Flags = PD_RETURNDC | PD_NOPAGENUMS |

                    if (!PrintDlg (&printdlg))
                              return 0 ;

                    if (NULL == (hdcPrn = printdlg.hDC))
                    {
        MessageBox (    hwnd, TEXT ("Cannot obtain printer D
                  szAppName, MB_ICONEXCLAMATION | MB_OK
                              return 0 ;
                    }
                              // Get size of printable area (

                    rect.left            = 0 ;
                    rect.right           = GetDeviceCaps (hd
```

```
                    rect.top            = 0 ;
                   rect.bottom          = GetDeviceCaps (h
                    bSuccess = FALSE ;
                              // Play the EMF to the printer
                    SetCursor (LoadCursor (NULL, IDC_WA
                    ShowCursor (TRUE) ;
                    if ((StartDoc (hdcPrn, &di) > 0) && (St
          {
                               PlayEnhMetaFile (hdcPrn, he
                          if (EndPage (hdcPrn) > 0)
                          {
                                         bSuccess = TRUE ;
                                         EndDoc (hdcPrn) ;
                          }
                   }
```

```
                    ShowCursor (FALSE) ;
                    SetCursor (LoadCursor (NULL, IDC_ARR

                   DeleteDC (hdcPrn) ;

                    if (!bSuccess)
MessageBox (       hwnd, TEXT ("Could not print MetaFil
                szAppName, MB_ICONEXCLAMATION | MB_O
                    return 0 ;

  case   IDM_FILE_PROPERTIES:
                    if (!hemf)
                                return 0 ;

                    iLength = GetEnhMetaFileDescription (
                    pBuffer = malloc ((iLength + 256) * siz

                   GetEnhMetaFileHeader (hemf, sizeof (E

                         // Format header file information

                    i  = wsprintf (pBuffer,
```

```
                                        TEXT ("Bound
                                    header.rclBound
                                    header.rclBound

          i += wsprintf (pBuffer + i,
                                        TEXT ("Frame
                                    header.rclFrame
                                    header.rclFrame

        i += wsprintf (pBuffer + i,
                                        TEXT ("Resolu
                                        TEXT (" = (%i
                                   header.szlDevice
                                    header.szlMillim
                                    header.szlMillim

          i += wsprintf (pBuffer + i,
                                      TEXT ("Size =
                                       TEXT ("Handle
                                   header.nBytes, h
```

```
                                                header.nHandles

                                        // Include the MetaFile d

                    if (iLength)

                    {

                                        i += wsprintf (pBuffer +

                                    GetEnhMetaFileDescriptio

                                        pBuffer [lstrlen (pBuffer

                    }

                    MessageBox (hwnd, pBuffer, TEXT ("M

                    free (pBuffer) ;

                   return 0 ;

          case   IDM_EDIT_COPY:

          case   IDM_EDIT_CUT:

                    if (!hemf)

                                return 0 ;

                                // Transfer MetaFile copy to t
```

```
                    hemfCopy = CopyEnhMetaFile (hemf, 

                    OpenClipboard (hwnd) ;
                    EmptyClipboard () ;
                    SetClipboardData (CF_ENHMetaFile, he
                    CloseClipboard () ;

                    if (LOWORD (wParam) == IDM_EDIT_C
                              return 0 ;
                         // fall through if IDM_EDIT_CUT
          case   IDM_EDIT_DELETE:
                    if (hemf)
                    {
                         DeleteEnhMetaFile (hemf) ;
                        hemf = NULL ;
                         InvalidateRect (hwnd, NULL, TRU
                    }
                    return 0 ;

     case   IDM_EDIT_PASTE:
```

```
                    OpenClipboard (hwnd) ;
                    hemfCopy = GetClipboardData (CF_EN

                    CloseClipboard () ;
                    if (hemfCopy && hemf)
                    {
                         DeleteEnhMetaFile (hemf) ;
                        hemf = NULL ;
                    }

                    hemf = CopyEnhMetaFile (hemfCopy,
                    InvalidateRect (hwnd, NULL, TRUE) ;
                    return 0 ;

     case   IDM_APP_ABOUT:
                    MessageBox (  hwnd, TEXT ("Enhance
          TEXT ("(c) Charles Petzold, 1998"),
                                                szAppName

                    return 0 ;

     case   IDM_APP_EXIT:
```

```
                    SendMessage (hwnd, WM_CLOSE, 0, 0
                    return 0 ;
          }
          break ;

     case   WM_PAINT:
               hdc = BeginPaint (hwnd, &ps) ;

               if (hemf)
               {
                    if (   hPalette = CreatePaletteFromMeta
                    {
                         SelectPalette (hdc, hPalette,
                         RealizePalette (hdc) ;
                    }
                    GetClientRect (hwnd, &rect) ;
                    PlayEnhMetaFile (hdc, hemf, &rect) ;

                    if (hPalette)
```

```
                                        DeleteObject (hPalette) ;
                }
                EndPaint (hwnd, &ps) ;
                return 0 ;

        case   WM_QUERYNEWPALETTE:
                if (!hemf || !(hPalette = CreatePaletteFromMetaF
                                        return FALSE ;

                hdc = GetDC (hwnd) ;
                SelectPalette (hdc, hPalette, FALSE) ;
               RealizePalette (hdc) ;
                InvalidateRect (hwnd, NULL, FALSE) ;

                DeleteObject (hPalette) ;
                ReleaseDC (hwnd, hdc) ;
                return TRUE ;

        case   WM_PALETTECHANGED:
                if ((HWND) wParam == hwnd)
```

```
                    break ;

                if (!hemf || !(hPalette = CreatePaletteFromMetaF
                    break ;

                hdc = GetDC (hwnd) ;
                SelectPalette (hdc, hPalette, FALSE) ;
                RealizePalette (hdc) ;
                UpdateColors (hdc) ;

                DeleteObject (hPalette) ;
                ReleaseDC (hwnd, hdc) ;
                break ;

        case   WM_DESTROY:
                if (hemf)
                    DeleteEnhMetaFile (hemf) ;

                PostQuitMessage (0) ;
                return 0 ;
```

```
        }
    return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

EMFVIEW.RC

```
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
//////////////////////////////////////////////////////////////////////////////
// Menu
EMFVIEW MENU DISCARDABLE
BEGIN
   POPUP "&File"
   BEGIN
      MENUITEM "&Open\tCtrl+O",  IDM_FILE_OPEN
      MENUITEM "Save &As...",    IDM_FILE_SAVE_AS
          MENUITEM SEPARATOR
          MENUITEM "&Print...\tCtrl+P",IDM_FILE_PRINT
      MENUITEM SEPARATOR
```

```
        MENUITEM "&Properties",       IDM_FILE_PROPERTIES
        MENUITEM SEPARATOR
        MENUITEM "E&xit",             IDM_APP_EXIT
END
  POPUP "&Edit"
  BEGIN
    MENUITEM "Cu&t\tCtrl+X",                    IDM_EDIT_CUT
    MENUITEM "&Copy\tCtrl+C",                IDM_EDIT_COPY
        MENUITEM "&Paste\tCtrl+V",          IDM_EDIT_PASTE
    MENUITEM "&Delete\tDel",                    IDM_EDIT_DELETE
  END
  POPUP "Help"
  BEGIN
    MENUITEM "&About EmfView...",          IDM_APP_ABOUT
  END
END

//////////////////////////////////////////////////////////////////////////
// Accelerator
```

```
EMFVIEW ACCELERATORS DISCARDABLE
BEGIN
"C",IDM_EDIT_COPY, VIRTKEY, CONTROL, NOINVERT
"O",IDM_FILE_OPEN, VIRTKEY, CONTROL, NOINVERT
"P",IDM_FILE_PRINT,VIRTKEY, CONTROL, NOINVERT
"V",IDM_EDIT_PASTE,VIRTKEY, CONTROL, NOINVERT
VK_DELETE,IDM_EDIT_DELETE,VIRTKEY, NOINVERT
"X",IDM_EDIT_CUT,VIRTKEY, CONTROL, NOINVERT
END
```

```
RESOURCE.H
// Microsoft Developer Studio generated include file.
// Used by EmfView.rc
#define          IDM_FILE_OPEN           40001
#define          IDM_FILE_SAVE_AS        40002
#define          IDM_FILE_PRINT          40003
#define          IDM_FILE_PROPERTIES     40004
#define          IDM_APP_EXIT            40005
#define          IDM_EDIT_CUT            40006
```

```
#define         IDM_EDIT_COPY           40007
#define         IDM_EDIT_PASTE          40008
#define         IDM_EDIT_DELETE         40009
#define         IDM_APP_ABOUT           40010
```

EMFVIEWMetaFileSelectpaletteCreatePaletteFromMetaFile
WM_PAINTMetaFileWM_QUERYNEWPALETTEWM_PALETTECHANGED

PrintEMFVIEWMetaFileEMFVIEWMetaFile

FilePropertiesEMFVIEWMetaFile

EMF2.EMFMetaFile

MetaFile

MetaFileMetaFile

MetaFileMetaFileMetaFileMetaFile

MetaFileMetaFileMetaFile

MetaFileMetaFileMetaFile

PlayEnhMetaFileMetaFile

MetaFileMetaFile

EMF.CEMF.RCRESOURCE.H18-10EMF8.C6

18-10 EMF8

```
EMF8.C
```

```
/*------------------------------------------------------------------------
  EMF8.C -- Enhanced MetaFile Demo #8
                                        (c) Charles Petzold, 1998
------------------------------------------------------------------------*/
#include <windows.h>
TCHAR szClass             [] = TEXT ("EMF8") ;
TCHAR szTitle             [] = TEXT ("EMF8: Enhanced MetaFile Den

void DrawRuler (HDC hdc, int cx, int cy)
{
          int                    iAdj, i, iHeight ;
          LOGFONT             lf ;
          TCHAR                    ch ;

          iAdj = GetVersion () & 0x80000000 ? 0 : 1 ;
                              // Black pen with 1-point width
          SelectObject (hdc, CreatePen (PS_SOLID, cx / 72 / 6, 0)) ;
                              // Rectangle surrounding entire pen (with a
          Rectangle (hdc, iAdj, iAdj, cx + iAdj + 1, cy + iAdj + 1) ;
```

```
                    // Tick marks

for (i = 1 ; i < 96 ; i++)

{

              if (i % 16 == 0) iHeight = cy /  2 ;        // inches

 else   if (i %  8 == 0) iHeight = cy /  3 ;       // half inches

 else   if (i %  4 == 0) iHeight = cy /  5 ;        // quarter in

 else   if (i %  2 == 0) iHeight = cy /  8 ;        // eighths

else                iHeight = cy / 12 ;           // sixteenths

      MoveToEx (hdc, i * cx / 96, cy, NULL) ;

     LineTo   (hdc, i * cx / 96, cy - iHeight) ;

}

                    // Create logical font

FillMemory (&lf, sizeof (lf), 0) ;

lf.lfHeight = cy / 2 ;

lstrcpy (lf.lfFaceName, TEXT ("Times New Roman")) ;

SelectObject (hdc, CreateFontIndirect (&lf)) ;

SetTextAlign (hdc, TA_BOTTOM | TA_CENTER) ;

SetBkMode           (hdc, TRANSPARENT) ;
```

```
                    // Display numbers

        for (i = 1 ; i <= 5 ; i++)
        {
                ch = (TCHAR) (i + '0') ;
                TextOut (hdc, i * cx / 6, cy / 2, &ch, 1) ;
        }
                    // Clean up
        DeleteObject (SelectObject (hdc, GetStockObject (SYSTEM
        DeleteObject (SelectObject (hdc, GetStockObject (BLACK_
}

void CreateRoutine (HWND hwnd)
{
        HDC                         hdcEMF ;
        HENHMetaFile hemf ;
        int                         cxMms, cyMms, cxPix, cyPix, xDp

        hdcEMF = CreateEnhMetaFile (NULL, TEXT ("emf8.emf"),
```

```
                        TEXT ("EMF8\0EMF Demo #8\0")) ;

        if (hdcEMF == NULL)

                return ;

        cxMms       = GetDeviceCaps (hdcEMF, HORZSIZE) ;

        cyMms       = GetDeviceCaps (hdcEMF, VERTSIZE) ;

        cxPix       = GetDeviceCaps (hdcEMF, HORZRES) ;

        cyPix       = GetDeviceCaps (hdcEMF, VERTRES) ;


        xDpi        = cxPix * 254 / cxMms / 10 ;

        yDpi        = cyPix * 254 / cyMms / 10 ;


        DrawRuler (hdcEMF, 6 * xDpi, yDpi) ;

        hemf = CloseEnhMetaFile (hdcEMF) ;

        DeleteEnhMetaFile (hemf) ;

}


void PaintRoutine (HWND hwnd, HDC hdc, int cxArea, int cyArea)

{

        ENHMETAHEADER       emh ;

        HENHMetaFile        hemf ;
```

```
          int                                   cxImage, cyImage ;
          RECT                                  rect ;

          hemf = GetEnhMetaFile (TEXT ("emf8.emf")) ;
          GetEnhMetaFileHeader (hemf, sizeof (emh), &emh) ;
          cxImage = emh.rclBounds.right - emh.rclBounds.left ;
          cyImage = emh.rclBounds.bottom - emh.rclBounds.top ;

          rect.left           = (cxArea - cxImage) / 2 ;
          rect.right          = (cxArea + cxImage) / 2 ;
          rect.top            = (cyArea - cyImage) / 2 ;
          rect.bottom         = (cyArea + cyImage) / 2 ;

          PlayEnhMetaFile (hdc, hemf, &rect) ;
          DeleteEnhMetaFile (hemf) ;
}
```

EMF.C

```
/*------------------------------------------------------------------------
  EMF.C --      Enhanced MetaFile Demonstration Shell Program
                              (c) Charles Petzold, 1998
```

```
------------------------------------------------------------------------*/
#include <windows.h>
#include <commdlg.h>
#include "..\\emf8\\resource.h"

extern void CreateRoutine (HWND) ;
extern void PaintRoutine  (HWND, HDC, int, int) ;

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
HANDLE hInst ;
extern TCHAR szClass [] ;
extern TCHAR szTitle [] ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                                PSTR szCmdLine, int iCr
{
          TCHAR                    szResource [] = TEXT ("EMF")
          HWND                      hwnd ;
          MSG                      msg ;
          WNDCLASS                 wndclass ;
```

```
     hInst = hInstance ;
     wndclass.style                    = CS_HREDRAW | CS_V
     wndclass.lpfnWndProc                = WndProc ;
     wndclass.cbClsExtra                 = 0 ;
     wndclass.cbWndExtra                 = 0 ;
     wndclass.hInstance                  = hInstance ;
     wndclass.hIcon                    = LoadIcon (NULL, IDI_
     wndclass.hCursor                    = LoadCursor (NULL,
     wndclass.hbrBackground            = GetStockObject (WH
     wndclass.lpszMenuName             = szResource ;
     wndclass.lpszClassName            = szClass ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (  NULL, TEXT ("This program requires W
                                          szClass, MB_ICONE
          return 0 ;
     }
```

```
        hwnd = CreateWindow (szClass, szTitle,
                    WS_OVERLAPPEDWINDOW,
                    CW_USEDEFAULT, CW_USEDEFAULT,
                    CW_USEDEFAULT, CW_USEDEFAULT,
                   NULL, NULL, hInstance, NULL) ;

        ShowWindow (hwnd, iCmdShow) ;
        UpdateWindow (hwnd) ;

        while (GetMessage (&msg, NULL, 0, 0))
        {
             TranslateMessage (&msg) ;
             DispatchMessage (&msg) ;
        }
        return msg.wParam ;
}

BOOL PrintRoutine (HWND hwnd)
{
```

```
     static DOCINFO              di ;
     static PRINTDLG             printdlg = { sizeof (PRINTDLG) }
     static TCHAR                    szMessage [32] ;
     BOOL                                    bSuccess = FALS
     HDC                                     hdcPrn ;
     int                                    cxPage, cyPage ;

     printdlg.Flags = PD_RETURNDC | PD_NOPAGENUMS | PD_
     if (!PrintDlg (&printdlg))
               return TRUE ;
     if (NULL == (hdcPrn = printdlg.hDC))
               return FALSE ;
   cxPage = GetDeviceCaps (hdcPrn, HORZRES) ;
     cyPage = GetDeviceCaps (hdcPrn, VERTRES) ;

     lstrcpy (szMessage, szClass) ;
     lstrcat (szMessage, TEXT (": Printing")) ;

     di.cbSize                  = sizeof (DOCINFO) ;
     di.lpszDocName       = szMessage ;
```

```
        if (StartDoc (hdcPrn, &di) > 0)
   {
            if (StartPage (hdcPrn) > 0)
            {
                        PaintRoutine (hwnd, hdcPrn, cxPage, cyPag
                        if (EndPage (hdcPrn) > 0)
                        {
                                    EndDoc (hdcPrn) ;
                                     bSuccess = TRUE ;
                        }
            }
        }
        DeleteDC (hdcPrn) ;
        return bSuccess ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
```

```
        BOOL                    bSuccess ;
        static int      cxClient, cyClient ;
        HDC                     hdc ;
        PAINTSTRUCT     ps ;

        switch (message)
        {
        case   WM_CREATE:
                CreateRoutine (hwnd) ;
                return 0 ;

        case   WM_COMMAND:
                switch (wParam)
                {
                case   IDM_PRINT:
                                SetCursor (LoadCursor (NULL, IDC
                                 ShowCursor (TRUE) ;

                                 bSuccess = PrintRoutine (hwnd) ;
```

```
                    ShowCursor (FALSE) ;
                    SetCursor (LoadCursor (NULL, IDC

                    if (!bSuccess)
MessageBox (hwnd, TEXT ("Error encountered during print
          szClass, MB_ICONASTERISK | MB_OK) ;
               return 0 ;

          case   IDM_EXIT:
                    SendMessage (hwnd, WM_CLOSE,
                    return 0 ;

          case   IDM_ABOUT:
MessageBox (hwnd, TEXT ("Enhanced MetaFile Demo Prog
          TEXT ("Copyright (c) Charles Petzold, 1998"),
          szClass, MB_ICONINFORMATION | MB_OK) ;
          return 0 ;
```

```
                }
                break ;

        case   WM_SIZE:
                cxClient = LOWORD (lParam) ;
                cyClient = HIWORD (lParam) ;
                return 0 ;

case   WM_PAINT:
                hdc = BeginPaint (hwnd, &ps) ;

                PaintRoutine (hwnd, hdc, cxClient, cyClient) ;

                EndPaint (hwnd, &ps) ;
                return 0 ;

        case   WM_DESTROY :
                PostQuitMessage (0) ;
                return 0 ;
```

```
        }
        return DefWindowProc (hwnd, message, wParam, lParam)
}
```

EMF.RC

```
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
//////////////////////////////////////////////////////////////////////////////
// Menu
EMF MENU DISCARDABLE
BEGIN
  POPUP "&File"
  BEGIN
                    MENUITEM "&Print...",                       IDM_P
                  MENUITEM SEPARATOR
                    MENUITEM "E&xit",                            IDM_
  END
  POPUP "&Help"
```

```
  BEGIN
                    MENUITEM "&About...",                   IDM
  END
END
```

```
RESOURCE.H
// Microsoft Developer Studio generated include file.
// Used by Emf.rc
//
#define IDM_PRINT              40001
#define IDM_EXIT               40002
#define IDM_ABOUT              40003
```

WM_CREATEEMF.CCreateRoutineMetaFileEMF.CPaintRoutine
WM_PAINTPrintRoutine

MetaFileEMF8MetaFile6115TrueType

6EMF8.CCreateRoutineMetaFileCreateEnhMetaFileGetDeviceCaps

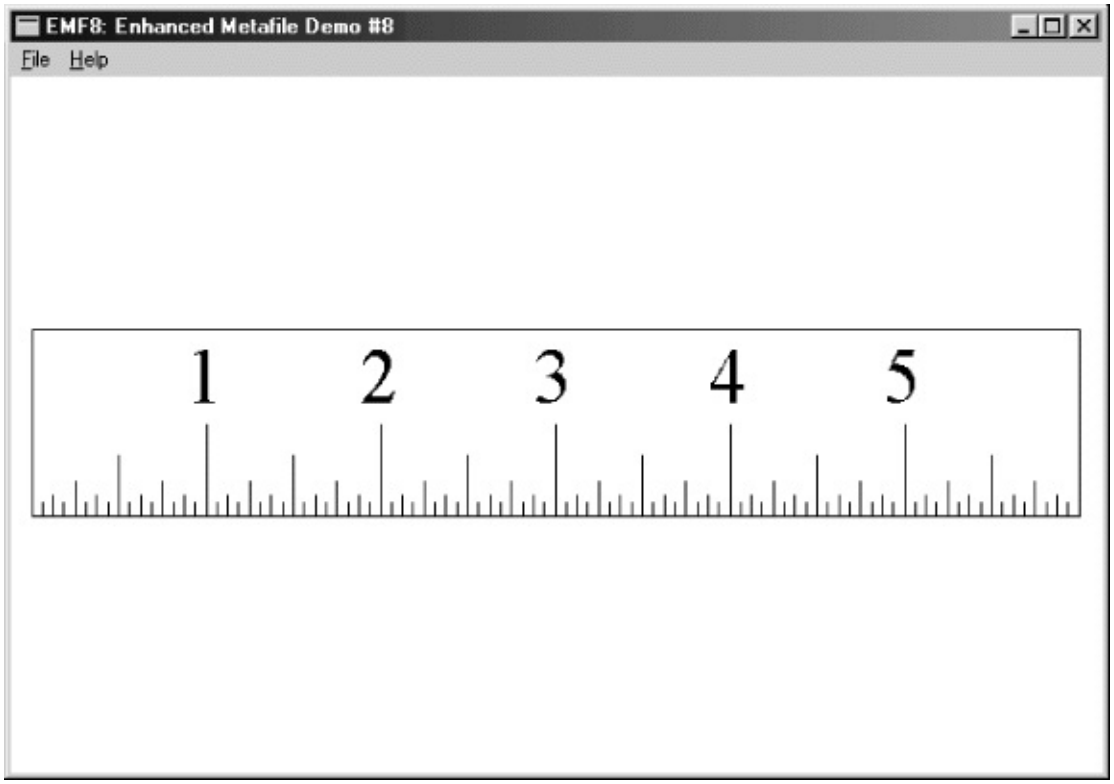
MetaFileGDI

CreateEnhMetaFileGDIMetaFileNULLEMF8GDI
EMF8GetDeviceCaps

EMF8.C25.41… 5.4…

MetaFilePlayEnhMetaFile

EMF8PaintRoutineGetEnhMetaFileHeaderMetaFile
ENHMETAHEADERrclBoundsMetaFile18-6



18-6 EMF8

300dpi11/3

ENHMETAHEADERrclBoundsEMF8rclFrame0.01MetaFile
MetaFileszlDeviceszlMillimetersSIZELGetDeviceCaps

18-11 EMF9

```
EMF9.C

/*------------------------------------------------------------------------
  EMF9.C -- Enhanced MetaFile Demo #9
                                        (c) Charles Petzold, 1998
------------------------------------------------------------------------*/

#include <windows.h>
#include <string.h>

TCHAR szClass              [] = TEXT ("EMF9") ;
TCHAR szTitle              [] = TEXT ("EMF9: Enhanced MetaFile Dem

void CreateRoutine (HWND hwnd)
{
}

void PaintRoutine (HWND hwnd, HDC hdc, int cxArea, int cyArea
{
          ENHMETAHEADER        emh ;
```

```
HENHMetaFile        hemf ;
int                                 cxMms, cyMms, cxPix, cyPix
RECT                                  rect ;

cxMms       = GetDeviceCaps (hdc, HORZSIZE) ;
cyMms       = GetDeviceCaps (hdc, VERTSIZE) ;
cxPix       = GetDeviceCaps (hdc, HORZRES) ;
cyPix       = GetDeviceCaps (hdc, VERTRES) ;

hemf = GetEnhMetaFile (TEXT ("..\\emf8\\emf8.emf")) ;
GetEnhMetaFileHeader (hemf, sizeof (emh), &emh) ;
cxImage                  = emh.rclFrame.right - emh.rclFra
cyImage                  = emh.rclFrame.bottom - emh.rclF

cxImage                  = cxImage * cxPix / cxMms / 100 ;
cyImage                  = cyImage * cyPix / cyMms / 100 ;

rect.left           = (cxArea - cxImage) / 2 ;
rect.right           = (cxArea + cxImage) / 2 ;
rect.top             = (cyArea - cyImage) / 2 ;
```

```
        rect.bottom         = (cyArea + cyImage) / 2 ;

        PlayEnhMetaFile (hdc, hemf, &rect) ;

        DeleteEnhMetaFile (hemf) ;
}
```

EMF9EMF8MetaFileEMF8

EMF9PaintRoutineGetDeviceCapsEMF8CreateRoutineMetaFile
rclFrame0.01MetaFile

1000.01PaintRoutine

EMF9EMF8EMF96…

EMF8MetaFile661MetaFile
PlayEnhMetaFile

18-12 EMF10

18-12 EMF10

```
EMF10.C

/*-------------------------------------------------------------------------

  EMF10.C --    Enhanced MetaFile Demo #10

                                    (c) Charles Petzold, 1998

-------------------------------------------------------------------------*/
```

```
#include <windows.h>
TCHAR szClass             [] = TEXT ("EMF10") ;
TCHAR szTitle             [] = TEXT ("EMF10: Enhanced MetaFile De
void CreateRoutine (HWND hwnd)
{
}

void PaintRoutine (HWND hwnd, HDC hdc, int cxArea, int cyArea
{
          ENHMETAHEADER        emh ;
          float                              fScale ;
          HENHMetaFile         hemf ;
          int                                cxMms, cyMms, cxPix, cyPix
          RECT                               rect ;

          cxMms        = GetDeviceCaps (hdc, HORZSIZE) ;
          cyMms        = GetDeviceCaps (hdc, VERTSIZE) ;
          cxPix        = GetDeviceCaps (hdc, HORZRES) ;
          cyPix        = GetDeviceCaps (hdc, VERTRES) ;
```

```
hemf   = GetEnhMetaFile (TEXT ("..\\emf8\\emf8.emf")) ;

GetEnhMetaFileHeader (hemf, sizeof (emh), &emh) ;

cxImage      = emh.rclFrame.right - emh.rclFrame.left ;
cyImage      = emh.rclFrame.bottom - emh.rclFrame.top ;

cxImage      = cxImage * cxPix / cxMms / 100 ;
cyImage      = cyImage * cyPix / cyMms / 100 ;

fScale = min ((float) cxArea / cxImage, (float) cyArea / cy

cxImage                  = (int) (fScale * cxImage) ;
cyImage                  = (int) (fScale * cyImage) ;

rect.left           = (cxArea - cxImage) / 2 ;
rect.right           = (cxArea + cxImage) / 2 ;
rect.top                  = (cyArea - cyImage) / 2 ;
rect.bottom = (cyArea + cyImage) / 2 ;
PlayEnhMetaFile (hdc, hemf, &rect) ;
DeleteEnhMetaFile (hemf) ;
```

```
}
```

EMF10

EMF9EMF10.CPaintRoutineEMF9.C6…

fScale

MetaFile

… DI

MetaFile

MetaFileSetMapModeMetaFileGDI18-13

18-13 EMF11

```
EMF11.C

/*--------------------------------------------------------------------

  EMF11.C -- Enhanced MetaFile Demo #11

                                          (c) Charles Petzold, 1998

----------------------------------------------------------------------*/

#include <windows.h>

TCHAR szClass              [] = TEXT ("EMF11") ;

TCHAR szTitle              [] = TEXT ("EMF11: Enhanced MetaFile De

void DrawRuler (HDC hdc, int cx, int cy)
```

```
{
          int                    i, iHeight ;
          LOGFONT              lf ;
          TCHAR                     ch ;

                    // Black pen with 1-point width

          SelectObject (hdc, CreatePen (PS_SOLID, cx / 72 / 6, 0)) ;
                    // Rectangle surrounding entire pen (with adjustr
          if (GetVersion () & 0x80000000)      `                    // Windo
               Rectangle (hdc, 0, -2, cx + 2, cy) ;
          else
// Windows NT
                    Rectangle (hdc, 0, -1, cx + 1, cy) ;

                    // Tick marks


         for (i = 1 ; i < 96 ; i++)
          {
       if(i %16== 0)   iHeight = cy /2      ;      // inches
```

```
else if(i % 8     == 0)iHeight = cy /   3 ;    // half inches
 else if(i % 4     == 0)iHeight = cy /   5 ;    // quarter inches
 else if(i % 2     == 0)iHeight = cy /   8 ;    // eighths
 else iHeight = cy /12     ;      // sixteenths

    MoveToEx (hdc, i * cx / 96, 0, NULL) ;
         LineTo   (hdc, i * cx / 96, iHeight) ;
   }
         // Create logical font
   FillMemory (&lf, sizeof (lf), 0) ;
   lf.lfHeight = cy / 2 ;
   lstrcpy (lf.lfFaceName, TEXT ("Times New Roman")) ;

   SelectObject (hdc, CreateFontIndirect (&lf)) ;
   SetTextAlign (hdc, TA_BOTTOM | TA_CENTER) ;
   SetBkMode            (hdc, TRANSPARENT) ;

         // Display numbers

   for (i = 1 ; i <= 5 ; i++)
```

```
        {
             ch = (TCHAR) (i + '0') ;
             TextOut (hdc, i * cx / 6, cy / 2, &ch, 1) ;
        }
             // Clean up
        DeleteObject (SelectObject (hdc, GetStockObject (SYSTEM
    DeleteObject (SelectObject (hdc, GetStockObject (BLACK_PEN
}

void CreateRoutine (HWND hwnd)
{
        HDC         hdcEMF ;
         HENHMetaFile hemf ;

   hdcEMF = CreateEnhMetaFile (NULL, TEXT ("emf11.emf"), NUL
                          TEXT ("EMF11\0EMF Demo #11\0")) ;
         SetMapMode (hdcEMF, MM_LOENGLISH) ;
         DrawRuler (hdcEMF, 600, 100) ;
         hemf = CloseEnhMetaFile (hdcEMF) ;
```

```
        DeleteEnhMetaFile (hemf) ;
}

void PaintRoutine (HWND hwnd, HDC hdc, int cxArea, int cyArea
{
        ENHMETAHEADER        emh ;
        HENHMetaFile         hemf ;
        int                  cxMms, cyMms, cxPix, cyPix, cxImage, cyI
        RECT                 rect ;

        cxMms        = GetDeviceCaps (hdc, HORZSIZE) ;
        cyMms        = GetDeviceCaps (hdc, VERTSIZE) ;
        cxPix        = GetDeviceCaps (hdc, HORZRES) ;
        cyPix        = GetDeviceCaps (hdc, VERTRES) ;

        hemf = GetEnhMetaFile (TEXT ("emf11.emf")) ;
        GetEnhMetaFileHeader (hemf, sizeof (emh), &emh) ;
        cxImage      = emh.rclFrame.right - emh.rclFrame.left ;
        cyImage      = emh.rclFrame.bottom - emh.rclFrame.top ;
```

```
        cxImage      = cxImage * cxPix / cxMms / 100 ;
        cyImage      = cyImage * cyPix / cyMms / 100 ;

        rect.left           = (cxArea - cxImage) / 2 ;
        rect.top            = (cyArea - cyImage) / 2 ;
        rect.right          = (cxArea + cxImage) / 2 ;
        rect.bottom         = (cyArea + cyImage) / 2 ;

        PlayEnhMetaFile (hdc, hemf, &rect) ;
        DeleteEnhMetaFile (hemf) ;
}
```

EMF11CreateRoutineEMF8MetaFileGetDeviceCaps…
SetMapModeMM_LOENGLISH0.01…

MoveToExLineToEMF11DrawRulerEMF9MM_TEXT
MM_LOENGLISHRectangle

EMF11PaintRoutineEMF9EMF11EMF11.EMFEMF9EMF8
EMF8.EMF

EMF11EMF9SetMapModeMetaFileMetaFileMetaFile

EMF11GetDeviceCapsGDIMM_HIMETRIC0.01MetaFile

18-14EMF12EMF8DrawRulerMM_HIMETRICMetaFile

18-14　EMF12

EMF12.C

```
/*--------------------------------------------------------------------------
  EMF12.C --    Enhanced MetaFile Demo #12
                                        (c) Charles Petzold, 1998
--------------------------------------------------------------------------*/
#include <windows.h>
TCHAR szClass             [] = TEXT ("EMF12") ;
TCHAR szTitle             [] = TEXT ("EMF12: Enhanced MetaFile De
void DrawRuler (HDC hdc, int cx, int cy)
{
          int                    iAdj, i, iHeight ;
          LOGFONT              lf ;
          TCHAR                     ch ;

          iAdj = GetVersion () & 0x80000000 ? 0 : 1 ;
```

```
                // Black pen with 1-point width

    SelectObject (hdc, CreatePen (PS_SOLID, cx / 72 / 6, 0)) ;

                // Rectangle surrounding entire pen (with adjust

    Rectangle (hdc, iAdj, iAdj, cx + iAdj + 1, cy + iAdj + 1) ;

                // Tick marks

    for (i = 1 ; i < 96 ; i++)

    {

  if (i % 16 == 0) iHeight = cy /  2 ;              // inches

  else if (i %  8 == 0)     iHeight = cy /  3 ;          // half inches

  else if (i %  4 == 0)     iHeight = cy /  5 ;          // quarter in

 else if (i %  2 == 0)     iHeight = cy /  8 ;          // eighths

  else iHeight = cy / 12 ;         // sixteenths

          MoveToEx (hdc, i * cx / 96, cy, NULL) ;

          LineTo   (hdc, i * cx / 96, cy - iHeight) ;

    }

                // Create logical font

    FillMemory (&lf, sizeof (lf), 0) ;

    lf.lfHeight = cy / 2 ;
```

```
        lstrcpy (lf.lfFaceName, TEXT ("Times New Roman")) ;

        SelectObject (hdc, CreateFontIndirect (&lf)) ;
        SetTextAlign (hdc, TA_BOTTOM | TA_CENTER) ;
        SetBkMode    (hdc, TRANSPARENT) ;

                    // Display numbers

        for (i = 1 ; i <= 5 ; i++)
   {
              ch = (TCHAR) (i + '0') ;
              TextOut (hdc, i * cx / 6, cy / 2, &ch, 1) ;
   }
              // Clean up
        DeleteObject (SelectObject (hdc, GetStockObject (SYSTEM
        DeleteObject (SelectObject (hdc, GetStockObject (BLACK_
}

void CreateRoutine (HWND hwnd)
{
        HDC                         hdcEMF ;
```

```
        HENHMetaFile hemf ;
        int                         cxMms, cyMms, cxPix, cyPix, xDp

        hdcEMF = CreateEnhMetaFile (NULL, TEXT ("emf12.emf")
                        TEXT ("EMF13\0EMF Demo #12\0")) ;

        cxMms = GetDeviceCaps (hdcEMF, HORZSIZE) ;
        cyMms = GetDeviceCaps (hdcEMF, VERTSIZE) ;
        cxPix = GetDeviceCaps (hdcEMF, HORZRES) ;
        cyPix = GetDeviceCaps (hdcEMF, VERTRES) ;

        xDpi = cxPix * 254 / cxMms / 10 ;
        yDpi = cyPix * 254 / cyMms / 10 ;

        DrawRuler (hdcEMF, 6 * xDpi, yDpi) ;
        hemf = CloseEnhMetaFile (hdcEMF) ;
        DeleteEnhMetaFile (hemf) ;
}
```

```
void PaintRoutine (HWND hwnd, HDC hdc, int cxArea, int cyArea
{
          ENHMETAHEADER        emh ;
          HENHMetaFile         hemf ;
          POINT                                pt ;
          int                                  cxImage, cyImage ;
          RECT                                 rect ;

          SetMapMode (hdc, MM_HIMETRIC) ;
          SetViewportOrgEx (hdc, 0, cyArea, NULL) ;
          pt.x = cxArea ;
          pt.y = 0 ;

          DPtoLP (hdc, &pt, 1) ;
          hemf = GetEnhMetaFile (TEXT ("emf12.emf")) ;
          GetEnhMetaFileHeader (hemf, sizeof (emh), &emh) ;
          cxImage      = emh.rclFrame.right - emh.rclFrame.left ;
          cyImage      = emh.rclFrame.bottom - emh.rclFrame.top ;
```

```
        rect.left           = (pt.x - cxImage) / 2 ;
        rect.top            = (pt.y + cyImage) / 2 ;
        rect.right          = (pt.x + cxImage) / 2 ;
        rect.bottom           = (pt.y - cyImage) / 2 ;

        PlayEnhMetaFile (hdc, hemf, &rect) ;
        DeleteEnhMetaFile (hemf) ;
}
```

EMF12PaintRoutineMM_HIMETRICyy
SetViewportOrgEx

(cxArea,0)DPtoLP0.01

MetaFile0.01MetaFile

MetaFile

18-15 EMF13

18-15　EMF13

```
EMF13.C
/*------------------------------------------------------------------------
  EMF13.C --    Enhanced MetaFile Demo #13
```

```
                                        (c) Charles Petzold, 1998
------------------------------------------------------------------------*/
#include <windows.h>
TCHAR szClass              [] = TEXT ("EMF13") ;
TCHAR szTitle              [] = TEXT ("EMF13: Enhanced MetaFile De

void CreateRoutine (HWND hwnd)
{
}

void PaintRoutine (HWND hwnd, HDC hdc, int cxArea, int cyArea)
{
          ENHMETAHEADER        emh ;
          HENHMetaFile         hemf ;
          POINT                                pt ;
          int                                cxImage, cyImage ;
          RECT                                 rect ;

          SetMapMode (hdc, MM_HIMETRIC) ;
```

```
        SetViewportOrgEx (hdc, 0, cyArea, NULL) ;

        pt.x   = cxArea ;

        pt.y   = 0 ;


        DPtoLP (hdc, &pt, 1) ;


        hemf = GetEnhMetaFile (TEXT ("..\\emf11\\emf11.emf")) ;


        GetEnhMetaFileHeader (hemf, sizeof (emh), &emh) ;


        cxImage      = emh.rclFrame.right - emh.rclFrame.left ;

        cyImage      = emh.rclFrame.bottom - emh.rclFrame.top ;


        rect.left           = (pt.x - cxImage) / 2 ;

        rect.top             = (pt.y + cyImage) / 2 ;

        rect.right           = (pt.x + cxImage) / 2 ;

        rect.bottom          = (pt.y - cyImage) / 2 ;


        PlayEnhMetaFile (hdc, hemf, &rect) ;
```

```
        DeleteEnhMetaFile (hemf) ;

}
```

EMF13 MetaFile EMF11 MetaFileEMF13 MetaFileEMF11

MetaFile GDIMetaFile
GDI PlayEnhMetaFile M

MDIMicrosoft WindowsWindowsMDIWindows
MDIWindowsMicrosoft Excel

MDI

MDIWindows 2.0MDIWindows
95Windows 98Microsoft Windows NT

MDI

MDI//

MDI//

WindowsMDIWS_CHILDMDI

- MDIMDI
- MDI
- CtrlAltAlt+F4Ctrl+F4Ctrl+F6MDIAlt+
  Alt+-
- MDI
- Microsoft Excel
- WindowHelp

Windows 98MDI

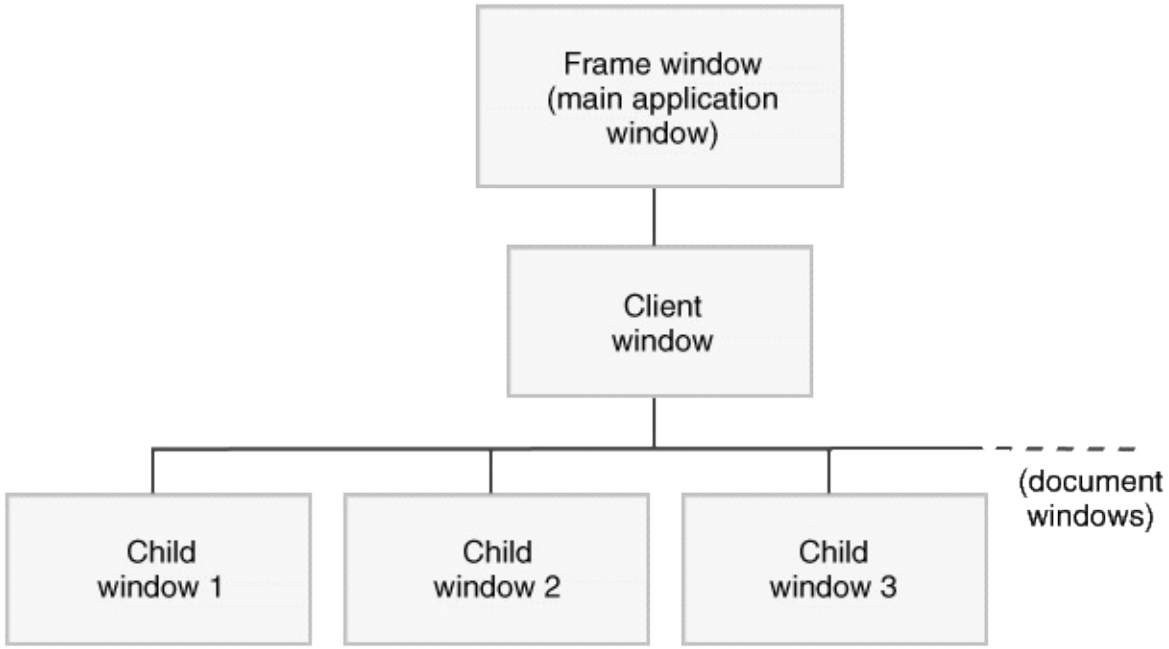
MDI

WindowsMDIWindowsWS_OVERLAPPEDWINDOW

MDIMDICLIENTWS_CHILDCreateWindow
CLIENTCREATESTRUCTMDICOLOR_APPWORKSPACE

MDICREATESTRUCTWM_MDICREATE

-19-1



19-1 Windows MDI-

Windows 98MDI12MDIMDICLIENT
CLIENTCREATESTRUCTMDICREATESTRUCTMDIDefWindowProc
DefWindowProcDefFrameProcDefMDIChildProcMDI
TranslateMDISysAccelTranslateAcceleratorMDIArrangeIconicWindows

MDIMDI

MDICreateMDIWindow

12MDI9WM_MDIWM_MDICREATE
WM_MDIACTIVATE

MDI

19-1 MDIDEMOMDI

19-1　　MDIDEMO

MDIDEMO.C

```
/*----------------------------------------------------------------------------
  MDIDEMO.C -- Multiple-Document Interface Demonstration
                                                (c) Charles Petzold, 1998
----------------------------------------------------------------------------*/
#include <windows.h>
#include "resource.h"

#define INIT_MENU_POS                0
#define HELLO_MENU_POS               2
#define RECT_MENU_POS                1

#define IDM_FIRSTCHILD            50000
```

```
LRESULT     CALLBACK FrameWndProc                   (HWND, UIN
BOOL            CALLBACK CloseEnumProc          (HWND, LPAR
LRESULT     CALLBACK HelloWndProc                   (HWND, UINT
LRESULT     CALLBACK RectWndProc                    (HWND, UINT

        // structure for storing data unique to each Hello child win

typedef struct tagHELLODATA
{
        UINT                    iColor ;
        COLORREF            clrText ;
}
HELLODATA, * PHELLODATA ;
        // structure for storing data unique to each Rect child win
typedef struct tagRECTDATA
{
        short       cxClient ;
        short       cyClient ;
}
```

```
RECTDATA, * PRECTDATA ;

        // global variables

TCHAR                   szAppName[]         = TEXT ("MDIDem
TCHAR                   szFrameClass[]      = TEXT ("MdiFram
TCHAR                   szHelloClass[]      = TEXT ("MdiHelloC
TCHAR                   szRectClass[]       = TEXT ("MdiRectCh
HINSTANCE       hInst ;
HMENU                   hMenuInit, hMenuHello, hMenuRect ;
HMENU                   hMenuInitWindow, hMenuHelloWindow

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                        PSTR szCmdLine, int iCr
{
        HACCEL                  hAccel ;
        HWND                    hwndFrame, hwndClient ;
        MSG                     msg ;
        WNDCLASS                wndclass ;

        hInst = hInstance ;
```

```
                // Register the frame window class
     wndclass.style                  = CS_HREDRAW | CS_V
     wndclass.lpfnWndProc               = FrameWndProc ;
     wndclass.cbClsExtra                = 0 ;
     wndclass.cbWndExtra                = 0 ;
    wndclass.hInstance                  = hInstance ;
     wndclass.hIcon                     = LoadIcon (NULL, IDI_
     wndclass.hCursor                   = LoadCursor (NULL,
     wndclass.hbrBackground             = (HBRUSH) (COLOR_A
     wndclass.lpszMenuName              = NULL ;
     wndclass.lpszClassName             = szFrameClass ;

     if (!RegisterClass (&wndclass))
     {
               MessageBox (  NULL, TEXT ("This program requi
                                                szAppName, M
               return 0 ;
}
```

```
                // Register the Hello child window class

     wndclass.style                                  = CS_HREDRA
     wndclass.lpfnWndProc                    = HelloWndProc ;
     wndclass.cbClsExtra                     = 0 ;
     wndclass.cbWndExtra                      = sizeof (HANDLE)
     wndclass.hInstance                      = hInstance ;
     wndclass.hIcon                          = LoadIcon (NULL, IDI_
     wndclass.hCursor                         = LoadCursor (NULL,
     wndclass.hbrBackground             = (HBRUSH) GetStockO
     wndclass.lpszMenuName               = NULL ;
     wndclass.lpszClassName             = szHelloClass ;

     RegisterClass (&wndclass) ;
                // Register the Rect child window class
     wndclass.style                          = CS_HREDRAW | CS_V
    wndclass.lpfnWndProc                     = RectWndProc ;
     wndclass.cbClsExtra                     = 0 ;
     wndclass.cbWndExtra                      = sizeof (HANDLE)
```

```
     wndclass.hInstance                  = hInstance ;
     wndclass.hIcon                      = LoadIcon (NULL, IDI_
    wndclass.hCursor                     = LoadCursor (NULL, ID
     wndclass.hbrBackground              = (HBRUSH) GetStockO
     wndclass.lpszMenuName               = NULL ;
     wndclass.lpszClassName              = szRectClass ;

     RegisterClass (&wndclass) ;
               // Obtain handles to three possible menus & sub
     hMenuInit           = LoadMenu    (hInstance, TEXT ("MdiM
     hMenuHello          = LoadMenu    (hInstance, TEXT ("Mdi
     hMenuRect           = LoadMenu    (hInstance, TEXT ("Mdi

     hMenuInitWindow           = GetSubMenu (hMenuInit,   IN
     hMenuHelloWindow          = GetSubMenu (hMenuHello,
     hMenuRectWindow           = GetSubMenu (hMenuRect,

               // Load accelerator table
```

```
hAccel = LoadAccelerators (hInstance, szAppName) ;

            // Create the frame window

hwndFrame     = CreateWindow (szFrameClass, TEXT ("MI
        WS_OVERLAPPEDWINDOW | WS_CLIPCHILDREN,
        CW_USEDEFAULT, CW_USEDEFAULT,
       CW_USEDEFAULT, CW_USEDEFAULT,
        NULL, hMenuInit, hInstance, NULL) ;
hwndClient = GetWindow (hwndFrame, GW_CHILD) ;
ShowWindow (hwndFrame, iCmdShow) ;
UpdateWindow (hwndFrame) ;


            // Enter the modified message loop

while (GetMessage (&msg, NULL, 0, 0))
{
     if (   !TranslateMDISysAccel (hwndClient, &msg) &&
                    !TranslateAccelerator (hwndFrame, hAc
            {
                     TranslateMessage (&msg) ;
                    DispatchMessage (&msg) ;
```

```
                }
        }
                // Clean up by deleting unattached menus
        DestroyMenu (hMenuHello) ;
        DestroyMenu (hMenuRect) ;

        return msg.wParam ;
        }

LRESULT CALLBACK FrameWndProc (HWND hwnd, UINT message
{
        static HWND                 hwndClient ;
        CLIENTCREATESTRUCT          clientcreate ;
        HWND                        hwndChild ;
        MDICREATESTRUCT             mdicreate ;

        switch (message)
        {
```

```
        case   WM_CREATE:                              // Create the clie


    clientcreate.hWindowMenu         = hMenuInitWindow ;
      clientcreate.idFirstChild          = IDM_FIRSTCHILD ;


    hwndClient = CreateWindow  (     TEXT ("MDICLIENT"), NUL
                                WS_CHILD | WS_CLIPCHILDREN | WS_V
                                0, 0, 0, 0, hwnd, (HMENU) 1, hInst,
                                (PSTR) &clientcreate) ;
                    return 0 ;
        case   WM_COMMAND:
                    switch (LOWORD (wParam))
                    {
                    case   IDM_FILE_NEWHELLO:  // Create a Hello cl
                    mdicreate.szClass           = szHelloClass ;
                                mdicreate.szTitle          = TEXT ("Hell
                                mdicreate.hOwner           = hInst ;
                                mdicreate.x                      = CW
                                mdicreate.y                      = CW
```

```
                    mdicreate.cx                = CW_US
                    mdicreate.cy                = CW_US
                    mdicreate.style             = 0 ;
                    mdicreate.lParam            = 0 ;

  hwndChild = (HWND) SendMessage (hwndClient,
WM_MDICREATE, 0, (LPARAM) (LPMDICREATESTRUCT) &
   return 0 ;

          case   IDM_FILE_NEWRECT:     // Create a Rect ch

          mdicreate.szClass             = szRectClass ;
          mdicreate.szTitle             = TEXT ("Rectangles
          mdicreate.hOwner              = hInst ;
          mdicreate.x                   = CW_USEDEFAULT
          mdicreate.y                   = CW_USEDEFAULT
          mdicreate.cx                  = CW_USEDEFAULT
          mdicreate.cy                  = CW_USEDEFAULT
```

```
                mdicreate.style                 = 0 ;
                mdicreate.lParam                  = 0 ;

               hwndChild = (HWND) SendMessage (hwndClient
                    WM_MDICREATE, 0,
                   (LPARAM) (LPMDICREATESTRUCT) &mdicreat
                                return 0 ;

                case   IDM_FILE_CLOSE:       // Close the active w

               hwndChild = (HWND) SendMessage (hwndClient
                    WM_MDIGETACTIVE, 0, 0) ;

                if (SendMessage (hwndChild, WM_QUERYENDSE
                        SendMessage (hwndClient, WM_MDIDESTR
            (WPARAM) hwndChild, 0) ;
                        return 0 ;

                case   IDM_APP_EXIT:// Exit the program
```

```
                SendMessage (hwnd, WM_CLOSE, 0, 0) ;
                return 0 ;

                // messages for arranging windows
                case   IDM_WINDOW_TILE:
                           SendMessage (hwndClient, WM_MDITIL
                           return 0 ;

                case   IDM_WINDOW_CASCADE:
                           SendMessage (hwndClient, WM_MDICA
                           return 0 ;

                case   IDM_WINDOW_ARRANGE:
                           SendMessage (hwndClient, WM_MDIIC
                           return 0 ;

                case   IDM_WINDOW_CLOSEALL:  // Attempt to c
```

```
                    EnumChildWindows (hwndClient, Close
                    return 0 ;

               default:                  // Pass to active child...

                    hwndChild = (HWND) SendMessage (h
                   WM_MDIGETACTIVE, 0, 0) ;
                    if (IsWindow (hwndChild))
              SendMessage (hwndChild, WM_COMMAND, wPar

                    break ;    // ...and then to DefFrameProc
               }
               break ;

     case   WM_QUERYENDSESSION:
     case   WM_CLOSE:     // Attempt to close all children

        SendMessage (hwnd, WM_COMMAND, IDM_WINDOW_C
```

```
            if (NULL != GetWindow (hwndClient, GW_CHILD))
                return 0 ;

                    break ;       // i.e., call DefFrameProc
          case   WM_DESTROY:
                    PostQuitMessage (0) ;
                    return 0 ;
          }
       // Pass unprocessed messages to DefFrameProc (not DefWin

          return DefFrameProc (hwnd, hwndClient, message, wPara
}

BOOL CALLBACK CloseEnumProc (HWND hwnd, LPARAM lParam)
{
          if (GetWindow (hwnd, GW_OWNER)) // Check for icon title
                    return TRUE ;
```

```
        SendMessage (GetParent (hwnd), WM_MDIRESTORE, (WPA
        if (!SendMessage (hwnd, WM_QUERYENDSESSION, 0, 0))
      return TRUE ;
        SendMessage (GetParent (hwnd), WM_MDIDESTROY, (WPA
        return TRUE ;
}

LRESULT CALLBACK HelloWndProc (HWND hwnd, UINT message,
WPARAM wParam, LPARAM lParam)
{
        static COLORREF clrTextArray[] = {   RGB (0,   0, 0), RGB
      RGB (0, 255, 0), RGB (  0, 0, 255),
     RGB (255, 255, 255) } ;
        static HWND               hwndClient, hwndFrame ;
        HDC                     hdc ;
        HMENU                    hMenu ;
        PHELLODATA                pHelloData ;
        PAINTSTRUCT               ps ;
        RECT                    rect ;
```

```
switch (message)
{
case   WM_CREATE:
   // Allocate memory for window private data

            pHelloData = (PHELLODATA) HeapAlloc (GetProc
                  HEAP_ZERO_MEMORY, sizeof (HELLODATA)
            pHelloData->iColor  = IDM_COLOR_BLACK ;
            pHelloData->clrText = RGB (0, 0, 0) ;
            SetWindowLong (hwnd, 0, (long) pHelloData) ;

                        // Save some window handles

            hwndClient = GetParent (hwnd) ;
            hwndFrame  = GetParent (hwndClient) ;
            return 0 ;

case   WM_COMMAND:
```

```
            switch (LOWORD (wParam))
            {
                  case IDM_COLOR_BLACK:
                  case IDM_COLOR_RED:
                  case IDM_COLOR_GREEN:
                  case IDM_COLOR_BLUE:
                  case IDM_COLOR_WHITE:
      // Change the text color

 pHelloData = (PHELLODATA) GetWindowLong (hwnd, 0)

 hMenu = GetMenu (hwndFrame) ;

 CheckMenuItem (hMenu, pHelloData->iColor, MF_UNCH
pHelloData->iColor = wParam ;
 CheckMenuItem (hMenu, pHelloData->iColor, MF_CHEC

 pHelloData->clrText = clrTextArray[wParam - IDM_COLC
```

```
    InvalidateRect (hwnd, NULL, FALSE) ;
   }
  return 0 ;

 case   WM_PAINT:
       // Paint the window

            hdc = BeginPaint (hwnd, &ps) ;

            pHelloData = (PHELLODATA) GetWindowLong (h
            SetTextColor (hdc, pHelloData->clrText) ;

            GetClientRect (hwnd, &rect) ;

            DrawText (hdc, TEXT ("Hello, World!"), -1, &rect,
        DT_SINGLELINE | DT_CENTER | DT_VCENTER) ;
            EndPaint (hwnd, &ps) ;
            return 0 ;
```

```
        case   WM_MDIACTIVATE:
                // Set the Hello menu if gaining focus

                if (lParam == (LPARAM) hwnd)
            SendMessage (hwndClient, WM_MDISETMENU,(WPARA
hMenuHello, (LPARAM) hMenuHelloWindow) ;

                                // Check or uncheck menu item

                pHelloData = (PHELLODATA) GetWindowLong (h
               CheckMenuItem (hMenuHello, pHelloData->iColo
                           (lParam == (LPARAM) hwnd) ? MF_CHE

                                // Set the Init menu if losing focus

                if (lParam != (LPARAM) hwnd)
                                SendMessage (hwndClient, WM_M
```

```
hMenuInit,(LPARAM) hMenuInitWindow) ;

                    DrawMenuBar (hwndFrame) ;
                    return 0 ;

        case   WM_QUERYENDSESSION:
        case   WM_CLOSE:
     if (IDOK != MessageBox (hwnd, TEXT ("OK to close window?
                         TEXT ("Hello"),
                MB_ICONQUESTION | MB_OKCANCEL))
                    return 0 ;

                    break ;             // i.e., call DefMDIChildProc

        case   WM_DESTROY:
                    pHelloData = (PHELLODATA) GetWindowLong (h
                    HeapFree (GetProcessHeap (), 0, pHelloData) ;
                    return 0 ;
   }
```

```
                        // Pass unprocessed message to DefMDIChi


        return DefMDIChildProc (hwnd, message, wParam, lParam
}
LRESULT CALLBACK RectWndProc (     HWND hwnd, UINT messag
{
        static HWND   hwndClient, hwndFrame ;
        HBRUSH        hBrush ;
        HDC           hdc ;
        PRECTDATA     pRectData ;
        PAINTSTRUCT   ps ;
        int                         xLeft, xRight, yTop, yBottom ;
        short                               nRed, nGreen, nBlue ;


        switch (message)
   {
        case   WM_CREATE:
                            // Allocate memory for window private


```

```
                pRectData = (PRECTDATA) HeapAlloc (GetProces
                   HEAP_ZERO_MEMORY, sizeof (RECTDATA)) ;

                SetWindowLong (hwnd, 0, (long) pRectData) ;

                                                // Start the timer g

                SetTimer (hwnd, 1, 250, NULL) ;

                                                // Save some windo
                hwndClient    = GetParent (hwnd) ;
               hwndFrame     = GetParent (hwndClient) ;
                return 0 ;

      case   WM_SIZE:    // If not minimized, save the window si

                if (wParam != SIZE_MINIMIZED)
                {
```

```
                        pRectData = (PRECTDATA) GetWi


                        pRectData->cxClient = LOWORD
                        pRectData->cyClient = HIWORD (
                }

                break ;                 // WM_SIZE must be proce

        case    WM_TIMER:               // Display a random rectar

                pRectData    = (PRECTDATA) GetWindowLong (h
                xLeft                = rand () % pRectData->cx(
                xRight             = rand () % pRectData->cxClie
                yTop                 = rand () % pRectData->cy
                yBottom           = rand () % pRectData->cyCli
                nRed                 = rand () & 255 ;
                nGreen            = rand () & 255 ;
                nBlue                = rand () & 255 ;
```

```
                hdc = GetDC (hwnd) ;
                hBrush = CreateSolidBrush (RGB (nRed, nGreen
                SelectObject (hdc, hBrush) ;

                Rectangle (hdc, min (xLeft, xRight), min (yTop, y
                                                max (xLeft, xRight)

               ReleaseDC (hwnd, hdc) ;
                DeleteObject (hBrush) ;
                return 0 ;

    case   WM_PAINT:                                    // Clear the v

                InvalidateRect (hwnd, NULL, TRUE) ;
                hdc = BeginPaint (hwnd, &ps) ;
                EndPaint (hwnd, &ps) ;
                return 0 ;
```

```
        case   WM_MDIACTIVATE:                    // Set the appropri
                    if (lParam == (LPARAM) hwnd)
                SendMessage (hwndClient, WM_MDISETMENU, (WPAR
                    else
                 SendMessage (hwndClient, WM_MDISETMENU, (WPAF


                          DrawMenuBar (hwndFrame) ;
                          return 0 ;


        case   WM_DESTROY:
                    pRectData = (PRECTDATA) GetWindowLong (hw
                   HeapFree (GetProcessHeap (), 0, pRectData) ;
                    KillTimer (hwnd, 1) ;
                    return 0 ;
   }
       // Pass unprocessed message to DefMDIChildProc
   return DefMDIChildProc (hwnd, message, wParam, lParam) ;
}
```

```
MDIDEMO.RC

//Microsoft Developer Studio generated resource script.

#include "resource.h"

#include "afxres.h"

//////////////////////////////////////////////////////////////////////////////

// Menu

MDIMENUINIT MENU DISCARDABLE

BEGIN

  POPUP "&File"

  BEGIN

                MENUITEM "New &Hello",                  IDM_FIL

                MENUITEM "New &Rectangle",              IDM_FIL

                MENUITEM SEPARATOR

                MENUITEM "E&xit", IDM_APP_EXIT

  END

END

MDIMENUHELLO MENU DISCARDABLE

BEGIN

  POPUP "&File"
```

```
  BEGIN
MENUITEM "New &Hello",            IDM_FILE_NEWHELLO
MENUITEM "New &Rectangle", IDM_FILE_NEWRECT
MENUITEM "&Close",                IDM_FILE_CLOSE
MENUITEM SEPARATOR
MENUITEM "E&xit",                 IDM_APP_EXIT
END
  POPUP "&Color"
  BEGIN
     MENUITEM "&Black",             IDM_COLOR_BLACK
     MENUITEM "&Red",               IDM_COLOR_RED
     MENUITEM "&Green",             IDM_COLOR_GREEN
     MENUITEM "B&lue",              IDM_COLOR_BLUE
     MENUITEM "&White",                     IDM_COLOR_WHITE
  END
  POPUP "&Window"
  BEGIN
     MENUITEM "&Cascade\tShift+F5",         IDM_WINDOW_CASC
```

```
    MENUITEM "&Tile\tShift+F4",          IDM_WINDOW_TILE
    MENUITEM "Arrange &Icons",               IDM_WINDOW_AI
    MENUITEM "Close &All",                   IDM_WINDOW_CLO
  END
END
MDIMENURECT MENU DISCARDABLE
BEGIN
  POPUP "&File"
  BEGIN
    MENUITEM "New &Hello",                   IDM_FILE_NEWHE
    MENUITEM "New &Rectangle",           IDM_FILE_NEWRECT
    MENUITEM "&Close",           IDM_FILE_CLOSE
    MENUITEM SEPARATOR
    MENUITEM "E&xit",                        IDM_APP_EXIT
  END
  POPUP "&Window"
  BEGIN
    MENUITEM "&Cascade\tShift+F5",       IDM_WINDOW_CASC
    MENUITEM "&Tile\tShift+F4",          IDM_WINDOW_TILE
```

```
    MENUITEM "Arrange &Icons",          IDM_WINDOW_ARRANG
    MENUITEM "Close &All",              IDM_WINDOW_CLOSEALL
  END
END

//////////////////////////////////////////////////////////////////////////
// Accelerator
MDIDEMO ACCELERATORS DISCARDABLE
BEGIN
  VK_F4,    IDM_WINDOW_TILE,     VIRTKEY, SHIFT, NOINVERT
  VK_F5,    IDM_WINDOW_CASCADE,  VIRTKEY, SHIFT, NOINVERT
END
```

RESOURCE.H

```
// Microsoft Developer Studio generated include file.
// Used by MDIDemo.rc
#define IDM_FILE_NEWHELLO
#define IDM_FILE_NEWRECT
#define IDM_APP_EXIT                                        400
```

```
#define IDM_FILE_CLOSE                                  40

#define IDM_COLOR_BLACK

#define IDM_COLOR_RED                                    4

#define IDM_COLOR_GREEN

#define IDM_COLOR_BLUE                                   4

#define IDM_COLOR_WHITE

#define IDM_WINDOW_CASCADE

#define IDM_WINDOW_TILE

#define IDM_WINDOW_ARRANGE

#define IDM_WINDOW_CLOSEALL                              4
```

MDIDEMO"Hello,
World!"

MDIDEMO.RC

MdiMenuInit

MdiMenuHelloHello, World!FileColorWindow

MdiMenuRectColorMdiMenuHello

RESOURCE.HMDIDEMO.C

```
#define INIT_MENU_POS           0
```

```
#define HELLO_MENU_POS          2

#define RECT_MENU_POS           1
```

WindowsMdiMenuInitWindows0

MDIDEMO.CIDM_FIRSTCHILDWindowsID

MDIDEMO.CWinMainFrameWndProcHelloWndProcRectWndProc
IDI_APPLICATION

WNDCLASShbrBackgroundCOLOR_APPWORKSPACE

lpszMenuNameNULLHelloRectCreateWindow

HelloRectWNDCLASScbWndExtraHELLODATARECTDATA
MDIDEMO.C

WinMainLoadMenuGetSubMenuWindows
LoadAccelerators

WinMainCreateWindowFrameWndProcWM_CREATE
CreateWindowMDICLIENTMDIWindowsMDIMDICLIENT
CreateWindowCLIENTCREATESTRUCT

- hWindowMenuMDIDEMOhMenuInitWindowWinMain
- idFirstChildIDIDM_FIRSTCHILD.

WinMainMDIDEMOGetMessageMDITranslateMDISysAccel
TranslateAcceleratorMDIDEMO

TranslateMDISysAccelMDICtrl-F6WM_SYSCOMMAND

TranslateMDISysAccelTranslateAcceleratorTRUETranslateMessage DispatchMessage

TranslateMDISysAccelTranslateAcceleratorhwndClienthwndFrame WinMainGW_CHILDGetWindowhwndClient

FrameWndProcWM_COMMANDFrameWndProcwParamID

IDIDM_FILE_NEWHELLOIDM_FILE_NEWRECTFrameWndProc MDICREATESTRUCTCreateWindowWM_MDICREATElParam CreateMDIWindow

MDICREATESTRUCTszTitleWS_HSCROLLWS_VSCROLL WS_MINIMIZEWS_MAXIMIZE

MDICREATESTRUCTlParamWM_CREATElParam CREATESTRUCTlpCreateParamsMDICREATESTRUCT

WM_MDICREATEMDICLIENTSTRUCTMDIDEMO MdiMenuInitFileMdiMenuHelloMdiMenuRectWindows

9199More

FrameWndProc

FileCloseMDIDEMOWM_MDIGETACTIVE WM_QUERYENDSESSIONMDIDEMOWM_MDIDESTROY

FileExitWM_CLOSE

WindowTileCascadeArrangeWM_MDITILE WM_MDICASCADEWM_MDIICONARRANGE

Close AllFrameWndProcEnumChildWindowsCloseEnumProc

WM_MDIRESTOREWM_QUERYENDSESSIONWM_MDIDESTROY
GW_OWNERGetWindowNULL

FrameWndProcColorWM_COMMANDFrameWndProc
WM_COMMAND

DefFrameProcDefWindowProcWM_MENUCHARWM_SETFOCUS
WM_SIZEDefFrameProc

WM_COMMANDDefFrameProcFrameWndProcWM_COMMAND
WindowswParamIDM_FIRSTCHILDDefFrameProc

Close

HelloWndProcHello, World!

WNDCLASScbWndExtra

MDIDEMOHELLODATAWM_CREATEHelloWndProc
SetWindowLong

WM_COMMANDHelloWndProcGetWindowLong
HELLODATAHelloWndProc

WM_MDIACTIVATElParamMDIDEMOMdiMenuInit
HelloMdiMenuHelloRectMdiMenuRect

WM_MDIACTIVATElParamHelloWndProcMdiMenuHello
lParamHelloWndProcMdiMenuInit

HelloWndProcWM_MDISETMENUMdiMenuInit
MdiMenuHelloMDISetMenu

ColorHelloWndProc

WM_MDIACTIVATEwParamlParamWM_MDIACTIVATElParam

lParamWM_MDIACTIVATElParamMdiMenuInit
WM_MDIACTIVATElParamMdiMenuHelloMdiMenuRect
MdiMenuInit

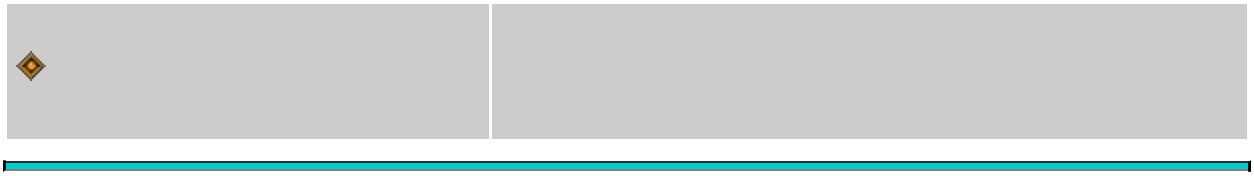
CloseClose AllFrameWndProcWM_QUERYENDSES
HelloWndProcWM_QUERYENDSESSIONWM_CLOSE
0

WM_DESTROYHelloWndProcWM_CREATE

DefMDIChildProcDefWindowProcDefMDIChildProc
WM_CHILDACTIVATEWM_GETMINMAXINFOWM_MENUCHAR
WM_MOVEWM_SETFOCUSWM_SIZEWM_SYSCOMMAND

RectWndProcHelloWndProcHelloWndProcWM_SIZE
RectWndProcbreakWM_SIZEDefMDIChildProc

WinMainMDIDEMOLoadMenuWindowsInit
MDIDEMOWinMainDestroyMenuHelloRect



PCMicrosoft

PC

DOS

PCIntel 8088 8088

DOS

DOSTSRterminate-and-stay-residentTSRTSR
SideKickDOSTSR

DOSshellQuarterdeckDesqViewWindows

Microsoft1985Windows 1.0DOSWindows

UNIX

WindowsDDEOLE

WindowsWindows16WindowsWindows
Windows

Windows16timer tickWindowsWindowsWindows

16WindowsWindowsDOS

16WindowsWindows

PeekMessage RANDRECT
PeekMessagePeekMessage

Presentation Manager

MicrosoftDOS/WindowsIBMOS/2Presentation
Presentation ManagerPMPM

PMWindows

32Windows

OS/2Presentation ManagerWindowsPM32Windows

PMWindows

CmainWindowsWinMainCreateThread

OS/2Presentation ManagerPM

PMPM1/101/10PM1/10

PM

Windows

race conditionsemaphore
critical section

deadlock

3216

```
lCount++ ;
```

lCount32longC16116lCount0x0000FFFFlCount0
lCount0x00010000

1632

Windows

32WindowsWindows NTWindows 98

Windows NTWindows 98PM

Windows NTWindows 98OS/2

Windows NTWindows 98TLSthread l

WindowsMicrosoftC

PeekMessage

1/101/10

Windows

APICreateThread

```
hThread = CreateThread (&security_attributes, dwStackSize, Thr
                        pParam, dwFlags, &idThread) ;
```

SECURITY_ATTRIBUTESWindows 98Wind
0Windows

CreateThread

```
DWORD WINAPI ThreadProc (PVOID pParam) ;
```

CreateThreadThreadProc

CreateThread0CREATE_SUSPENDEDResumeThreadID

WindowsPROCESS.HC_beginthread

```
hThread = _beginthread (ThreadProc, uiStackSize, pParam) ;
```

```
void __cdecl ThreadProc (void * pParam) ;
```

20-1 RNDRCTMTRANDRECTRANDRECTPeekMessage

20-1 RNDRCTMT

```
RNDRCTMT.C

/*-----------------------------------------------------------------------
  RNDRCTMT.C -- Displays Random Rectangles
                                          (c) Charles Petzold, 1998
-----------------------------------------------------------------------*/

#include <windows.h>
#include <process.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
HWND       hwnd ;
int cxClient, cyClient ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                PSTR szCmdLine, int iCmdShow)
{
        static TCHAR szAppName[] = TEXT ("RndRctMT") ;
         MSG         msg ;
         WNDCLASS    wndclass ;
```

```
     wndclass.style                  = CS_HREDRAW | CS_V
     wndclass.lpfnWndProc              = WndProc ;
     wndclass.cbClsExtra               = 0 ;
     wndclass.cbWndExtra               = 0 ;
     wndclass.hInstance                = hInstance ;
     wndclass.hIcon                  = LoadIcon (NULL, IDI_
     wndclass.hCursor                = LoadCursor (NULL,
     wndclass.hbrBackground          = (HBRUSH) GetStockO
     wndclass.lpszMenuName           = NULL ;
     wndclass.lpszClassName          = szAppName ;

     if (!RegisterClass (&wndclass))
     {
   MessageBox (NULL, TEXT ("This program requires Windows
                szAppName, MB_ICONERROR) ;
          return 0 ;
}
     hwnd = CreateWindow ( szAppName, TEXT ("Random Rec
```

```
                    WS_OVERLAPPEDWINDOW,
                    CW_USEDEFAULT, CW_USEDEFAULT,
                   CW_USEDEFAULT, CW_USEDEFAULT,
                    NULL, NULL, hInstance, NULL) ;

          ShowWindow (hwnd, iCmdShow) ;
          UpdateWindow (hwnd) ;

          while (GetMessage (&msg, NULL, 0, 0))
     {
               TranslateMessage (&msg) ;
               DispatchMessage (&msg) ;
          }
          return msg.wParam ;
}

VOID Thread (PVOID pvoid)
{
     HBRUSH hBrush ;
```

```
     HDC         hdc ;
     int         xLeft, xRight, yTop, yBottom, iRed, iGreen, iBlue


     while (TRUE)
{
          if (cxClient != 0 || cyClient != 0)
          {
                    xLeft              = rand () % cxClient ;
                    xRight              = rand () % cxClient ;
                    yTop         = rand () % cyClient ;
                    yBottom       = rand () % cyClient ;
                   iRed                = rand () & 255 ;
                    iGreen              = rand () & 255 ;
                    iBlue               = rand () & 255 ;

                   hdc = GetDC (hwnd) ;
                   hBrush = CreateSolidBrush (RGB (iRed, iGreen,
                   SelectObject (hdc, hBrush) ;
```

```
            Rectangle (hdc,min (xLeft, xRight), min (yTop, yBotto
          max (xLeft, xRight), max (yTop, yBottom)) ;


                              ReleaseDC (hwnd, hdc) ;
                              DeleteObject (hBrush) ;
                    }
        }
}
LRESULT CALLBACK WndProc ( HWND hwnd, UINT message, WPA
{
        switch (message)
        {
        case WM_CREATE:
                    _beginthread (Thread, 0, NULL) ;
                    return 0 ;


        case   WM_SIZE:
                    cxClient = LOWORD (lParam) ;
```

```
                cyClient = HIWORD (lParam) ;
                return 0 ;

        case   WM_DESTROY:
                PostQuitMessage (0) ;
               return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam)
}
```

WindowsProject SettingsC/C++CategoryCode GenerationUse Run-Time LibraryReleaseSingle-Threaded DebugDebug Single-ThreadedMultithreadedDebug Multithreaded/MT.OBJLIBCMT.LIBLIBC.LIB

LIBC.LIBLIBCMT.LIBCCstrtokstrtokstrtok

RNDRCTMT.CPROCESS.H_beginthread_MT/MT

RNDRCTMT.CWinMainCreateWindowhwndcxClientcyClient WM_SIZE

_beginthreadThread0VOIDVOIDRNDRCTMTThread

_beginthreadcxClientcyClientThread

1986103MicrosoftQuickBASIC
MicrosoftStorm

FibonacciFibonacci010112358
Escape

198610DOSWindows

EscapeDOSCPU

Windows 1.020-2 MULTI132Unicode

20-2 MULTI1

```
MULTI1.C

/*--------------------------------------------------------------------------

  MULTI1.C --   Multitasking Demo

                                          (c) Charles Petzold, 1998

--------------------------------------------------------------------------*/

#include <windows.h>

#include <math.h>


LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int cyChar ;
```

```
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                    PSTR szCmdLine, int iCmdShow)
{
          static TCHAR szAppName[] = TEXT ("Multi1") ;
          HWND                         hwnd ;
          MSG                          msg ;
          WNDCLASS                     wndclass ;

          wndclass.style                = CS_HREDRAW | CS_VREDR
          wndclass.lpfnWndProc             = WndProc ;
          wndclass.cbClsExtra              = 0 ;
          wndclass.cbWndExtra              = 0 ;
          wndclass.hInstance               = hInstance ;
          wndclass.hIcon                = LoadIcon (NULL, IDI_APPL
          wndclass.hCursor                 = LoadCursor (NULL, IDC_A
          wndclass.hbrBackground        = (HBRUSH) GetStockObjec
          wndclass.lpszMenuName         = NULL ;
          wndclass.lpszClassName        = szAppName ;
```

```
        if (!RegisterClass (&wndclass))
        {
    MessageBox (NULL, TEXT ("This program requires Windows
                szAppName, MB_ICONERROR) ;
               return 0 ;
}
hwnd = CreateWindow ( szAppName, TEXT ("Multitasking Der
                    WS_OVERLAPPEDWINDOW,
                     CW_USEDEFAULT, CW_USEDEFAULT,
                     CW_USEDEFAULT, CW_USEDEFAULT,
                     NULL, NULL, hInstance, NULL) ;

        ShowWindow (hwnd, iCmdShow) ;
        UpdateWindow (hwnd) ;

       while (GetMessage (&msg, NULL, 0, 0))
        {
             TranslateMessage (&msg) ;
```

```
            DispatchMessage (&msg) ;
        }
    return msg.wParam ;
}

int CheckBottom (HWND hwnd, int cyClient, int iLine)
{
        if (iLine * cyChar + cyChar > cyClient)
        {
            InvalidateRect (hwnd, NULL, TRUE) ;
            UpdateWindow (hwnd) ;
            iLine = 0 ;
    }
        return iLine ;
}

// ------------------------------------------------------------------
// Window 1: Display increasing sequence of numbers
// ------------------------------------------------------------------
```

```
LRESULT APIENTRY WndProc1 (HWND hwnd, UINT message, WPA
{
        static int        iNum, iLine, cyClient ;
        HDC                         hdc ;
        TCHAR                        szBuffer[16] ;

        switch (message)
        {
        case   WM_SIZE:
                cyClient = HIWORD (lParam) ;
                return 0 ;

        case   WM_TIMER:
                if (iNum < 0)
                                iNum = 0 ;

                iLine = CheckBottom (hwnd, cyClient, iLine) ;
                hdc = GetDC (hwnd) ;

                TextOut (hdc, 0, iLine * cyChar, szBuffer,
```

```
                                        wsprintf (szBuffer, TEXT ("%

                    ReleaseDC (hwnd, hdc) ;

                    iLine++ ;

                    return 0 ;

          }

          return DefWindowProc (hwnd, message, wParam, lParam)

}

// ------------------------------------------------------------------------

// Window 2: Display increasing sequence of prime numbers

// ------------------------------------------------------------------------

LRESULT APIENTRY WndProc2 (HWND hwnd, UINT message, WPA

{

          static int          iNum = 1, iLine, cyClient ;

          HDC                           hdc ;

          int                         i, iSqrt ;

          TCHAR                         szBuffer[16] ;
```

```
        switch (message)
        {
        case   WM_SIZE:
                   cyClient = HIWORD (lParam) ;
                   return 0 ;

        case   WM_TIMER:
                   do          {
                                   if (++iNum < 0)
                                              iNum = 0 ;

                                   iSqrt = (int) sqrt (iNum) ;

                                  for (i = 2 ; i <= iSqrt ; i++)
                                              if (iNum % i == 0)
                                                         break ;
                  }
                   while (i <= iSqrt) ;
```

```
                    iLine = CheckBottom (hwnd, cyClient, iLine) ;

                    hdc = GetDC (hwnd) ;

                    TextOut (     hdc, 0, iLine * cyChar, szBuffer,
                                                wsprintf (szBuffer, TEXT

                    ReleaseDC (hwnd, hdc) ;

                    iLine++ ;

                    return 0 ;
          }
          return DefWindowProc (hwnd, message, wParam, lParam)
}

// ------------------------------------------------------------------------
// Window 3: Display increasing sequence of Fibonacci numbers
// ------------------------------------------------------------------------

LRESULT APIENTRY WndProc3 (HWND hwnd, UINT message, WPA
{
          static int          iNum = 0, iNext = 1, iLine, cyClient ;
          HDC                           hdc ;
```

```
int                     iTemp ;
TCHAR                       szBuffer[16] ;

switch (message)
{
case WM_SIZE:
        cyClient = HIWORD (lParam) ;
        return 0 ;

case   WM_TIMER:
        if (iNum < 0)
        {
                iNum  = 0 ;
                iNext = 1 ;
        }

        iLine = CheckBottom (hwnd, cyClient, iLine) ;
        hdc = GetDC (hwnd) ;
```

```
                    TextOut (     hdc, 0, iLine * cyChar, szBuffer,
                                                wsprintf (szBuffer, "%d",
                    ReleaseDC (hwnd, hdc) ;
                    iTemp        =      iNum ;
                    iNum         =      iNext ;
                   iNex         +=     iTemp ;
                    iLine++ ;
                    return 0 ;
          }
          return DefWindowProc (hwnd, message, wParam, lParam)
}

// ---------------------------------------------------------------------
// Window 4: Display circles of random radii
// ---------------------------------------------------------------------

LRESULT APIENTRY WndProc4 (HWND hwnd, UINT message, WPA
{
          static int          cxClient, cyClient ;
          HDC                           hdc ;
```

```
int                         iDiameter ;

switch (message)
{
case   WM_SIZE:
            cxClient = LOWORD (lParam) ;
            cyClient = HIWORD (lParam) ;
            return 0 ;

case   WM_TIMER:
            InvalidateRect (hwnd, NULL, TRUE) ;
           UpdateWindow (hwnd) ;

            iDiameter = rand() % (max (1, min (cxClient, cy
            hdc = GetDC (hwnd) ;

            Ellipse (hdc,        (cxClient - iDiameter) / 2,
                        (cyClient - iDiameter) / 2,
```

```
                              (cxClient + iDiameter) / 2,
                              (cyClient + iDiameter) / 2) ;

                    ReleaseDC (hwnd, hdc) ;
                    return 0 ;
          }
          return DefWindowProc (hwnd, message, wParam, lParam)
}
// ---------------------------------------------------------------------
// Main window to create child windows
// ---------------------------------------------------------------------

LRESULT APIENTRY WndProc ( HWND hwnd, UINT message, WPA
{
          static HWND          hwndChild[4] ;
         static TCHAR * szChildClass[] =      { TEXT ("Child1"), TEXT
       TEXT ("Child3"), TEXT ("Child4") } ;
          static WNDPROC       ChildProc[] = { WndProc1, WndProc
          HINSTANCE                 hInstance ;
```

```
int                         i, cxClient, cyClient ;
WNDCLASS                    wndclass ;

switch (message)
{
case   WM_CREATE:
          hInstance = (HINSTANCE) GetWindowLong (hwn

           wndclass.style                    = CS_HR
          wndclass.cbClsExtra          = 0 ;
           wndclass.cbWndExtra          = 0 ;
           wndclass.hInstance           = hInstance ;
           wndclass.hIcon               = NULL ;
           wndclass.hCursor             = LoadCursor (NU
          wndclass.hbrBackground       = (HBRUSH) GetSt
           wndclass.lpszMenuName        = NULL ;

           for (i = 0 ; i < 4 ; i++)
           {
```

```
                    wndclass.lpfnWndProc   = ChildPr
                    wndclass.lpszClassName = szChil

                    RegisterClass (&wndclass) ;

                    hwndChild[i] = CreateWindow (sz
                WS_CHILDWINDOW | WS_BORDER | WS_VISIB
                0, 0, 0, 0,
               hwnd, (HMENU) i, hInstance, NULL) ;
            }

            cyChar = HIWORD (GetDialogBaseUnits ()) ;
            SetTimer (hwnd, 1, 10, NULL) ;
            return 0 ;

    case   WM_SIZE:
            cxClient = LOWORD (lParam) ;
            cyClient = HIWORD (lParam) ;
```

```
                for (i = 0 ; i < 4 ; i++)
          MoveWindow (hwndChild[i],     (i % 2) * cxClient / 2,
                                  (i > 1) * cyClient / 2,
               cxClient / 2, cyClient / 2, TRUE) ;
                     return 0 ;

   case   WM_TIMER:
                for (i = 0 ; i < 4 ; i++)
                           SendMessage (hwndChild[i], WM_TIME

                return 0 ;

  case   WM_CHAR:
                if (wParam == '\x1B')
                                    DestroyWindow (hwnd) ;

                return 0 ;
```

```
        case   WM_DESTROY:
                    KillTimer (hwnd, 1) ;
                    PostQuitMessage (0) ;
                    return 0 ;
   }
    return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

WindowsWM_TIMER

WindowsWM_PAINTMULTI1

WndProc21

MULTI1WindowsWindowsWM_TIMER

WM_TIMER25MHz38650MHz486100-GHzPentium

20-3 MULTI2

20-3 MULTI2

```
MULTI2.C
/*------------------------------------------------------------------------
  MULTI2.C --   Multitasking Demo
```

```
                                        (c) Charles Petzold, 1998
----------------------------------------------------------------------*/
#include <windows.h>
#include <math.h>
#include <process.h>

typedef struct
{
     HWND        hwnd ;
     int         cxClient ;
     int         cyClient ;
     int         cyChar ;
     BOOL bKill ;
}
PARAMS, *PPARAMS ;
LRESULT APIENTRY WndProc (HWND, UINT, WPARAM, LPARAM) ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
               PSTR szCmdLine, int iCmdShow)
{
```

```
     static TCHAR szAppName[] = TEXT ("Multi2") ;
     HWND                        hwnd ;
     MSG                         msg ;
     WNDCLASS                    wndclass ;

     wndclass.style                    = CS_HREDRAW | CS_V
     wndclass.lpfnWndProc                = WndProc ;
     wndclass.cbClsExtra                 = 0 ;
     wndclass.cbWndExtra                 = 0 ;
     wndclass.hInstance                  = hInstance ;
     wndclass.hIcon                    = LoadIcon (NULL, IDI_
     wndclass.hCursor                    = LoadCursor (NULL,
     wndclass.hbrBackground            = (HBRUSH) GetStockO
     wndclass.lpszMenuName             = NULL ;
     wndclass.lpszClassName            = szAppName ;

     if (!RegisterClass (&wndclass))
     {
MessageBox (NULL, TEXT ("This program requires Windows
```

```
        szAppName, MB_ICONERROR) ;
        return 0 ;
    }

    hwnd = CreateWindow ( szAppName, TEXT ("Multitasking
                WS_OVERLAPPEDWINDOW,
                CW_USEDEFAULT, CW_USEDEFAULT,
               CW_USEDEFAULT, CW_USEDEFAULT,
                NULL, NULL, hInstance, NULL) ;

   ShowWindow (hwnd, iCmdShow) ;
    UpdateWindow (hwnd) ;

    while (GetMessage (&msg, NULL, 0, 0))
    {
        TranslateMessage (&msg) ;
        DispatchMessage (&msg) ;
    }
```

```
        return msg.wParam ;
}

int CheckBottom (HWND hwnd, int cyClient, int cyChar, int iLine)
{
        if (iLine * cyChar + cyChar > cyClient)
        {
              InvalidateRect (hwnd, NULL, TRUE) ;
              UpdateWindow (hwnd) ;
             iLine = 0 ;
        }
        return iLine ;
}

// ---------------------------------------------------------------------
// Window 1: Display increasing sequence of numbers
// ---------------------------------------------------------------------

void Thread1 (PVOID pvoid)
{
```

```
        HDC                     hdc ;
        int                     iNum = 0, iLine = 0 ;
        PPARAMS             pparams ;
        TCHAR                   szBuffer[16] ;

        pparams      = (PPARAMS) pvoid ;

        while (!pparams->bKill)
        {
             if (iNum < 0)
                             iNum = 0 ;
             iLine = CheckBottom ( pparams->hwnd,       pparam
pparams->cyChar,iLine) ;

             hdc = GetDC (pparams->hwnd) ;

             TextOut (     hdc, 0, iLine * pparams->cyChar, szBuffe
                                  wsprintf (szBuffer, TEXT ("%c
```

```
            ReleaseDC (pparams->hwnd, hdc) ;
            iLine++ ;
        }
   _endthread () ;
}

LRESULT APIENTRY WndProc1 (HWND hwnd, UINT message, WPA
{
        static PARAMS params ;
        switch (message)
        {
       case   WM_CREATE:
                    params.hwnd = hwnd ;
                    params.cyChar = HIWORD (GetDialogBaseUnits
                    _beginthread (Thread1, 0, s) ;
                    return 0 ;

        case   WM_SIZE:
                   params.cyClient = HIWORD (lParam) ;
```

```
                return 0 ;

        case   WM_DESTROY:
                params.bKill = TRUE ;
                return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}

// ------------------------------------------------------------------------
// Window 2: Display increasing sequence of prime numbers
// ------------------------------------------------------------------------

void Thread2 (PVOID pvoid)
{
   HDC                  hdc ;
        int                 iNum = 1, iLine = 0, i, iSqrt ;
        PPARAMS             pparams ;
        TCHAR                   szBuffer[16] ;
```

```
        pparams = (PPARAMS) pvoid ;
        while (!pparams->bKill)
        {
             do
             {
                        if (++iNum < 0)
                                   iNum = 0 ;
                        iSqrt = (int) sqrt (iNum) ;
                        for (i = 2 ; i <= iSqrt ; i++)
                                   if (iNum % i == 0)
                                              break ;
             }
             while (i <= iSqrt) ;
             iLine = CheckBottom   (      pparams->hwnd,      ppa
pparams->cyChar,iLine) ;

             hdc = GetDC (pparams->hwnd) ;
```

```
            TextOut (     hdc, 0, iLine * pparams->cyChar, szBuffe
                                        wsprintf (szBuffer, TEXT ("%

            ReleaseDC (pparams->hwnd, hdc) ;
            iLine++ ;
        }
   _endthread () ;
}

LRESULT APIENTRY WndProc2 (HWND hwnd, UINT message, WPA
{
        static PARAMS params ;
        switch (message)
        {
        case   WM_CREATE:
                    params.hwnd = hwnd ;
                    params.cyChar = HIWORD (GetDialogBaseUnits
                    _beginthread (Thread2, 0, s) ;
                    return 0 ;
```

```
        case   WM_SIZE:
                    params.cyClient = HIWORD (lParam) ;
                    return 0 ;

        case   WM_DESTROY:
                    params.bKill = TRUE ;
                    return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}

// Window 3: Display increasing sequence of Fibonacci numbers
// -------------------------------------------------------

void Thread3 (PVOID pvoid)
{
        HDC                     hdc ;
        int             iNum = 0, iNext = 1, iLine = 0, iTemp ;
        PPARAMS         pparams ;
```

```
        TCHAR                   szBuffer[16] ;

        pparams = (PPARAMS) pvoid ;
        while (!pparams->bKill)
{
                if (iNum < 0)
                {
                        iNum  = 0 ;
                        iNext = 1 ;
            }
            iLine = CheckBottom ( pparams->hwnd,       pparam
pparams->cyChar, iLine) ;

            hdc = GetDC (pparams->hwnd) ;

            TextOut (hdc, 0, iLine * pparams->cyChar, szBuffer,
                                    wsprintf (szBuffer, TEXT ("%

          ReleaseDC (pparams->hwnd, hdc) ;
```

```
            iTemp  = iNum ;
            iNum   = iNext ;
            iNext += iTemp ;
            iLine++ ;
        }
        _endthread () ;
}

LRESULT APIENTRY WndProc3 (HWND hwnd, UINT message, WPA
{
        static PARAMS params ;
        switch (message)
        {
        case   WM_CREATE:
                    params.hwnd = hwnd ;
                    params.cyChar = HIWORD (GetDialogBaseUnits
                    _beginthread (Thread3, 0, s) ;
                    return 0 ;
```

```
        case   WM_SIZE:
                    params.cyClient = HIWORD (lParam) ;
                    return 0 ;

        case   WM_DESTROY:
                    params.bKill = TRUE ;
                    return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}

// ------------------------------------------------------------------------
// Window 4: Display circles of random radii
// ------------------------------------------------------------------------

void Thread4 (PVOID pvoid)
{
        HDC                     hdc ;
        int                     iDiameter ;
        PPARAMS             pparams ;
```

```
pparams = (PPARAMS) pvoid ;
while (!pparams->bKill)
{
      InvalidateRect (pparams->hwnd, NULL, TRUE) ;
      UpdateWindow (pparams->hwnd) ;

      iDiameter =   rand() % (max (1,
   min (pparams->cxClient, pparams->cyClient))) ;

      hdc = GetDC (pparams->hwnd) ;

      Ellipse (hdc, (pparams->cxClient - iDiameter) / 2,
                    (pparams->cyClient - iDiameter) / 2,
                    (pparams->cxClient + iDiameter) / 2,
              (pparams->cyClient + iDiameter) / 2) ;

      ReleaseDC (pparams->hwnd, hdc) ;
```

```
        }
   _endthread () ;
}

LRESULT APIENTRY WndProc4 (HWND hwnd, UINT message,WPA
{
        static PARAMS params ;
        switch (message)
        {
        case   WM_CREATE:
                    params.hwnd = hwnd ;
                    params.cyChar = HIWORD (GetDialogBaseUnits
                    _beginthread (Thread4, 0, s) ;
                    return 0 ;

        case   WM_SIZE:
                    params.cxClient = LOWORD (lParam) ;
                    params.cyClient = HIWORD (lParam) ;
                   return 0 ;
```

```
        case   WM_DESTROY:
                    params.bKill = TRUE ;
                    return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam)
}

// ------------------------------------------------------------------------
// Main window to create child windows
// ------------------------------------------------------------------------

LRESULT APIENTRY WndProc ( HWND hwnd, UINT message, WPA
{
        static HWND          hwndChild[4] ;
        static TCHAR * szChildClass[] = { TEXT ("Child1"), TEXT ("
      TEXT ("Child3"), TEXT ("Child4") } ;
        static WNDPROC ChildProc[] = { WndProc1, WndProc2, W
        HINSTANCE            hInstance ;
        int                          i, cxClient, cyClient ;
```

```
WNDCLASS                wndclass ;


switch (message)
{
case   WM_CREATE:
            hInstance = (HINSTANCE) GetWindowLong (hwn
            wndclass.style                              = CS
            wndclass.cbClsExtra                 = 0 ;
            wndclass.cbWndExtra                   = 0 ;
            wndclass.hInstance                  = hInstance
            wndclass.hIcon                      = NULL ;
            wndclass.hCursor                    = LoadCurso
            wndclass.hbrBackground              = (HBRUSH) (
            wndclass.lpszMenuName  = NULL ;

            for (i = 0 ; i < 4 ; i++)
            {
                        wndclass.lpfnWndProc        = ChildPr
```

```
                    wndclass.lpszClassName        = szChil

                    RegisterClass (&wndclass) ;

                    hwndChild[i] = CreateWindow (szChild
               WS_CHILDWINDOW | WS_BORDER | WS_VISIBLE
               0, 0, 0, 0,
                hwnd, (HMENU) i, hInstance, NULL) ;
               }

               return 0 ;

     case   WM_SIZE:
               cxClient = LOWORD (lParam) ;
               cyClient = HIWORD (lParam) ;

               for (i = 0 ; i < 4 ; i++)
                              MoveWindow (hwndChild[i], (i %
                                        (i > 1) * cyClient / 2,
```

```
                        cxClient / 2, cyClient / 2, TRUE) ;
                return 0 ;

        case   WM_CHAR:
                if (wParam == '\x1B')
                        DestroyWindow (hwnd) ;

                return 0 ;

        case   WM_DESTROY:
                PostQuitMessage (0) ;
                return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

MULTI2.CWinMainWndProcMULTI1.CWndProcWM_SIZE WndProcWindowsWM_TIMER

MULTI2WM_CREATE_beginthreadMULTI2Thread1Thread2

RNDRCTMT_beginthread32MULTI2

MULTI2PARAMSPPARAMSbKill

WndProc1PARAMSWM_CREATEhwndcyChar
_beginthreadThread1WM_SIZEWndProc1cyClient
WM_DESTROYbKillTRUEThread1_endthread
_endthread

Thread1PARAMSwhilebKillTRUEFALSEFALSEMULTI1.C
WM_TIMERTextOut

Windows 98MULTI2MULTI1MULTI1MULTI2
MULTI2

MULTI2MULTI2.CWndProc1Thread1

WndProc1MULTI2Thread1Windows 98Thread1
TRUETRUEWindows 98WndProc1WM_DESTR
Thread1

Windows 98

EnterCriticalSectionLeaveCriticalSection
EnterCriticalSectionLeaveCriticalSection

MULTI2WM_ERASEBKGNDWM_PAINTWindows
98

Windows 98GDIGDI

Sleep

WindowsWM_TIMER

SleepSleepSleeptickSleep0

SleepSCRAMBLESleep

SleepSCRAMBLESleep

CRITICAL_SECTION

```
CRITICAL_SECTION cs ;
```

CRITICAL_SECTIONWindows

```
InitializeCriticalSection (&cs) ;
```

cs

```
EnterCriticalSection (&cs) ;
```

EnterCriticalSection

```
LeaveCriticalSection (&cs) ;
```

EnterCriticalSection

```
DeleteCriticalSection (&cs) ;
```

cs1cs2

mutex
exclusion

1/101/101/10

BIGJOB1

logexpatantan1

BIGJOB120-4

20-4　　BIGJOB1

```
BIGJOB1.C
```

```
/*--------------------------------------------------------------------------
  BIGJOB1.C -- Multithreading Demo
                                                (c) Charles Petzold, 1998
--------------------------------------------------------------------------*/

#include <windows.h>
#include <math.h>
#include <process.h>

#define REP                                     1000000

#define STATUS_READY                    0
#define STATUS_WORKING                  1
#define STATUS_DONE                     2

#define WM_CALC_DONE                (WM_USER + 0)
#define WM_CALC_ABORTED             (WM_USER + 1)

typedef struct
{
        HWND hwnd ;
```

```
        BOOL bContinue ;
}
PARAMS, *PPARAMS ;
LRESULT APIENTRY WndProc (HWND, UINT, WPARAM, LPARAM) ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                szCmdLine, int iCmdShow)
{
        static TCHAR        szAppName[] = TEXT ("BigJob1") ;
        HWND                            hwnd ;
        MSG                             msg ;
        WNDCLASS                        wndclass ;
        wndclass.style                  = CS_HREDRAW | CS_VREDR
        wndclass.lpfnWndProc              = WndProc ;
        wndclass.cbClsExtra               = 0 ;
        wndclass.cbWndExtra               = 0 ;
        wndclass.hInstance                = hInstance ;
        wndclass.hIcon                  = LoadIcon (NULL, IDI_APPL
        wndclass.hCursor            = LoadCursor (NULL, IDC_ARRO
        wndclass.hbrBackground       = (HBRUSH) GetStockObjec
```

```
        wndclass.lpszMenuName         = NULL ;
        wndclass.lpszClassName        = szAppName ;

        if (!RegisterClass (&wndclass))
        {
    MessageBox (NULL, TEXT ("This program requires Windows
                    szAppName, MB_ICONERROR) ;
            return 0 ;
}

        hwnd = CreateWindow (szAppName, TEXT ("Multithreadin
                          WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;

        ShowWindow (hwnd, iCmdShow) ;
        UpdateWindow (hwnd) ;
```

```
        while (GetMessage (&msg, NULL, 0, 0))
        {
             TranslateMessage (&msg) ;
             DispatchMessage (&msg) ;
        }
        return msg.wParam ;
}

void Thread (PVOID pvoid)
{
        double              A = 1.0 ;
        INT                      i ;
        LONG                     lTime ;
        volatile            PPARAMS pparams ;

        pparams = (PPARAMS) pvoid ;
        lTime = GetCurrentTime () ;
        for (i = 0 ; i < REP && pparams->bContinue ; i++)
```

```
                        A = tan (atan (exp (log (sqrt (A * A))))) + 1
        if (i == REP)
        {
             lTime = GetCurrentTime () - lTime ;
             SendMessage (pparams->hwnd, WM_CALC_DONE, 0,
  }
       else
             SendMessage (pparams->hwnd, WM_CALC_ABORTED
        _endthread () ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
        static INT          iStatus ;
        static LONG         lTime ;
        static PARAMSparams ;
        static       TCHAR *szMessage[] = { TEXT ("Ready (left mo
              TEXT ("Working (right mouse button ends)"),
              TEXT ("%d repetitions in %ld msec") } ;
```

```
HDC                                 hdc ;
PAINTSTRUCT                         ps ;
RECT                                rect ;
TCHAR                               szBuffer[64] ;

switch (message)
{
case   WM_LBUTTONDOWN:
            if (iStatus == STATUS_WORKING)
            {
                        MessageBeep (0) ;
                        return 0 ;
            }

            iStatus = STATUS_WORKING ;

            params.hwnd = hwnd ;
            params.bContinue = TRUE ;
```

```
                _beginthread (Thread, 0, s) ;

                InvalidateRect (hwnd, NULL, TRUE) ;
                return 0 ;

        case   WM_RBUTTONDOWN:
                params.bContinue = FALSE ;
                return 0 ;

        case   WM_CALC_DONE:
                lTime = lParam ;
                iStatus = STATUS_DONE ;
                InvalidateRect (hwnd, NULL, TRUE) ;
                return 0 ;

        case   WM_CALC_ABORTED:
                iStatus = STATUS_READY ;
                InvalidateRect (hwnd, NULL, TRUE) ;
```

```
                return 0 ;

        case   WM_PAINT:
                hdc = BeginPaint (hwnd, &ps) ;

                GetClientRect (hwnd, &rect) ;

                wsprintf (szBuffer, szMessage[iStatus], REP, lTin
                DrawText (hdc, szBuffer, -1, &rect,
                DT_SINGLELINE | DT_CENTER | DT_VCENTER) ;

                EndPaint (hwnd, &ps) ;
                return 0 ;

        case   WM_DESTROY:
                PostQuitMessage (0) ;
                return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam)
```

```
}
```

BIGJOB11,000,000300MHzPentium

iStatusSTATUSWM_PAINTiStatus

PARAMShwndbContinue

iStatusSTATUS_WORKINGPARAMShwndbContinueTRUE

_beginthreadThreadGetCurrentTimeGetCurrentTimeWindowsfor 1,000,000bContinueFALSE

for1,000,000GetCurrentTimeSendMessageWM_USER_DONE lParamPARAMSbContinueFALSEWM_USER_ABORTED _endthread

PARAMSbContinueFALSE

Threadpparamsvolatilepparams->bContinueforvolatile

WM_USER_DONEWM_USER_DONEWM_USER_ABORTED InvalidateRectWM_PAINT

bContinueKillThread

WindowsSendMessage_endthread

BIGJOB1

```
hEvent = CreateEvent (&sa, fManual, fInitial, pszName) ;
```

SECURITY_ATTRIBUTESNULLfInitialTRUE
fInitialFALSEfManual

```
SetEvent (hEvent) ;
```

```
ResetEvent (hEvent) ;
```

```
WaitForSingleObject (hEvent, dwTimeOut) ;
```

INFINITE

CreateEventfManualFALSEWaitForSingleObjectResetEvent

20-5BIGJOB2.C

20-5　　BIGJOB2

```
BIGJOB2.C

/*---------------------------------------------------------------------

  BIGJOB2.C -- Multithreading Demo
```

```
                                        (c) Charles Petzold, 1998
----------------------------------------------------------------------------*/
#include <windows.h>
#include <math.h>
#include <process.h>

#define REP                                     1000000

#define STATUS_READY                    0
#define STATUS_WORKING                  1
#define STATUS_DONE                     2

#define WM_CALC_DONE                    (WM_USER + 0)
#define WM_CALC_ABORTED                 (WM_USER + 1)

typedef struct
{
        HWND                hwnd ;
        HANDLE          hEvent ;
        BOOL                bContinue ;
```

```
}
PARAMS, *PPARAMS ;
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                PSTR szCmdLine, int iCmdShow)
{
        static TCHAR szAppName[] = TEXT ("BigJob2") ;
        HWND                        hwnd ;
       MSG                         msg ;
   WNDCLASS                        wndclass ;

        wndclass.style                  = CS_HREDRAW | CS_VREDR
        wndclass.lpfnWndProc              = WndProc ;
        wndclass.cbClsExtra               = 0 ;
        wndclass.cbWndExtra               = 0 ;
        wndclass.hInstance         = hInstance ;
        wndclass.hIcon             = LoadIcon (NULL, IDI_APPLICAT
        wndclass.hCursor            = LoadCursor (NULL, IDC_ARRO
        wndclass.hbrBackground       = (HBRUSH) GetStockObjec
```

```
        wndclass.lpszMenuName         = NULL ;
        wndclass.lpszClassName        = szAppName ;

        if (!RegisterClass (&wndclass))
{
  MessageBox (NULL, TEXT ("This program requires Windows N
              szAppName, MB_ICONERROR) ;
             return 0 ;
}

        hwnd = CreateWindow (szAppName, TEXT ("Multithreadin
                             WS_OVERLAPPEDWINDOW,
                             CW_USEDEFAULT, CW_USEDEFAULT,
                             CW_USEDEFAULT, CW_USEDEFAULT,
                             NULL, NULL, hInstance, NULL) ;

        ShowWindow (hwnd, iCmdShow) ;
        UpdateWindow (hwnd) ;
```

```
        while (GetMessage (&msg, NULL, 0, 0))
        {
             TranslateMessage (&msg) ;
             DispatchMessage (&msg) ;
        }
        return msg.wParam ;
}

void Thread (PVOID pvoid)
{
        double              A = 1.0 ;
        INT                      i ;
        LONG                       lTime ;
        volatile            PPARAMS pparams ;

        pparams = (PPARAMS) pvoid ;
        while (TRUE)
        {
             WaitForSingleObject (pparams->hEvent, INFINITE) ;
```

```
            lTime = GetCurrentTime () ;
            for (i = 0 ; i < REP && pparams->bContinue ; i++)
                        A = tan (atan (exp (log (sqrt (A * A)))))
            if (i == REP)
            {
                        lTime = GetCurrentTime () - lTime ;
                        PostMessage (pparams->hwnd, WM_C
            }
            else
                        PostMessage (pparams->hwnd, WM_C
        }
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
        static HANDLE       hEvent ;
        static INT          iStatus ;
        static LONG         lTime ;
        static PARAMS params ;
```

```
static       TCHAR *szMessage[] = { TEXT ("Ready (left mo
     TEXT ("Working (right mouse button ends)"),
      TEXT ("%d repetitions in %ld msec") } ;
HDC                                  hdc ;
PAINTSTRUCT                         ps ;
RECT                                 rect ;
TCHAR                                 szBuffer[64] ;

switch (message)
{
case   WM_CREATE:
            hEvent = CreateEvent (NULL, FALSE, FALSE, NUL

            params.hwnd = hwnd ;
            params.hEvent = hEvent ;
            params.bContinue = FALSE ;

            _beginthread (Thread, 0, s) ;
```

```
                return 0 ;

case   WM_LBUTTONDOWN:
                if (iStatus == STATUS_WORKING)
                {
                        MessageBeep (0) ;
                        return 0 ;
                }
                iStatus = STATUS_WORKING ;
                params.bContinue = TRUE ;

                SetEvent (hEvent) ;

                InvalidateRect (hwnd, NULL, TRUE) ;
                return 0 ;

      case   WM_RBUTTONDOWN:
                params.bContinue = FALSE ;
```

```
                return 0 ;

        case   WM_CALC_DONE:
                lTime = lParam ;
                iStatus = STATUS_DONE ;
                InvalidateRect (hwnd, NULL, TRUE) ;
                return 0 ;

        case   WM_CALC_ABORTED:
                iStatus = STATUS_READY ;
                InvalidateRect (hwnd, NULL, TRUE) ;
                return 0 ;

case   WM_PAINT:
                hdc = BeginPaint (hwnd, &ps) ;

                GetClientRect (hwnd, &rect) ;
```

```
                wsprintf (    szBuffer, szMessage[iStatus], REP, lT
                 DrawText (    hdc, szBuffer, -1, &rect,
                                                DT_SINGLELINE | DT_CE


                 EndPaint (hwnd, &ps) ;
                return 0 ;


        case   WM_DESTROY:
                 PostQuitMessage (0) ;
                 return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam)
}
```

WM_CREATE

ThreadwhileWaitForSingleObjectPARAMSSetEvent
WaitForSingleObjectWaitForSingleObject

BIGJOB1

TLS

CstrtokCWindowsMicrosoftC

API

```
typedef struct
{
        int a ;
        int b ;
}
DATA, * PDATA ;
```

TlsAlloc

```
dwTlsIndex = TlsAlloc () ;
```

TlsSetValue

```
TlsSetValue (dwTlsIndex, GlobalAlloc (GPTR, sizeof (DATA)) ;
```

```
PDATA pdata ;
...
```

```
pdata = (PDATA) TlsGetValue (dwTlsIndex) ;
```

pdata->apdata->b

```
GlobalFree (TlsGetValue (dwTlsIndex)) ;
```

```
TlsFree (dwTlsIndex) ;
```

WindowsTlsAlloc0TlsSetValue88
IDGetCurrentThreadIdTlsSetValueTlsSetValueIDTlsFree

MicrosoftC__declspec

```
__declspec (thread) int iGlobal = 1 ;
```

```
__declspec (thread) static int iLocal = 2 ;
```

DLLMicrosoft WindowsWindowsWindows

WindowsDLL

Windows(.OBJ)(.LIB)(.RES)Windows.EXE

KERNEL32.DLLUSER32.DLLGDI32.DLLKEYBOARD.DRV
SYSTEM.DRVMOUSE.DRVWindows

WindowsWindows
Windows

.EXE.FON.DLL.DLLWindowsLoadLibrary
LoadLibraryEx

Windows.LIBLINK
ACCOUNT.DLL

3DGDI3.DLLDLLGDI3.DLL

.LIB.EXEMicrosoft

.LIB.EXEMicrosoftKERNEL32.LIB
USER32.LIBGDI32.LIBWindowsRectangleRectangleLINK
GDI32.DLL.EXEWindowsGDI32.DLL

Windows.EXEWindowsWindowsMS-DOS
PATHWindows

DLL

DLL

EDRLIB.DLLDLLEDReasy
EdrCenterTextEDRTEST.EXEEDRLIB.DLL

Visual C++ Visual C++
.EXE.DLLEDRTESTEDRTEST.EXEEDRLIB.DLL
EDRTEST

Visual C++FileNewWorkspacesLocation
Workspace NameEDRTESTEnter

Developer StudioEDRTESTEDRTEST.DSW

FileNewProjectsWin32
Dynamic-Link LibraryAdd To Current WorkspaceEDRTE
Project NameEDRLIBOKProject Namel
C++LocationEDRLIBEDRTESTLocationEDRLIB
EDRTESTOKDLLAn
Visual C++EDRLIB.DSPEDRLIB.MAKTools
Makefile

FileNewFilesC/C++
FileNewFilesC++

21-1 EDRLIB

```
EDRLIB.H

/*----------------------------------------------------------------------

  EDRLIB.H header file

----------------------------------------------------------------------*/
```

```
#ifdef      __cplusplus
#define     EXPORT extern "C" __declspec (dllexport)
#else
#define     EXPORT __declspec (dllexport)
#endif

EXPORT      BOOL CALLBACK EdrCenterTextA (HDC, PRECT, PCST
EXPORT      BOOL CALLBACK EdrCenterTextW (HDC, PRECT, PCW

#ifdef      UNICODE
#define     EdrCenterText EdrCenterTextW
#else
#define     EdrCenterText EdrCenterTextA
#endif
```

EDRLIB.C

```
/*-----------------------------------------------------------------------
  EDRLIB.C -- Easy Drawing Routine Library module
                                          (c) Charles Petzold, 1998
-----------------------------------------------------------------------*/
```

```
#include  windows.h>

#include "edrlib.h"

int WINAPI DllMain (HINSTANCE hInstance, DWORD fdwReason, P

{

        return TRUE ;

}

EXPORT BOOL CALLBACK EdrCenterTextA (     HDC hdc, PRECT pr

{

        int  iLength ;

        SIZE size ;

        iLength = lstrlenA (pString) ;

        GetTextExtentPoint32A (hdc, pString, iLength, &size) ;

        return TextOutA (hdc,(prc->right - prc->left - size.cx) / 2,

                (     prc->bottom - prc->top - size.cy) / 2,

                        pString, iLength) ;

}

EXPORT BOOL CALLBACK EdrCenterTextW (HDC hdc, PRECT prc,
```

```
{
        int  iLength ;
        SIZE size ;

        iLength = lstrlenW (pString) ;
        GetTextExtentPoint32W (hdc, pString, iLength, &size) ;
        return TextOutW (hdc, (      prc->right - prc->left - size.cx)
                        (      prc->bottom - prc->top - size.cy) / 2,
                                pString, iLength) ;
}
```

ReleaseDebugEDRLIB.DLLRELEASEDEBUGEDRLIB.LIB
EDRLIB.DLL

UNICODEUnicodeUnicodeDLLUnicodeUnicodeEDRLIB.C
EdrCenterTextAANSIEdrCenterTextWEdrCenterTextA
PCSTRconstEdrCenterTextWPCWSTRconstEdrCenterTextA
lstrlenAGetTextExtentPoint32ATextOutAEdrCenterTextWlstrlenW
GetTextExtentPoint32WTextOutWUNICODEEDRLIB.HEdrCenterText
EdrCenterTextWEdrCenterTextAWindows

EDRLIB.HDllMainDLLWinMaindeinitializationDllMainTRUE

EXPORTDLLexportedEDRLIB.LIB
EDRLIB.DLLEXPORT__declspecdllexportC++CC++name
manglingCC++DLL

DllMainDllMainDialogBoxhInstanceDllMain

fdwReasonWindowsDllMainWindowsprocess

fdwReasonDLL_PROCESS_ATTACHDllMain
DLL_PROCESS_ATTACHDLLDLL_PROCESS_ATTACHDllMain

DllMain00Windows

fdwReasonDLL_PROCESS_DETACHDLL32Windows

DLL_THREAD_ATTACHfdwReasonDllMainWindows
DLL_THREAD_DETACHfdwReasonDllMain
DLL_THREAD_ATTACHDLL_THREAD_DETACH

DLL_THREAD_DETACHDllMainPostMessage

EDRTESTEDRTESTEDRLIB.DLLVisual
NewNewProjectsWin32 Appli
WorkspaceEDRTESTLocationsEDRTESTOKAn
ProjectFinish

FileNewFilesC++ Source FileAd
EDRTESTEDRLIBEDRTEST.C21-2EdrCenterText

21-2 EDRTEST

```
EDRTEST.C

/*--------------------------------------------------------------------

  EDRTEST.C -- Program using EDRLIB dynamic-link library

                                        (c) Charles Petzold, 1998

--------------------------------------------------------------------*/
```

```
#include <windows.h>
#include "edrlib.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                        PSTR szCmdLine, int iCr
{
        static TCHAR szAppName[] = TEXT ("StrProg") ;
        HWND                        hwnd ;
        MSG                         msg ;
        WNDCLASS                wndclass ;

        wndclass.style                  = CS_HREDRAW | CS_VREDR
        wndclass.lpfnWndProc              = WndProc ;
        wndclass.cbClsExtra               = 0 ;
        wndclass.cbWndExtra               = 0 ;
        wndclass.hInstance                = hInstance ;
        wndclass.hIcon                  = LoadIcon (NULL, IDI_APPLI
        wndclass.hCursor                  = LoadCursor (NULL, IDC_A
```

```
        wndclass.hbrBackground        = (HBRUSH) GetStockObjec
        wndclass.lpszMenuName         = NULL ;
        wndclass.lpszClassName        = szAppName ;

        if (!RegisterClass (&wndclass))
        {
             MessageBox (  NULL, TEXT ("This program requires W
                                                  szAppName, MB_IC
             return 0 ;
   }

        hwnd = CreateWindow (szAppName, TEXT ("DLL Demons
                   WS_OVERLAPPEDWINDOW,
                   CW_USEDEFAULT, CW_USEDEFAULT,
                   CW_USEDEFAULT, CW_USEDEFAULT,
                   NULL, NULL, hInstance, NULL) ;

        ShowWindow (hwnd, iCmdShow) ;
        UpdateWindow (hwnd) ;
```

```
        while (GetMessage (&msg, NULL, 0, 0))
        {
             TranslateMessage (&msg) ;
             DispatchMessage (&msg) ;
        }
       return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
        HDC                hdc ;
        PAINTSTRUCT        ps ;
        RECT               rect ;

        switch (message)
        {
        case   WM_PAINT:
                   hdc = BeginPaint (hwnd, &ps) ;
```

```
                    GetClientRect (hwnd, &rect) ;

                    EdrCenterText (hdc, &rect,
                TEXT ("This string was displayed by a DLL")) ;

                    EndPaint (hwnd, &ps) ;
                    return 0 ;

          case   WM_DESTROY:
                    PostQuitMessage (0) ;
                    return 0 ;
          }
    return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

EdrCenterTextEDRTEST.CEDRLIB.HWM_PAINT

ProjectSelect Activ
EDRTESTProjectDependenciesSelect
To ModifyEDRTESTDependent On The Following Proj
EDRLIBEDRTESTEDRLIBEDRTESTEDRTEST

EDRLIB

ProjectSettingsGeneralEDRLIBEDRTESTWin32
ReleaseIntermediate FilesOutput FilesRELEASEV
DebugDEBUGEDRLIB.DLLEDRTEST.EXEDLL

Project SettingEDRTESTC/C++Preprocessor
DefinitionsUNICODEDebug

DebugReleaseEDRTEST.EXEVisual
RELEASEDEBUGEDRLIB.LIBEDRLIB.DLLDeveloper
EDRTEST

EDRTEST.EXEEdrCenterTextEDRLIB.DLLEdrCenterText
EDRTEST.EXEEDRLIB.DLL

EDRTEST.EXEWindowsWindowsWindowsEDRLIBWindows
EDRLIB.DLLEDRLIBEDRTESTEdrCenterTextEDRLIB

EDRTEST.CEDRLIB.HWINDOWS.HEDRLIB.LIBWindows
USER32.LIBEDLIB.DLLUSER32.DLLWindows

DLLWindowsDLLDLLDLLDLLDLLWindows

DLLDLL

DLL

WindowsDLL

STRPROGstring programSTR
STRLIBSTRPROGSTRLIBSTRPROGcallback

STRLIB256STRLIBSTRLIBSTRPROGSTRLIB
STRPROGEnterDeleteSTRPROGSTRLIB

STRLIBSTRLIB

```
EXPORT BOOL CALLBACK AddString (pStringIn)
```

pStringInAddStringSTRLIBAddStringTRUE0
FALSE00256FALSE

STRLIBSTRLIB

```
EXPORT BOOL CALLBACK DeleteString (pStringIn)
```

pStringInDeleteStringTRUE0FALSE0
FALSE0

STRLIBcallbackSTRLIB

```
EXPORT int CALLBACK GetStrings (pfnGetStrCallBack, pParam)
```

callback

```
EXPORT BOOL CALLBACK GetStrCallBack (PSTR pString, PVOID p
```

GetStringspfnGetStrCallBackcallbackcallbackFALSE0GetStrings
GetStrCallBackGetStringscallbackpParam

UnicodeSTRLIBUnicodeUnicodeEDRLIBAW
STRLIBUnicodeUnicodeSTRLIBAddStringADeleteStringA
GetStringsAUnicodeUnicode

STRPROGSTRLIBSTRPROGEDRTEST21-3STRLIB.DLL

21-3 STRLI

STRLIB.H

```
/*--------------------------------------------------------------------------
  STRLIB.H header file
--------------------------------------------------------------------------*/

#ifdef __cplusplus
#define EXPORT extern "C" __declspec (dllexport)
#else
#define EXPORT __declspec (dllexport)
#endif

          // The maximum number of strings STRLIB will store and t
#define     MAX_STRINGS 256
#define     MAX_LENGTH  63

          // The callback function type definition uses generic string

typedef BOOL (CALLBACK * GETSTRCB) (PCTSTR, PVOID) ;

          // Each function has ANSI and Unicode versions

EXPORT      BOOL CALLBACK AddStringA (PCSTR) ;
```

```
EXPORT      BOOL CALLBACK AddStringW (PCWSTR) ;

EXPORT      BOOL CALLBACK DeleteStringA (PCSTR) ;
EXPORT      BOOL CALLBACK DeleteStringW (PCWSTR) ;

EXPORT      int CALLBACK GetStringsA (GETSTRCB, PVOID) ;
EXPORT      int CALLBACK GetStringsW (GETSTRCB, PVOID) ;

          // Use the correct version depending on the UNICODE ide

#ifdef      UNICODE
#define     AddString    AddStringW
#define     DeleteString         DeleteStringW
#define     GetStrings           GetStringsW
#else
#define     AddString    AddStringA
#define     DeleteString         DeleteStringA
#define     GetStrings           GetStringsA
#endif
```

STRLIB.C

```
/*------------------------------------------------------------------------
  STRLIB.C -    Library module for STRPROG program
                                        (c) Charles Petzold, 1998
------------------------------------------------------------------------*/

#include <windows.h>
#include <wchar.h>                              // for wide-character str
#include "strlib.h"

   // shared memory section (requires /SECTION:shared,RWS in li
#pragma          data_seg ("shared")
int                 iTotal = 0 ;
WCHAR                    szStrings [MAX_STRINGS][MAX_LENGTH + 1
#pragma          data_seg ()

#pragma          comment(linker,"/SECTION:shared,RWS")

int WINAPI DllMain (HINSTANCE hInstance, DWORD fdwReason, P
{
   return TRUE ;
}
```

```
EXPORT BOOL CALLBACK AddStringA (PCSTR pStringIn)
{
          BOOL          bReturn ;
         int           iLength ;
          PWSTR         pWideStr ;

    // Convert string to Unicode and call AddStringW
          iLength = MultiByteToWideChar (CP_ACP, 0, pStringIn, -1,
          pWideStr = malloc (iLength) ;
         MultiByteToWideChar (CP_ACP, 0, pStringIn, -1, pWideStr,
          bReturn = AddStringW (pWideStr) ;
          free (pWideStr) ;

          return bReturn ;
}

EXPORT BOOL CALLBACK AddStringW (PCWSTR pStringIn)
{
          PWSTR         pString ;
```

```
int          i, iLength ;


if (iTotal == MAX_STRINGS - 1)
             return FALSE ;
if ((iLength = wcslen (pStringIn)) == 0)
             return FALSE ;
      // Allocate memory for storing string, copy it, convert
pString = malloc (sizeof (WCHAR) * (1 + iLength)) ;
wcscpy (pString, pStringIn) ;
_wcsupr (pString) ;

      //  Alphabetize the strings
for (i = iTotal ; i > 0 ; i-)
{
      if (wcscmp (pString, szStrings[i - 1]) >= 0)
                        break ;
      wcscpy (szStrings[i], szStrings[i - 1]) ;
}
wcscpy (szStrings[i], pString) ;
```

```
        iTotal++ ;

        free (pString) ;
        return TRUE ;
}

EXPORT BOOL CALLBACK DeleteStringA (PCSTR pStringIn)
{
        BOOL        bReturn ;
        int         iLength ;
        PWSTR       pWideStr ;

                    // Convert string to Unicode and call DeleteStrin

        iLength = MultiByteToWideChar (CP_ACP, 0, pStringIn, -1,
        pWideStr = malloc (iLength) ;
        MultiByteToWideChar (CP_ACP, 0, pStringIn, -1, pWideStr,
        bReturn = DeleteStringW (pWideStr) ;
        free (pWideStr) ;

        return bReturn ;
```

```
}

EXPORT BOOL CALLBACK DeleteStringW (PCWSTR pStringIn)
{
        int i, j ;
        if (0 == wcslen (pStringIn))
                return FALSE ;
        for (i = 0 ; i < iTotal ; i++)
 {
             if (_wcsicmp (szStrings[i], pStringIn) == 0)
                                break ;
 }
             // If given string not in list, return without taking actio
        if (i == iTotal)
                return FALSE ;
             // Else adjust list downward
        for (j = i ; j < iTotal ; j++)
                wcscpy (szStrings[j], szStrings[j + 1]) ;
        szStrings[iTotal-][0] = '\0' ;
```

```
        return TRUE ;
}

EXPORT int CALLBACK GetStringsA (GETSTRCB pfnGetStrCallBac
{
        BOOL        bReturn ;
        int         i, iLength ;
        PSTR        pAnsiStr ;

   for (i = 0 ; i < iTotal ; i++)
        {
                        // Convert string from Unicode
iLength = WideCharToMultiByte (    CP_ACP, 0, szStrings[i], -1, N
                pAnsiStr = malloc (iLength) ;
       WideCharToMultiByte (    CP_ACP, 0, szStrings[i], -1, pAnsiS

                                // Call callback function

                bReturn = pfnGetStrCallBack (pAnsiStr, pParam) ;

                if (bReturn == FALSE)
```

```
                                    return i + 1 ;

                free (pAnsiStr) ;
        }
        return iTotal ;
}

EXPORT int CALLBACK GetStringsW (GETSTRCB pfnGetStrCallBac
{
        BOOL        bReturn ;
        int         i ;

        for (i = 0 ; i < iTotal ; i++)
        {
                bReturn = pfnGetStrCallBack (szStrings[i], pParam) ;
                if (bReturn == FALSE)
                                return i + 1 ;
        }
        return iTotal ;
```

```
}
```

DllMainSTRLIBEXPORTLINKSTRLIB.LIB

STRPROG

STRPROG21-4(EnterDelete)STRPROGAddStringDeleteString GetStringsGetStrCallBack

21-4　STRPROG

```
STRPROG.C

/*------------------------------------------------------------------------

  STRPROG.C -   Program using STRLIB dynamic-link library

                                        (c) Charles Petzold, 1998

------------------------------------------------------------------------*/

#include <windows.h>

#include "strlib.h"

#include "resource.h"


typedef struct

{

        HDC    hdc ;

        int    xText ;
```

```
        int    yText ;
        int    xStart ;
        int    yStart ;
        int    xIncr ;
        int    yIncr ;
        int    xMax ;
        int    yMax ;
}
CBPARAM ;
LRESULT    CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM
TCHAR            szAppName [] = TEXT ("StrProg") ;
TCHAR szString [MAX_LENGTH + 1] ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                                PSTR szCmdLine, int iCm
{
       HWND                    hwnd ;
        MSG                msg ;
        WNDCLASS            wndclass ;
```

```
     wndclass.style                    = CS_HREDRAW | CS_V
     wndclass.lpfnWndProc                = WndProc ;
     wndclass.cbClsExtra                 = 0 ;
     wndclass.cbWndExtra                 = 0 ;
     wndclass.hInstance                  = hInstance ;
     wndclass.hIcon                    = LoadIcon (NULL, IDI_
     wndclass.hCursor                = LoadCursor (NULL, IDC_AI
     wndclass.hbrBackground          = (HBRUSH) GetStockO
     wndclass.lpszMenuName            = szAppName ;
     wndclass.lpszClassName          = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (  NULL, TEXT ("This program requires W
                                          szAppName, MB_IC
          return 0 ;
}
```

```
        hwnd = CreateWindow (szAppName, TEXT ("DLL Demons
                             WS_OVERLAPPEDWINDOW,
                             CW_USEDEFAULT, CW_USEDEFAULT,
                            CW_USEDEFAULT, CW_USEDEFAULT,
                             NULL, NULL, hInstance, NULL) ;

        ShowWindow (hwnd, iCmdShow) ;
        UpdateWindow (hwnd) ;

        while (GetMessage (&msg, NULL, 0, 0))
        {
             TranslateMessage (&msg) ;
             DispatchMessage (&msg) ;
        }
        return msg.wParam ;
}

BOOL CALLBACK DlgProc (HWND hDlg, UINT message, WPARAM
{
```

```
        switch (message)
        {
        case   WM_INITDIALOG:
                   SendDlgItemMessage (hDlg, IDC_STRING, EM_LI
                   return TRUE ;

        case   WM_COMMAND:
                   switch (wParam)
                   {
                   case   IDOK:
                                  GetDlgItemText (hDlg, IDC_STRIN
                                EndDialog (hDlg, TRUE) ;
                                 return TRUE ;

                   case   IDCANCEL:
                                  EndDialog (hDlg, FALSE) ;
                                  return TRUE ;
                  }
        }
```

```
        return FALSE ;
}

BOOL CALLBACK GetStrCallBack (PTSTR pString, CBPARAM * pcb
{
        TextOut (     pcbp->hdc, pcbp->xText, pcbp->yText,
                                pString, lstrlen (pString)) ;

        if ((pcbp->yText += pcbp->yIncr) > pcbp->yMax)
        {
                pcbp->yText = pcbp->yStart ;
                if ((pcbp->xText += pcbp->xIncr) > pcbp->xMa
                                return FALSE ;
        }
        return TRUE ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
```

```
static HINSTANCE            hInst ;
static int                        cxChar, cyChar, cxClient, c
static UINT                  iDataChangeMsg ;
CBPARAM                              cbparam ;
HDC                                    hdc ;
PAINTSTRUCT                        ps ;
TEXTMETRIC                        tm ;

switch (message)
{
case   WM_CREATE:
            hInst        = ((LPCREATESTRUCT) lParam)->hIns
           hdc   = GetDC (hwnd) ;
            GetTextMetrics (hdc, &tm) ;
            cxChar = (int) tm.tmAveCharWidth ;
           cyChar = (int) (tm.tmHeight + tm.tmExternalLea
            ReleaseDC (hwnd, hdc) ;

                        // Register message for notifying instan
```

```
iDataChangeMsg = RegisterWindowMessage (TEXT ("StrProgD
            return 0 ;
      case   WM_COMMAND:
                  switch (wParam)
                  {
                  case   IDM_ENTER:
                                    if (DialogBox (hInst, TEXT ("Enter
                                    {
            if (AddString (szString))
            PostMessage (HWND_BROADCAST, iDataChangeMsg,
            else
                MessageBeep (0) ;
                 }
            break ;

                  case   IDM_DELETE:
                           if (DialogBox (hInst, TEXT ("DeleteDlg"), hv
                                    {
```

```
            if (DeleteString (szString))
            PostMessage (HWND_BROADCAST, iDataChangeMsg
           else
           MessageBeep (0) ;
                                  }
                          break ;
               }
               return 0 ;

     case   WM_SIZE:
               cxClient = (int) LOWORD (lParam) ;
               cyClient = (int) HIWORD (lParam) ;
               return 0 ;

     case   WM_PAINT:
               hdc = BeginPaint (hwnd, &ps) ;

               cbparam.hdc          = hdc ;
               cbparam.xText= cbparam.xStart = cxChar ;
```

```
                cbparam.yText= cbparam.yStart = cyChar ;
                cbparam.xIncr= cxChar * MAX_LENGTH ;
                cbparam.yIncr= cyChar ;
                cbparam.xMax = cbparam.xIncr * (1 + cxClient
                cbparam.yMax  = cyChar * (cyClient / cyChar - 1

                GetStrings ((GETSTRCB) GetStrCallBack, (PVOID

                EndPaint (hwnd, &ps) ;
                return 0 ;

        case   WM_DESTROY:
                PostQuitMessage (0) ;
                return 0 ;

        default:
             if (message == iDataChangeMsg)
                              InvalidateRect (hwnd, NULL, TRUE) ;
             break ;
```

```
        }
        return DefWindowProc (hwnd, message, wParam, lParam)
}
```

STRPROG.RC

```
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
/////////////////////////////////////////////////////////////////////////////
// Dialog
ENTERDLG DIALOG DISCARDABLE  20, 20, 186, 47
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSM
CAPTION "Enter"
FONT 8, "MS Sans Serif"
BEGIN
    LTEXT                   "&Enter:",IDC_STATIC,7,7,2
    EDITTEXT                IDC_STRING,31,7,148,12,ES_A
    DEFPUSHBUTTON           "OK",IDOK,32,26,50,14
    PUSHBUTTON              "Cancel",IDCANCEL,104,26,50,1
```

```
END
DELETEDLG DIALOG DISCARDABLE  20, 20, 186, 47
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSM
CAPTION "Delete"
FONT 8, "MS Sans Serif"
BEGIN
   LTEXT                                  "&Delete:",IDC_STATIC,7,7
   EDITTEXT                             IDC_STRING,31,7,148,12,ES_A
   DEFPUSHBUTTON              "OK",IDOK,32,26,50,14
   PUSHBUTTON                    "Cancel",IDCANCEL,104,26,50,1
END

//////////////////////////////////////////////////////////////////////////
// Menu
STRPROG MENU DISCARDABLE
BEGIN
    MENUITEM "&Enter!",                                    IDM_ENTE
   MENUITEM "&Delete!",                                       IDM_DE
END
```

```
RESOURCE.H

// Microsoft Developer Studio generated   include file.

// Used by StrProg.rc

#define          IDC_STRING                                    10

#define          IDM_ENTER                                      40

#define          IDM_DELETE                                    40

#define          IDC_STATIC                                      -1
```

STRPROG.CSTRLIB.HSTRPROGSTRLIB

STRPROGSTRLIBSTRPROG

STRPROG

WindowsWin32STRPROGSTRLIBSTRPROG

STRLIBSTRLIB

```
#pragma    data_seg ("shared")

int              iTotal = 0 ;

WCHAR            szStrings [MAX_STRINGS][MAX_LENGTH + 1] =

#pragma    data_seg ()
```

#pragmashared#pragmashared#pragmashared

sharedProject SettingsLinkSTRLIBProject
ReleaseDebug

```
/SECTION:shared,RWS
```

RWSDLLSTRLIB.C

```
#pragma comment(linker,"/SECTION:shared,RWS")
```

iTotalszStringsSTRLIBMAX_STRINGS256MAX_LENGTH63
32,772iTotal4256128

/Platform
Communication/File Mapping

DLL

GetMessagePeekMessageWindows

.EXE

CreateWindow

DialogBoxDialogBoxhwndParentNULL

WindowsRectangle

```
Rectangle (hdc, xLeft, yTop, xRight, yBottom) ;
```

GDI32.LIBRectangle

RectangletypedefRectangle

```
typedef BOOL (WINAPI * PFNRECT) (HDC, int, int, int, int) ;
```

```
HANDLE      hLibrary ;

PFNRECT     pfnRectangle ;
```

hLibrarylpfnRectangleRectangle

```
hLibrary = LoadLibrary (TEXT ("GDI32.DLL"))

pfnRectangle = (PFNPRECT) GetProcAddress (hLibrary, TEXT ("R
```

LoadLibraryNULL

```
pfnRectangle (hdc, xLeft, yTop, xRight, yBottom) ;

FreeLibrary (hLibrary) ;
```

Rectangle

LoadLibraryFreeLibraryWindowsLoadLibraryWindows
FreeLibraryWindows

WindowsDLLDLL

WindowsLoadBitmapWindows

21-59BITLIB.DLLBITLIB.RCBITLIB.DLL9BITMAP1.BMP
BITMAP2.BMPVisual C++ID1

21-5 BITLIB

```
```

BITLIB.C

```
/*-------------------------------------------------------------
  BITLIB.C --   Code entry point for BITLIB dynamic-link library
                                                (c) Charles Petzold,  1998
-------------------------------------------------------------*/
#include <windows.h>
int WINAPI DllMain (HINSTANCE hInstance, DWORD fdwReason, P
{
          return TRUE ;
}
```

BITLIB.RC

```
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
//////////////////////////////////////////////////////////////////////////////
// Bitmap
1               BITMAP  DISCARDABLE     "bitmap1.bmp"
2               BITMAP  DISCARDABLE     "bitmap2.bmp"
```

```
3               BITMAP  DISCARDABLE     "bitmap3.bmp"
4               BITMAP  DISCARDABLE     "bitmap4.bmp"
5               BITMAP  DISCARDABLE     "bitmap5.bmp"
6               BITMAP  DISCARDABLE     "bitmap6.bmp"
7               BITMAP  DISCARDABLE     "bitmap7.bmp"
8               BITMAP  DISCARDABLE     "bitmap8.bmp"
9               BITMAP  DISCARDABLE     "bitmap9.bmp"
```

SHOWBITBITLIBSHOWBIT21-6SHOWBITBITLIB
SHOWBITBITLIB.LIBBITLIBBITLIB.LIBBITLIB
SHOWBITActive Project

SHOWBIT.CBITLIB

21-6 SHOWBIT

SHOWBIT.C

```
/*------------------------------------------------------------------------
  SHOWBIT.C -- Shows bitmaps in BITLIB dynamic-link library
                                               (c) Charles Petzold, 1998
------------------------------------------------------------------------*/
#include <windows.h>
```

```
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

TCHAR szAppName [] = TEXT ("ShowBit") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                          PSTR szCmdLine, int iCr
{
     HWND                hwnd ;
     MSG                 msg ;
     WNDCLASS            wndclass ;

     wndclass.style                = CS_HREDRAW | CS_V
     wndclass.lpfnWndProc            = WndProc ;
     wndclass.cbClsExtra             = 0 ;
     wndclass.cbWndExtra             = 0 ;
     wndclass.hInstance            = hInstance ;
     wndclass.hIcon                  = LoadIcon (NULL, IDI_
     wndclass.hCursor                = LoadCursor (NULL,
     wndclass.hbrBackground        = (HBRUSH) GetStockO
     wndclass.lpszMenuName  = NULL ;
```

```
     wndclass.lpszClassName             = szAppName ;


      if (!RegisterClass (&wndclass))
{
           MessageBox (  NULL, TEXT ("This program requires W
                                                szAppName, MB_IC
           return 0 ;
}

hwnd = CreateWindow (szAppName,
      TEXT ("Show Bitmaps from BITLIB (Press Key)"),
          WS_OVERLAPPEDWINDOW,
          CW_USEDEFAULT, CW_USEDEFAULT,
         CW_USEDEFAULT, CW_USEDEFAULT,
          NULL, NULL, hInstance, NULL) ;

if (!hwnd)
                  return 0 ;
      ShowWindow (hwnd, iCmdShow) ;
      UpdateWindow (hwnd) ;
```

```
  while (GetMessage (&msg, NULL, 0, 0))
        {
       TranslateMessage (&msg) ;
       DispatchMessage (&msg) ;
        }
        return msg.wParam ;
}

void DrawBitmap (HDC hdc, int xStart, int yStart, HBITMAP hBitm
{
        BITMAP       bm ;
        HDC                hMemDC ;
        POINT              pt ;

        hMemDC = CreateCompatibleDC (hdc) ;
        SelectObject (hMemDC, hBitmap) ;
        GetObject (hBitmap, sizeof (BITMAP), &bm) ;
        pt.x = bm.bmWidth ;
```

```
        pt.y = bm.bmHeight ;

   BitBlt (hdc, xStart, yStart, pt.x, pt.y, hMemDC, 0, 0, SRCCOPY)
        DeleteDC (hMemDC) ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
        static HINSTANCE            hLibrary ;
        static int                        iCurrent = 1 ;
        HBITMAP                                hBitmap ;
        HDC                                       hdc ;
        PAINTSTRUCT                          ps ;

        switch (message)
        {
        case   WM_CREATE:
                    if ((hLibrary = LoadLibrary (TEXT ("BITLIB.DLL"))
                    {
       MessageBox (    hwnd, TEXT ("Can't load BITLIB.DLL."),
```

```
            szAppName, 0) ;
        return -1 ;
                }
            return 0 ;

        case   WM_CHAR:
                if (hLibrary)
                {
                        iCurrent ++ ;
                        InvalidateRect (hwnd, NULL, TRUE) ;
                }
                return 0 ;
        case   WM_PAINT:
                hdc = BeginPaint (hwnd, &ps) ;

                if (hLibrary)
                {
                hBitmap = LoadBitmap (hLibrary, MAKEINTRESO
```

```
                    if (!hBitmap)
                    {
                        iCurrent = 1 ;
                        hBitmap = LoadBitmap (hLibrary,
                    MAKEINTRESOURCE (iCurrent)) ;
                    }
                    if (hBitmap)
                        {
                            DrawBitmap (hdc, 0, 0, hBitm
                        DeleteObject (hBitmap) ;
    }
            }
            EndPaint (hwnd, &ps) ;
            return 0 ;

case   WM_DESTROY:
          if (hLibrary)
                        FreeLibrary (hLibrary) ;
```

```
                    PostQuitMessage (0) ;
                    return 0 ;
          }
          return DefWindowProc (hwnd, message, wParam, lParam)
}
```

WM_CREATESHOWBITBITLIB.DLL

```
if ((hLibrary = LoadLibrary (TEXT ("BITLIB.DLL"))) == NULL)
```

BITLIB.DLLSHOWBIT.EXEWindowsLoadLibraryNULLSHOWBIT
WM_CREATE-1WinMainCreateWindowNULL

SHOWBITLoadBitmap

```
hBitmap = LoadBitmap (hLibrary, MAKEINTRESOURCE (iCurrent)
```

iCurrent

WM_DESTROYSHOWBIT

```
FreeLibrary (hLibrary) ;
```

SHOWBITBITLIB.DLL0metafilemetafile

Microsoft Windows1991Microsoft WindowsMultimedia Extensions to Microsoft Windows1992Windows 3.1APICD-ROM 90PC Windows

WINDOWS

WindowsAPI

.WAVPC

MIDIMIDIMusical Instrument MIDI

PCCD-ROMCDCDMIDICD

WindowsAVI.AVIaudio-video ActiveMovieQuickTimeMPEGPC

PCPioneerSony VISCAPCvideo

API

WindowsAPI

/Platform SDK/Graphi Reference/Multimedia Functions

waveInwaveOutmidiOutMIDIAPImidiInmidiStream

time1MIDI

mciMCIMedia

MCIMCIWindowsCMCIWindows
MCIMCIMCICDcdaudiowaveaudioMIDIsequencer
videodiscvcroverlaydatdigital audio tape
digitalvideoMCICD.WAV

DirectX API

MessageBeepPlaySound MessageBeep
.WAVPlaySound

TESTMCIMCI

WindowsMCITESTCMCIC22-1TESTMCI
MCITEST

22-1 TESTMCI

```
TESTMCI.C

/*------------------------------------------------------------------------
  TESTMCI.C -- MCI Command String Tester
                                                (c) Charles Petzold, 199
------------------------------------------------------------------------*/

#include <windows.h>
#include "resource.h"

#define ID_TIMER    1

BOOL CALLBACK DlgProc (HWND, UINT, WPARAM, LPARAM) ;
```

```
TCHAR szAppName [] = TEXT ("TestMci") ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                                PSTR szCmdLine, int iCr
{
        if (-1 == DialogBox (hInstance, szAppName, NULL, DlgPro
        {
             MessageBox (  NULL, TEXT ("This program requires W
                                                szAppName, MB_IC
        }
        return 0 ;
}

BOOL CALLBACK DlgProc (    HWND hwnd, UINT message, WPARA
{
        static HWND   hwndEdit ;
        int        iCharBeg, iCharEnd, iLineBeg, iLineEnd, iChar, iLi
        MCIERROR                  error ;
        RECT                      rect ;
        TCHAR                     szCommand [1024], szReturn [102
```

```
                    szError [1024], szBuffer [32] ;

    switch (message)
    {
    case   WM_INITDIALOG:
                        // Center the window on screen

            GetWindowRect (hwnd, &rect) ;
            SetWindowPos (hwnd, NULL,
            (GetSystemMetrics (SM_CXSCREEN) - rect.right
            (GetSystemMetrics (SM_CYSCREEN) - rect.botto
            0, 0, SWP_NOZORDER | SWP_NOSIZE) ;

            hwndEdit = GetDlgItem (hwnd, IDC_MAIN_EDIT)
            SetFocus (hwndEdit) ;
            return FALSE ;

    case   WM_COMMAND:
            switch (LOWORD (wParam))
            {
```

```
            case   IDOK:
             // Find the line numbers corresponding to the selecti

          SendMessage (hwndEdit, EM_GETSEL, (WPARAM) &iCha
                   (LPARAM) &iCharEnd) ;

            iLineBeg = SendMessage (hwndEdit, EM_LINEFROMC
            iLineEnd = SendMessage (hwndEdit, EM_LINEFROMC

             // Loop through all the lines

           for (iLine = iLineBeg ; iLine <= iLineEnd ; iLine++)
                   {
              // Get the line and terminate it; ignore if blank

             * (WORD *) szCommand = sizeof (szCommand) / siz

                              iLength = SendMessage (hwndEdit, EM
                                   (LPARAM) szCommand) ;
             szCommand [iLength] = '\0' ;

                                            if (iLength == 0)
```

```
                                        continue ;

     // Send the MCI command

     error =mciSendString (szCommand, szReturn,

     sizeof (szReturn) / sizeof (TCHAR), hwnd) ;

               // Set the Return String field

    SetDlgItemText (hwnd, IDC_RETURN_STRING, szReturn) ;

              // Set the Error String field (even if no error)

    mciGetErrorString (error, szError, sizeof (szError) / sizeof (

           SetDlgItemText (hwnd, IDC_ERROR_STRING, szError)

                  }

                  // Send the caret to the end of the last selected

                  iChar  = SendMessage (hwndEdit, EM_LINEINDE

                             iChar += SendMessage (hwndEdit, EM

                             SendMessage (hwndEdit, EM_SETSEL,
```

```
                    // Insert a carriage return/line feed com

          SendMessage (hwndEdit, EM_REPLACESEL, FALS
              (LPARAM) TEXT ("\r\n")) ;
                    SetFocus (hwndEdit) ;
                    return TRUE ;

          case   IDCANCEL:
                         EndDialog (hwnd, 0) ;
                         return TRUE ;
          case   IDC_MAIN_EDIT:
                         if (HIWORD (wParam) == EN_ERR
                        {
 MessageBox (hwnd, TEXT ("Error control out of space.
            szAppName, MB_OK | MB_ICONINFORMATION)
return TRUE ;
  }
  break ;
          }
```

```
                break ;

        case    MM_MCINOTIFY:
                EnableWindow (GetDlgItem (hwnd, IDC_NOTIFY_

                wsprintf (szBuffer, TEXT ("Device ID = %i"), lPar

                SetDlgItemText (hwnd, IDC_NOTIFY_ID, szBuffer

                EnableWindow (GetDlgItem (hwnd, IDC_NOTIFY_

                EnableWindow (GetDlgItem (hwnd, IDC_NOTIFY_
                        wParam & MCI_NOTIFY_SUCCESSFUL) ;

                EnableWindow (GetDlgItem (hwnd, IDC_NOTIFY_
                        wParam & MCI_NOTIFY_SUPERSEDED) ;

                EnableWindow (GetDlgItem (hwnd, IDC_NOTIFY_
                        wParam & MCI_NOTIFY_ABORTED) ;

                EnableWindow (GetDlgItem (hwnd, IDC_NOTIFY_
                        wParam & MCI_NOTIFY_FAILURE) ;

                SetTimer (hwnd, ID_TIMER, 5000, NULL) ;
```

```
        return TRUE ;

    case   WM_TIMER:
            KillTimer (hwnd, ID_TIMER) ;

            EnableWindow (GetDlgItem (hwnd, IDC_NOTIFY_
            EnableWindow (GetDlgItem (hwnd, IDC_NOTIFY_
           EnableWindow (GetDlgItem (hwnd, IDC_NOTIFY_
           EnableWindow (GetDlgItem (hwnd, IDC_NOTIFY_
            EnableWindow (GetDlgItem (hwnd, IDC_NOTIFY_
           EnableWindow (GetDlgItem (hwnd, IDC_NOTIFY_
            return TRUE ;

    case   WM_SYSCOMMAND:
            switch (LOWORD (wParam))
            {
            case   SC_CLOSE:
                        EndDialog (hwnd, 0) ;
                        return TRUE ;
            }
```

```
                    break ;
   }
        return FALSE ;
}
```

TESTMCI.RC

```
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
//////////////////////////////////////////////////////////////////////////////
// Dialog
TESTMCI     DIALOG DISCARDABLE  0, 0, 270, 276
STYLE             WS_MINIMIZEBOX | WS_VISIBLE | WS_CAPTION | W
CAPTION     "MCI Tester"
FONT 8,     "MS Sans Serif"
BEGIN
   EDITTEXT                   IDC_MAIN_EDIT,8,8,254,100,ES_MULTIL
                                    WS_VSCROLL
   LTEXT                          "Return String:",IDC_STATIC,8,114,6
```

```
  EDITTEXT                IDC_RETURN_STRING,8,126,120,50,ES_
                                  ES_AUTOVSCROLL | ES_READONL
  LTEXT                     "Error String:",IDC_STATIC,142,114,6
  EDITTEXT                IDC_ERROR_STRING,142,126,120,50,ES
                                  ES_AUTOVSCROLL | ES_READONL
  GROUPBOX                "MM_MCINOTIFY Message",IDC_STATI
  LTEXT                     "",IDC_NOTIFY_ID,26,198,100,8
  LTEXT                "MCI_NOTIFY_SUCCESSFUL",IDC_NOTIFY_SU
                                  8,WS_DISABLED
  LTEXT              "MCI_NOTIFY_SUPERSEDED",IDC_NOTIFY_SUP
                                  8,WS_DISABLED
  LTEXT              "MCI_NOTIFY_ABORTED",IDC_NOTIFY_ABORTE
                                  WS_DISABLED
  LTEXT              "MCI_NOTIFY_FAILURE",IDC_NOTIFY_FAILURE,1
                                  WS_DISABLED
  DEFPUSHBUTTON       "OK",IDOK,57,255,50,14
  PUSHBUTTON              "Close",IDCANCEL,162,255,50,14
END
```

RESOURCE.H

```
// Microsoft Developer Studio generated include file.
// Used by TestMci.rc
#define IDC_MAIN_EDIT                   10
#define IDC_NOTIFY_MESSAGE
#define IDC_NOTIFY_ID                   100
#define IDC_NOTIFY_SUCCESSFUL
#define IDC_NOTIFY_SUPERSEDED
#define IDC_NOTIFY_ABORTED
#define IDC_NOTIFY_FAILURE
#define IDC_SIGNAL_MESSAGE
#define IDC_SIGNAL_ID                   10
#define IDC_SIGNAL_PARAM
#define IDC_RETURN_STRING
#define IDC_ERROR_STRING
#define IDC_DEVICES                     101
#define IDC_STATIC                      -1
```

TESTMCITESTMCIWINMM.LIBMicrosoft SettingsLinks

mciSendStringmciGetErrorTextTESTMCIEnterOK
mciSendString

```
error =     mciSendString (szCommand, szReturn,

                          sizeof (szReturn) / sizeof (TCHAR), hwnd) ;
```

mciSendStringReturn
mciGetErrorStringTESTMCIError S

MCITEXTCD

CD-ROMCDMCI/Platform
Services/Multimedia Reference/Multimedia Command StringsMCI

CD-ROMCDBruce
CDTESTMCI

```
open cdaudio
```

EnteropenMCIcdaudioMCICD-ROMCD-ROMCD-ROMsysinfo

TESTMCIReturn StringmciSendStringopen1TESTMCIError
StringmciGetErrorStringmciSendStringmciSendStringError
"The specified command was carried out"

open

```
play cdaudio
```

CDThunder Road

```
pause cdaudio
```

```
stop cdaudio
```

CD

```
play cdaudio
```

```
status cdaudio position
```

Return String

```
01:15:25
```

CD

```
status cdaudio time format
```

Return String

```
msf
```

--CD75074

msfCD

```
status cdaudio length
```

Born to RunReturn String

```
39:28:19
```

392819

```
status cdaudio number of tracks
```

Return String

```
8
```

CDBorn to RunCDMCI1Born

```
status cdaudio length track 5
```

Return String

```
04:30:22
```

```
status cdaudio position track 5
```

Return String

```
17:36:35
```

```
play cdaudio from 17:36:35 to 22:06:57
```

4:30:2217:36:35

```
status cdaudio position track 6
```

```
set cdaudio time format tmsf
```

```
play cdaudio from 5:0:0:0 to 6:0:0:0
```

```
play cdaudio from 5 to 6
```

0

MCIwaitnotifyBorn

```
play cdaudio from 5:0:0 to 5:0:10 wait
```

Born to Run10mciSendString

```
play cdaudio wait
```

mciSendStringwaitMCIbreakmciSendString
Escape

```
break cdaudio on 27
```

27VK_ESCAPE

waitnotify

```
play cdaudio from 5:0:0 to 5:0:10 notify
```

mciSendStringMCImciSendStringMM_MCINOTIFYTESTMCI
MM_MCINOTIFYTESTMCI5MM_MCINOTIFY

waitnotifywaitnotify

CD

```
stop cdaudio
```

CD-ROMCD

```
eject cdaudio
```

```
close cdaudio
```

TESTMCITESTMCICtrl-CMCI
TESTMCIOKEnterTESTMCIMCIMCI

JunglelandThunder RoadBorn to

```
open cdaudio

set cdaudio time format tmsf

break cdaudio on 27

play       cdaudio from 8 wait

play       cdaudio from 1 to 2 wait

play       cdaudio from 5 to 6 wait

stop       cdaudio

eject      cdaudio

close      cdaudio
```

waitmciSendString

CDmciSendStringWindows1
WM_TIMER

```
status cdaudio mode
```

CD

```
status cdaudio position
```

CD

Windows.WAV

API

20Hz20,000Hz

20Hz40Hz40Hz80Hz1027.5

FourierJean
34

attack

Pulse Code **Modulation**

PCMpulse code modulat

ADCanalog-to-digital
analog converterADC

NyquistNyquist

CD44,10044.1kHz

20kHz40kHz44kHz30Hz25Hz
44.1kHz

44.1kHz22.05
22.05 kHz11.025 kHz8 kHz

4186 Hz11.025 kHz4186

Alexander Graham BelldB1101dB11d
1.1210201100

$$dB = 20 \cdot \log\left(\frac{A_1}{A_2}\right)$$

A1A210

8256

$$dB = 20 \cdot \log(256)$$

484816

$$dB = 20 \cdot \log(65536)$$

96

Windows81680x80160

1681CD244

APIwaveOutSINEWAVE1

Csin2π360sin-11SINEWAVEsin
API

PCM36020
Hz1,800,000DAC

SINEWAVE11,025Hz2,756.25Hz425Hz441
2πsin

002π2π2π0

11,025Hz1,000Hz11032.6565.3197.96130.61
163.27195.92228.57261.22293.88326.53359.1831.8464.49

97.14129.80162.45195.100

22-2FillBufferSINEWAVE

22-2 SINEWAVE

```
SINEWAVE.C
/*------------------------------------------------------------------------
  SINEWAVE.C --        Multimedia Windows Sine Wave Generator
                                                (c) Charles Petzold, 199
------------------------------------------------------------------------*/
#include <windows.h>
#include <math.h>
#include "resource.h"

#define          SAMPLE_RATE              11025
#define          FREQ_MIN             20
#define          FREQ_MAX                 5000
#define          FREQ_INIT                440
#define          OUT_BUFFER_SIZE        4096
#define          PI                 3.14159
```

```
BOOL CALLBACK DlgProc (HWND, UINT, WPARAM, LPARAM) ;

TCHAR szAppName [] = TEXT ("SineWave") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                                PSTR szCmdLine, int iCr
{
        if (-1 == DialogBox (hInstance, szAppName, NULL, DlgPro
   {
             MessageBox (  NULL, TEXT ("This program requires W
                                                szAppName, MB_IC
   }
        return 0 ;
}

VOID FillBuffer (PBYTE pBuffer, int iFreq)
{
        static double        fAngle ;
        int                          i ;

        for (i = 0 ; i < OUT_BUFFER_SIZE ; i++)
        {
```

```
            pBuffer [i] = (BYTE) (127 + 127 * sin (fAngle)) ;
            fAngle += 2 * PI * iFreq / SAMPLE_RATE ;
            if (   fAngle > 2 * PI)
                        fAngle -= 2 * PI ;
   }
}

BOOL CALLBACK DlgProc (    HWND hwnd, UINT message, WPARA
{
        static BOOL                     bShutOff, bClosing ;
        static HWAVEOUT                 hWaveOut ;
        static HWND                     hwndScroll ;
        static int                      iFreq = FREQ_INIT ;
        static PBYTE                    pBuffer1, pBuffer2 ;
        static PWAVEHDR                 pWaveHdr1, pWaveHd
        static WAVEFORMATEX         waveformat ;
        int                                     iDummy ;

        switch (message)
```

```
{
case   WM_INITDIALOG:
           hwndScroll = GetDlgItem (hwnd, IDC_SCROLL) ;
           SetScrollRange(hwndScroll, SB_CTL, FREQ_MIN,
           SetScrollPos (hwndScroll, SB_CTL, FREQ_INIT, TR
           SetDlgItemInt (hwnd, IDC_TEXT, FREQ_INIT, FAL

           return TRUE ;

case   WM_HSCROLL:
           switch (LOWORD (wParam))
           {
           case   SB_LINELEFT:         iFreq -=  1 ;  break ;
           case   SB_LINERIGHT:         iFreq +=  1 ;  break ;
          case   SB_PAGELEFT:         iFreq /=  2 ;  break ;
           case   SB_PAGERIGHT: iFreq *=  2 ;  break ;

          case   SB_THUMBTRACK:
```

```
                    iFreq = HIWORD (wParam) ;
                    break ;

               case   SB_TOP:
                    GetScrollRange (hwndScroll, SB_C
                    break ;

               case   SB_BOTTOM:
                    GetScrollRange (hwndScroll, SB_C
                    break ;
               }

               iFreq = max (FREQ_MIN, min (FREQ_MAX, iFreq)
               SetScrollPos (hwndScroll, SB_CTL, iFreq, TRUE) ;
               SetDlgItemInt (hwnd, IDC_TEXT, iFreq, FALSE) ;
               return TRUE ;

          case   WM_COMMAND:
               switch (LOWORD (wParam))
```

```
                {
                case   IDC_ONOFF:
                                // If turning on waveform, hWaveO

                                if (hWaveOut == NULL)
                                {
          // Allocate memory for 2 headers and 2 buffers

          pWaveHdr1     = malloc (sizeof (WAVEHDR)) ;
          pWaveHdr2     = malloc (sizeof (WAVEHDR)) ;
          pBuffer1           = malloc (OUT_BUFFER_SIZE) ;
          pBuffer2           = malloc (OUT_BUFFER_SIZE) ;

if (!pWaveHdr1 || !pWaveHdr2 || !pBuffer1 || !pBuffer2)
        {
if (!pWaveHdr1) free (pWaveHdr1) ;
if (!pWaveHdr2) free (pWaveHdr2) ;
if (!pBuffer1)  free (pBuffer1) ;
if (!pBuffer2)  free (pBuffer2) ;
```

```
    MessageBeep (MB_ICONEXCLAMATION) ;
    MessageBox (hwnd, TEXT ("Error allocating memory!"),
               szAppName, MB_ICONEXCLAMATION | MB_OK) ;
    return TRUE ;
   }

                // Variable to indicate Off button pressed

                                        bShutOff = FALSE ;

                // Open waveform audio for output

waveformat.wFormatTag                    = WAVE_FORMAT
waveformat.nChannels                        = 1 ;
waveformat.nSamplesPerSec                = SAMPLE_RATE ;
waveformat.nAvgBytesPerSec             = SAMPLE_RATE ;
waveformat.nBlockAlign                     = 1 ;
waveformat.wBitsPerSample                = 8 ;
waveformat.cbSize                          = 0 ;
```

```
   if (waveOutOpen (&hWaveOut, WAVE_MAPPER, &waveformat,
DWORD) hwnd, 0, CALLBACK_WINDOW)!= MMSYSERR_NOERRO
                                    {
       free (pWaveHdr1) ;
       free (pWaveHdr2) ;
       free (pBuffer1) ;
      free (pBuffer2) ;

       hWaveOut = NULL ;
       MessageBeep (MB_ICONEXCLAMATION) ;
       MessageBox (hwnd, TEXT ("Error opening waveform audio
                szAppName, MB_ICONEXCLAMATION | MB_OK) ;
      return TRUE ;
     }

      // Set up headers and prepare them

      pWaveHdr1->lpData                                    = pBuffe
      pWaveHdr1->dwBufferLength                                 = OU
```

```
 pWaveHdr1->dwBytesRecorded                    = 0
 pWaveHdr1->dwUser                              = 0 ;
pWaveHdr1->dwFlags                             = 0 ;
 pWaveHdr1->dwLoops                             = 1 ;
 pWaveHdr1->lpNext                             = NULL ;
 pWaveHdr1->reserved                            = 0 ;

 waveOutPrepareHeader (hWaveOut, pWaveHdr1, sizeof (WA

pWaveHdr2->lpData                             = pBuffer
 pWaveHdr2->dwBufferLength                      = OU
 pWaveHdr2->dwBytesRecorded                     = 0
 pWaveHdr2->dwUser                              = 0 ;
 pWaveHdr2->dwFlags                             = 0 ;
pWaveHdr2->dwLoops                             = 1 ;
 pWaveHdr2->lpNext                             = NULL ;
 pWaveHdr2->reserved                            = 0 ;

waveOutPrepareHeader (hWaveOut, pWaveHdr2, sizeof (WA
```

```
                                }
// If turning off waveform, reset waveform audio
        else
                                {
                                        bShutOff = TRUE ;
                                        waveOutReset (hWaveOut) ;
                                }
                                return TRUE ;
                }
                break ;
                        // Message generated from waveOutO

    case   MM_WOM_OPEN:
                SetDlgItemText (hwnd, IDC_ONOFF, TEXT ("Turn
                                        // Send two buffers to wavef

                FillBuffer (pBuffer1, iFreq) ;
               waveOutWrite (hWaveOut, pWaveHdr1, sizeof (W
```

```
                FillBuffer (pBuffer2, iFreq) ;
                waveOutWrite (hWaveOut, pWaveHdr2, sizeof (W
                return TRUE ;

                                // Message generated when a

        case   MM_WOM_DONE:
                if (bShutOff)
                {
                        waveOutClose (hWaveOut) ;
                        return TRUE ;
                }

                        // Fill and send out a new buffer

                FillBuffer (((PWAVEHDR) lParam)->lpData, iFreq)
                waveOutWrite (hWaveOut, (PWAVEHDR) lParam,
                return TRUE ;
```

```
        case   MM_WOM_CLOSE:
                   waveOutUnprepareHeader (hWaveOut, pWaveHd
                   waveOutUnprepareHeader (hWaveOut, pWaveHd

                    free (pWaveHdr1) ;
                    free (pWaveHdr2) ;
                    free (pBuffer1) ;
                    free (pBuffer2) ;

                    hWaveOut = NULL ;
                    SetDlgItemText (hwnd, IDC_ONOFF, TEXT ("Turn

                    if (bClosing)
                                        EndDialog (hwnd, 0) ;

                    return TRUE ;

        case   WM_SYSCOMMAND:
                   switch (wParam)
```

```
                {
                case SC_CLOSE:
                                if (hWaveOut != NULL)
                                {
                                        bShutOff = TRUE ;
                                        bClosing = TRUE ;

                                        waveOutReset (hWaveOut) ;
                                }
                                else
                                        EndDialog (hwnd, 0) ;

                                return TRUE ;
                }
                break ;
        }
        return FALSE ;
}
```

SINEWAVE.RC

```
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
//////////////////////////////////////////////////////////////////////////////
// Dialog
SINEWAVE          DIALOG DISCARDABLE  100, 100, 200, 50
STYLE                   WS_MINIMIZEBOX | WS_VISIBLE | WS_CAPTIC
CAPTION           "Sine Wave Generator"
FONT 8,           "MS Sans Serif"
BEGIN
   SCROLLBAR                        IDC_SCROLL,8,8,150,12
   RTEXT                                 "440",IDC_TEXT,160,10,2(
   LTEXT                                "Hz",IDC_STATIC,182,10,12
   PUSHBUTTON                      "Turn On",IDC_ONOFF,80,28,40,
END
```

RESOURCE.H

```
// Microsoft Developer Studio generated include file.
```

```
// Used by SineWave.rc

#define IDC_STATIC                      -

#define IDC_SCROLL                    100

#define IDC_TEXT                     1001

#define IDC_ONOFF                     100
```

FillBufferOUT_BUFFER_SIZESAMPLE_RATEPIFillBufferiFreq Hzsin0254sinfAngle2π

SINEWAVETurn

SINEWAVEWM_INITDIALOG20Hz5000Hz440 DlgProcWM_HSCROLLiFreqPage LeftPag

DlgProcWM_COMMAND42WAVEHDRpBuffer1pBuffer2

waveOutOpenSINEWAVEwaveOutOpen

```
waveOutOpen (&hWaveOut, wDeviceID, &waveformat, dwCallBa

                        dwCallBackData, dwFlags) ;
```

HWAVEOUThandle to waveform audio output

waveOutOpenID0waveOutGetNumDevswaveOutGetDevCaps WAVE_MAPPER-1

WAVEFORMATEXcallbackcallbackcallbackdwFlags CALLBACK_WINDOWCALLBACK_FUNCTION WAVE_FORMAT_QUERY

waveOutOpenWAVEFORMATEXMMSYSTEM.H

```
typedef struct waveformat_tag
{
        WORD  wFormatTag ;          // waveform format = WAVE_
        WORD  nChannels ;             // number of channels = 1
       DWORD nSamplesPerSec ;          // sample rate
        DWORD nAvgBytesPerSec ;      // bytes per second
        WORD  nBlockAlign ;           // block alignment
        WORD  wBitsPerSample ;          // bits per samples = 8
        WORD  cbSize ;                // 0 for PCM
}
WAVEFORMATEX, * PWAVEFORMATEX ;
```

nSamplesPerSecnBitsPerSamplenChannelsPCMPCM
0

PCMnBlockAlignnChannelswBitsPerSample8nAvgBytesPerSec
nSamplesPerSecnBlockAlign

SINEWAVEWAVEFORMATEXwaveOutOpen

```
waveOutOpen (      &hWaveOut, WAVE_MAPPER, &waveformat,
                (DWORD) hwnd, 0, CALLBACK_WINDOW)
```

waveOutOpenMMSYSERR_NOERROR00waveOutOpen0
SINEWAVE

SINEWAVEWAVEHDRAPIWAVEHDR

```
typedef struct wavehdr_tag
{
  LPSTR lpData;                                  // pointer to data bu
     DWORD dwBufferLength;                            // length of d
  DWORD dwBytesRecorded;                             // used for r
  DWORD dwUser;                                    // for program u
  DWORD dwFlags;                                   // flags
  DWORD dwLoops;                                    // number of re
  struct wavehdr_tag FAR *lpNext;                  // reserved
  DWORD reserved;                                   // reserved
}
WAVEHDR, *PWAVEHDR ;
```

SINEWAVElpDatadwBufferLengthdwLoops10NULLdwFlags
dwLoops

SINEWAVEwaveOutPrepareHeader

waveOutOpenwaveOutOpenMM_WOM_OPENwParam
MM_WOM_OPENSINEWAVEFillBufferpBufferSINEWAVE

WAVEHDRwaveOutWrite

waveOutWriteMM_WOM_DONEwParamlParamWAVEHDR
SINEWAVEwaveOutWrite

SINEWAVEWAVEHDRMM_WOM_DONESINEWAVE

Turn OffDlgProcWM_COMMANDDlgProcbShutOffTRUE
waveOutResetMM_WOM_DONEbShutOffTRUESINEWAVE
waveOutCloseMM_WOM_DONEMM_WOM_CLOSE
MM_WOM_CLOSESINEWAVEWAVEHDRwaveOutUnprepareHeader
Turn On

waveOutClosewaveOutResetMM_WOM_DONEwParam
SC_CLOSEDlgProcWM_SYSCOMMANDCloseDlgProc
waveOutResetEndDialog

Windows22-3RECORD1I/OAPI

22-3 RECORD1

```
RECORD1.C

/*------------------------------------------------------------------------

  RECORD1.C -- Waveform Audio Recorder

                                          (c) Charles Petzold, 1998

------------------------------------------------------------------------*/

#include <windows.h>

#include "resource.h"
```

```
#define INP_BUFFER_SIZE 16384
BOOL CALLBACK DlgProc (HWND, UINT, WPARAM, LPARAM) ;
TCHAR szAppName [] = TEXT ("Record1") ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                                PSTR szCmdLine, int iCr
{
        if (-1 == DialogBox (hInstance, TEXT ("Record"), NULL, Dl
        {
            MessageBox (  NULL, TEXT ("This program requires Wi
                                                    szAppName, MB_IC
        }
        return 0 ;
}
void ReverseMemory (BYTE * pBuffer, int iLength)
{
        BYTE   b ;
        int    i ;
```

```
        for (i = 0 ; i < iLength / 2 ; i++)
        {
             b = pBuffer [i] ;
             pBuffer [i] = pBuffer [iLength - i - 1] ;
             pBuffer [iLength - i - 1] = b ;
        }
}

BOOL CALLBACK DlgProc (    HWND hwnd, UINT message, WPARA
{
        static BOOL                    bRecording, bPlaying, bRever
                                                        bEnding, bTer
        static DWORD                     dwDataLength, dwRepetitio
        static HWAVEIN             hWaveIn ;
        static HWAVEOUT              hWaveOut ;
        static PBYTE                   pBuffer1, pBuffer2, pSaveBuf
        static PWAVEHDR            pWaveHdr1, pWaveHdr2 ;
        static       TCHAR             szOpenError[] = TEXT ("Error
```

```
        static       TCHAR                szMemError [] = TEXT ("Error

        static WAVEFORMATEX waveform ;


        switch (message)
{
        case   WM_INITDIALOG:
                                // Allocate memory for wave header

                   pWaveHdr1 = malloc (sizeof (WAVEHDR)) ;
                    pWaveHdr2 = malloc (sizeof (WAVEHDR)) ;

                                // Allocate memory for save buffer

                    pSaveBuffer = malloc (1) ;
                    return TRUE ;

        case   WM_COMMAND:
                   switch (LOWORD (wParam))
                    {
```

```
                case   IDC_RECORD_BEG:
                        // Allocate buffer memory

                        pBuffer1 = malloc (INP_BUFFER_SIZE)
                        pBuffer2 = malloc (INP_BUFFER_SIZE)

                        if (!pBuffer1 || !pBuffer2)
                        {
                              if (pBuffer1) free (pBuffer1) ;
                              if (pBuffer2) free (pBuffer2) ;

                              MessageBeep (MB_ICONEXCLAMA
                              MessageBox (hwnd, szMemError,
                        MB_ICONEXCLAMATION | MB_OK) ;
                              return TRUE ;
                        }

                              // Open waveform audio for input
```

```
                    waveform.wFormatTag
                    waveform.nChannels
                    waveform.nSamplesPerSec          =
                    waveform.nAvgBytesPerSec     = 1102
                    waveform.nBlockAlign              = 1 ;
                    waveform.wBitsPerSample           = 8
                    waveform.cbSize                  = 0 ;

    if (waveInOpen (&hWaveIn, WAVE_MAPPER, &wavefo
            (DWORD) hwnd, 0, CALLBACK_WINDOW))
                    {
                        free (pBuffer1) ;
                        free (pBuffer2) ;
                        MessageBeep (MB_ICONEXCLAMA
                      MessageBox (hwnd, szOpenError, s
                    MB_ICONEXCLAMATION | MB_OK) ;
                    }
                        // Set up headers and prepare the
```

```
                    pWaveHdr1->lpData          = pBuffer1
                    pWaveHdr1->dwBufferLength     = INP
                    pWaveHdr1->dwBytesRecorded    = 0
                   pWaveHdr1->dwUser           = 0 ;
                    pWaveHdr1->dwFlags          = 0 ;
                   pWaveHdr1->dwLoops          = 1 ;
                    pWaveHdr1->lpNext           = NULL ;
                    pWaveHdr1->reserved         = 0 ;
waveInPrepareHeader (hWaveIn, pWaveHdr1, sizeof (WAV


                  pWaveHdr2->lpData         = pBuffer2 ;
                    pWaveHdr2->dwBufferLength  = INP_B
                    pWaveHdr2->dwBytesRecorded
                   pWaveHdr2->dwUser                = 0 ;
                    pWaveHdr2->dwFlags               = 0 ;
                    pWaveHdr2->dwLoops                = 1
                    pWaveHdr2->lpNext           = NULL ;
                    pWaveHdr2->reserved               = 0
```

```
                    waveInPrepareHeader (hWaveIn, pWav
                    return TRUE ;

            case   IDC_RECORD_END:
                                    // Reset input to return last b

                    bEnding = TRUE ;
                    waveInReset (hWaveIn) ;
                    return TRUE ;

            case   IDC_PLAY_BEG:
                                    // Open waveform audio for

                    waveform.wFormatTag
                    waveform.nChannels          = 1 ;
                    waveform.nSamplesPerSec           =
                    waveform.nAvgBytesPerSec     = 1102
                    waveform.nBlockAlign             = 1 ;
```

```
                    waveform.wBitsPerSample      = 8
                    waveform.cbSize              = 0 ;

                    if (waveOutOpen (&hWaveOut, WAVE_
               (DWORD) hwnd, 0, CALLBACK_WINDOW))
                    {
          MessageBeep (MB_ICONEXCLAMATION) ;
          MessageBox (hwnd,     szOpenError, szAppName
          MB_ICONEXCLAMATION | MB_OK) ;
                    }
                    return TRUE ;

     case   IDC_PLAY_PAUSE:
                              // Pause or restart output

                    if (!bPaused)
                    {
waveOutPause (hWaveOut) ;
```

```
            SetDlgItemText (hwnd, IDC_PLAY_PAUSE, TEXT ("Resum
                        bPaused = TRUE ;
                              }
                              else
                              {
            waveOutRestart (hWaveOut) ;
            SetDlgItemText (hwnd, IDC_PLAY_PAUSE, TEXT ("Pause"
                        bPaused = FALSE ;
                              }
                                    return TRUE ;

                  case   IDC_PLAY_END:
                  // Reset output for close preparation

                                    bEnding = TRUE ;
                                    waveOutReset (hWaveOut) ;
                                    return TRUE ;

                  case   IDC_PLAY_REV:
```

```
                    // Reverse save buffer and p


                    bReverse = TRUE ;
                    ReverseMemory (pSaveBuffer, dw


                   SendMessage (hwnd, WM_COMMA
                    return TRUE ;


          case   IDC_PLAY_REP:
                    // Set infinite repetitions and


                    dwRepetitions = -1 ;
                    SendMessage (hwnd, WM_COMM
                    return TRUE ;


          case   IDC_PLAY_SPEED:
                    // Open waveform audio for
```

```
                                waveform.wFormatTag     = WAVE_
                                  waveform.nChannels      = 1 ;
                                  waveform.nSamplesPerSec =220
                                  waveform.nAvgBytesPerSec= 22(
                                  waveform.nBlockAlign    = 1 ;
                                  waveform.wBitsPerSample = 8 ;
                                  waveform.cbSize         = 0 ;
          if (waveOutOpen (&hWaveOut,   0, &waveform, (DWO
           CALLBACK_WINDOW))
                                  {
          essageBeep (MB_ICONEXCLAMATION) ;
MessageBox (hwnd, szOpenError, szAppName, MB_ICONEXC
                               }
                return TRUE ;
                }
                break ;

     case   MM_WIM_OPEN:
                                        // Shrink down the save buffe
```

```
                pSaveBuffer = realloc (pSaveBuffer, 1) ;

                                // Enable and disable button

                EnableWindow (GetDlgItem (hwnd, IDC_RECORD
                EnableWindow (GetDlgItem (hwnd, IDC_RECORD
                EnableWindow (GetDlgItem (hwnd, IDC_PLAY_BE
                EnableWindow (GetDlgItem (hwnd, IDC_PLAY_PA
                EnableWindow (GetDlgItem (hwnd, IDC_PLAY_EN
                EnableWindow (GetDlgItem (hwnd, IDC_PLAY_RE
                EnableWindow (GetDlgItem (hwnd, IDC_PLAY_RE
                EnableWindow (GetDlgItem (hwnd, IDC_PLAY_SP
                SetFocus (GetDlgItem (hwnd, IDC_RECORD_END

                                // Add the buffers

                waveInAddBuffer (hWaveIn, pWaveHdr1, sizeof
                waveInAddBuffer (hWaveIn, pWaveHdr2, sizeof
```

```
                                        // Begin sampling

                    bRecording = TRUE ;
                    bEnding = FALSE ;
                   dwDataLength = 0 ;
                    waveInStart (hWaveIn) ;
                    return TRUE ;

          case   MM_WIM_DATA:

                                        // Reallocate save buffer me

                    pNewBuffer = realloc (pSaveBuffer, dwDataLen
((PWAVEHDR) lParam)->dwBytesRecorded) ;

                    if (pNewBuffer == NULL)
                   {
```

```
                    waveInClose   (hWaveIn) ;
                    MessageBeep   (MB_ICONEXCLAM
                    MessageBox    (hwnd, szMemErro
          MB_ICONEXCLAMATION | MB_OK) ;
                    return TRUE ;
          }

          pSaveBuffer = pNewBuffer ;
          CopyMemory (pSaveBuffer + dwDataLength, ((P
     ((PWAVEHDR) lParam)->dwBytesRecorded) ;

          dwDataLength += ((PWAVEHDR) lParam)->dwBy

          if (bEnding)
          {
               waveInClose (hWaveIn) ;
               return TRUE ;
          }
```

```
                    // Send out a new buffer

                waveInAddBuffer (hWaveIn, (PWAVEHDR) lParam
                return TRUE ;

        case   MM_WIM_CLOSE:
                            // Free the buffer memory

                waveInUnprepareHeader (hWaveIn, pWaveHdr1
               waveInUnprepareHeader (hWaveIn, pWaveHdr2,

                free (pBuffer1) ;
                free (pBuffer2) ;

                            // Enable and disable buttons

                EnableWindow (GetDlgItem (hwnd, IDC_RECORD
                EnableWindow (GetDlgItem (hwnd, IDC_RECORD
                SetFocus (GetDlgItem (hwnd, IDC_RECORD_BEG
```

```
                if (dwDataLength > 0)
                {
                EnableWindow (GetDlgItem (hwnd, IDC_PLAY_BE
                EnableWindow (GetDlgItem (hwnd, IDC_PLAY_PA
                EnableWindow (GetDlgItem (hwnd, IDC_PLAY_EN
                EnableWindow (GetDlgItem (hwnd, IDC_PLAY_RE
                EnableWindow (GetDlgItem (hwnd, IDC_PLAY_RE
                EnableWindow (GetDlgItem (hwnd, IDC_PLAY_SP
                SetFocus (GetDlgItem (hwnd, IDC_PLAY_BEG)) ;
                }
               bRecording = FALSE ;

                if (bTerminating)
                        SendMessage (hwnd, WM_SYSCOMMA

                return TRUE ;

        case   MM_WOM_OPEN:
```

```
                    // Enable and disable buttons

     EnableWindow (GetDlgItem (hwnd, IDC_RECORD_BEG), FA
     EnableWindow (GetDlgItem (hwnd, IDC_RECORD_END), FA
     EnableWindow (GetDlgItem (hwnd, IDC_PLAY_BEG),   FALS
     EnableWindow (GetDlgItem (hwnd, IDC_PLAY_PAUSE),    T
     EnableWindow (GetDlgItem (hwnd, IDC_PLAY_END),   TRU
     EnableWindow (GetDlgItem (hwnd, IDC_PLAY_REP),    FAL
     EnableWindow (GetDlgItem (hwnd, IDC_PLAY_REV),   FALS
    EnableWindow (GetDlgItem (hwnd, IDC_PLAY_SPEED),  FAL
     SetFocus (GetDlgItem (hwnd, IDC_PLAY_END)) ;

                    // Set up header

          pWaveHdr1->lpData            = pSaveBuffer ;
          pWaveHdr1->dwBufferLength    = dwDataLeng
          pWaveHdr1->dwBytesRecorded   = 0 ;
          pWaveHdr1->dwUser            = 0 ;
```

```
                pWaveHdr1->dwFlags         = WHDR_BEGINLO
                pWaveHdr1->dwLoops          = dwRepetitions ;
                pWaveHdr1->lpNext          = NULL ;
                pWaveHdr1->reserved         = 0 ;

                        // Prepare and write

                waveOutPrepareHeader (hWaveOut, pWaveHdr1
               waveOutWrite (hWaveOut, pWaveHdr1, sizeof (W

                bEnding = FALSE ;
                bPlaying = TRUE ;
                return TRUE ;

        case   MM_WOM_DONE:
                 waveOutUnprepareHeader (hWaveOut, pWaveHdr
                   waveOutClose (hWaveOut) ;
                   return TRUE ;
```

```
case   MM_WOM_CLOSE:
                    // Enable and disable buttons

          EnableWindow (GetDlgItem (hwnd, IDC_RECORD
          EnableWindow (GetDlgItem (hwnd, IDC_RECORD
          EnableWindow (GetDlgItem (hwnd, IDC_PLAY_BE
          EnableWindow (GetDlgItem (hwnd, IDC_PLAY_PA
          EnableWindow (GetDlgItem (hwnd, IDC_PLAY_EN
          EnableWindow (GetDlgItem (hwnd, IDC_PLAY_RE
          EnableWindow (GetDlgItem (hwnd, IDC_PLAY_RE
          EnableWindow (GetDlgItem (hwnd, IDC_PLAY_SP
          SetFocus (GetDlgItem (hwnd, IDC_PLAY_BEG)) ;

          SetDlgItemText (hwnd, IDC_PLAY_PAUSE, TEXT (
          bPaused = FALSE ;
          dwRepetitions = 1 ;
          bPlaying = FALSE ;
```

```
                if (bReverse)
                {
                        ReverseMemory (pSaveBuffer, dwData
                        bReverse = FALSE ;
                }

                if (bTerminating)
                        SendMessage (hwnd, WM_SYSCOMMA

                return TRUE ;

        case   WM_SYSCOMMAND:
                switch (LOWORD (wParam))
                {
                case   SC_CLOSE:
                                if (bRecording)
                                {
                                        bTerminating = TRUE ;
                                        bEnding = TRUE ;
```

```
                    waveInReset (hWaveIn) ;
                    return TRUE ;
                }
                if (bPlaying)
                {
                    bTerminating = TRUE ;
                    bEnding = TRUE ;
                    waveOutReset (hWaveOut) ;
                    return TRUE ;
                }

                free (pWaveHdr1) ;
                free (pWaveHdr2) ;
                free (pSaveBuffer) ;
                EndDialog (hwnd, 0) ;
                return TRUE ;
            }
            break ;
```

```
        }
        return FALSE ;
}
```

RECORD.RC

```
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
//////////////////////////////////////////////////////////////////////////////
// Dialog
RECORD            DIALOG DISCARDABLE  100, 100, 152, 74
STYLE                   WS_MINIMIZEBOX | WS_VISIBLE | WS_CAPTIC
CAPTION           "Waveform Audio Recorder"
FONT 8,           "MS Sans Serif"
BEGIN
  PUSHBUTTON                "Record",IDC_RECORD_BEG,28,8,40
  PUSHBUTTON                "End",IDC_RECORD_END,76,8,40,14
  PUSHBUTTON                "Play",IDC_PLAY_BEG,8,30,40,14,WS
  PUSHBUTTON                "Pause",IDC_PLAY_PAUSE,56,30,40,1
```

```
  PUSHBUTTON      "End",IDC_PLAY_END,104,30,40,14,\
  PUSHBUTTON      "Reverse",IDC_PLAY_REV,8,52,40,14
  PUSHBUTTON      "Repeat",IDC_PLAY_REP,56,52,40,14
  PUSHBUTTON      "Speedup",IDC_PLAY_SPEED,104,52,
END
```

RESOURCE.H

```
// Microsoft Developer Studio generated include file.
// Used by Record.rc
#define         IDC_RECORD_BEG
#define         IDC_RECORD_END
#define         IDC_PLAY_BEG                    1
#define         IDC_PLAY_PAUSE                  1
#define         IDC_PLAY_END                    1
#define         IDC_PLAY_REV                   10
#define         IDC_PLAY_REP                   10
#define         IDC_PLAY_SPEED                  1
```

RECORD.RCRESOURCE.HRECORD2RECORD3

RECORD18RECORD1RecordRecordRecord

EndEndPlay Reverse RepeatSpe
PlayReverseRepeatSpeedupEnd
PausePauseResume

RECORD1EnableWindow

RECORD1RECORD111.025
22.050kHz

API

RECORD1DlgProcWM_INITDIALOGWAVEHDR
pWaveHdr1pWaveHdr2APIpSaveBuffer
RECORD1save

8REPORT1DlgProcWM_COMMANDRecordWM_COMMAND
wParamIDC_RECORD_BEGRECORD116KWAVEFORMATEX
waveInOpenWAVEHDR

waveInOpenMM_WIM_OPENRECORD111
MM_WIM_OPENRECORD1waveInAddBufferWAVEHDR
APIwaveInStart

11.025kHz816K1.5RECORD1MM_WIM_DATA
dwDataLengthWAVEHDRdwBytesRecordedRECORD1waveInClose

RECORD116KwaveInAddBufferRECORD1End

EndWM_COMMANDwParamIDC_RECORD_ENDRECORD1
bEndingTRUEwaveInResetwaveInResetMM_WIM_DATA
waveInCloseRECORD1

waveInCloseMM_WIM_CLOSERECORD116K

PlayDlgProcWM_COMMANDwParamIDC_PLAY_BEG
WAVEFORMATEXwaveOutOpen

waveOutOpenMM_WOM_OPENRECORD1PauseEnd
WAVEHDRwaveOutPrepareHeaderwaveOutWrite

MM_WOM_DONE A
waveOutCloseMM_WOM_CLOSERECORD1

EndEndWM_COMMANDwParamIDC_PLAY_END
waveOutResetMM_WOM_DONE

RECORD1PauseRECORD1waveOutPauseResume
ResumewaveOutRestart

ReverseRepeatSpeedupWM_COMMANDwParam
IDC_PLAY_REVIDC_PLAY_REPIDC_PLAY_SPEED

RECORD1ReverseMemoryWM_COMMAND
MM_WOM_CLOSE

RepeatAPIWAVEHDRdwLoopsdwFlags
WHDR_BEGINLOOPWHDR_ENDLOOPRECORD1dwFlags

WAVEFORMATEXnSamplesPerSecnAvgBytesPerSec22050
11025

MCI

RECORD1WindowsMedia

MCIMCIRECODE1MCI

MCIASCIITESTMCI

RECORD222-4MCIRECORD1

22-4 RECORD2

RECORD2.C

```
/*------------------------------------------------------------------------
  RECORD2.C -- Waveform Audio Recorder
                                                (c) Charles Petzold, 1998
------------------------------------------------------------------------*/
#include <windows.h>
#include "..\\record1\\resource.h"

BOOL CALLBACK DlgProc (HWND, UINT, WPARAM, LPARAM) ;
TCHAR szAppName [] = TEXT ("Record2") ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                                PSTR szCmdLine, int iCr
{
   if (-1 == DialogBox (hInstance, TEXT ("Record"), NULL, DlgPro
         {
               MessageBox (  NULL, TEXT ("This program requires W
                                                      szAppName, MB_IC
         }
         return 0 ;
}
```

```
void ShowError (HWND hwnd, DWORD dwError)
{
          TCHAR szErrorStr [1024] ;
         mciGetErrorString (dwError, szErrorStr, sizeof (szErrorStr)
          MessageBeep (MB_ICONEXCLAMATION) ;
          MessageBox (hwnd, szErrorStr, szAppName, MB_OK | MB_
}

BOOL CALLBACK DlgProc (    HWND hwnd, UINT message, WPARA
{
          static BOOL    bRecording, bPlaying, bPaused ;
          static TCHAR   szFileName[] = TEXT ("record2.wav") ;
          static WORD    wDeviceID ;
          DWORD                    dwError ;
          MCI_GENERIC_PARMS          mciGeneric ;
          MCI_OPEN_PARMS             mciOpen ;
          MCI_PLAY_PARMS             mciPlay ;
          MCI_RECORD_PARMS           mciRecord ;
          MCI_SAVE_PARMS             mciSave ;
```

```
      switch (message)
{
      case   WM_COMMAND:
                  switch (wParam)
                  {
                  case   IDC_RECORD_BEG:
                                    // Delete existing waveform

                                   DeleteFile (szFileName) ;

                                    // Open waveform audio

                                  mciOpen.dwCallback
                                   mciOpen.wDeviceID
                                   mciOpen.lpstrDeviceType
                                   mciOpen.lpstrElementName
                                  mciOpen.lpstrAlias                = N
```

```
                    dwError = mciSendCommand (0,
MCI_WAIT | MCI_OPEN_TYPE | MCI_OPEN_ELE
(DWORD) (LPMCI_OPEN_PARMS) &mciOpen)
                    if (dwError != 0)
                    {
                        ShowError (hwnd, dwError) ;
                        return TRUE ;
                    }
                        // Save the Device ID

                    wDeviceID = mciOpen.wDeviceID

                        // Begin recording

                    mciRecord.dwCallback
                    mciRecord.dwFrom
                    mciRecord.dwTo

                    mciSendCommand (wDeviceID, M
```

```
                (DWORD) (LPMCI_RECORD_PARMS) &mciRe

                // Enable and disable buttons

               EnableWindow (GetDlgItem (hwnd, IDC_REC
               EnableWindow (GetDlgItem (hwnd, IDC_REC
               EnableWindow (GetDlgItem (hwnd, IDC_PLA
               EnableWindow (GetDlgItem (hwnd, IDC_PLA
               EnableWindow (GetDlgItem (hwnd, IDC_PLA
                SetFocus (GetDlgItem (hwnd, IDC_RECORD

                              bRecording = TRUE ;
                              return TRUE ;

           case   IDC_RECORD_END:
                                          // Stop recording

           mciGeneric.dwCallback = 0 ;
```

```
        mciSendCommand (wDeviceID, MCI_STOP, MCI_
    (DWORD) (LPMCI_GENERIC_PARMS) &mciGeneric) ;

                                        // Save the file

        mciSave.dwCallback = 0 ;
        mciSave.lpfilename = szFileName ;

        mciSendCommand (wDeviceID, MCI_SAVE, MCI_
    (DWORD) (LPMCI_SAVE_PARMS) &mciSave) ;

                                        // Close the wavefo

        mciSendCommand (wDeviceID, MCI_CLOSE, MC
    (DWORD) (LPMCI_GENERIC_PARMS) &mciGeneric) ;

                                        // Enable and disab
```

```
        EnableWindow   (GetDlgItem   (hwnd, IDC_RECORD_BEG)
        EnableWindow   (GetDlgItem   (hwnd, IDC_RECORD_END)
        EnableWindow   (GetDlgItem   (hwnd, IDC_PLAY_BEG),
        EnableWindow   (GetDlgItem   (hwnd, IDC_PLAY_PAUSE),
        EnableWindow   (GetDlgItem   (hwnd, IDC_PLAY_END),
        SetFocus             (GetDlgItem (hwnd, IDC_PLAY_BEG)) ;

        bRecording = FALSE ;
        return TRUE ;

                case   IDC_PLAY_BEG:
                                        // Open waveform audio

                                        mciOpen.dwCallback              =
                                        mciOpen.wDeviceID               =
                                        mciOpen.lpstrDeviceType
                                        mciOpen.lpstrElementName
                                        mciOpen.lpstrAlias              =
```

```
                    dwError = mciSendCommand (
MCI_WAIT | MCI_OPEN_ELEMENT,
 (DWORD) (LPMCI_OPEN_PARMS) &mciOpen) ;

                    if (dwError != 0)
                    {
                          ShowError (hwnd, dwError) ;
                        return TRUE ;
                    }
                              // Save the Device ID

                    wDeviceID = mciOpen.wDeviceID

                              // Begin playing

                    mciPlay.dwCallback          = (DW
                    mciPlay.dwFrom
                    mciPlay.dwTo                    =
```

```
                                mciSendCommand (wDeviceID, M
                            (DWORD) (LPMCI_PLAY_PARMS) &mciPlay)

                                                // Enable and disable bu

        EnableWindow (GetDlgItem   (hwnd, IDC_RECORD_BEG),
        EnableWindow   (GetDlgItem   (hwnd, IDC_RECORD_END)
        EnableWindow   (GetDlgItem   (hwnd, IDC_PLAY_BEG), FA
        EnableWindow   (GetDlgItem   (hwnd, IDC_PLAY_PAUSE), T
        EnableWindow   (GetDlgItem   (hwnd, IDC_PLAY_END),   T
        SetFocus (GetDlgItem (hwnd, IDC_PLAY_END)) ;

        bPlaying = TRUE ;
        return TRUE ;

                    case   IDC_PLAY_PAUSE:
                                    if (!bPaused)
                                                    // Pause the play
```

```
                    {
          mciGeneric.dwCallback = 0 ;

          mciSendCommand (wDeviceID, MCI_PAUSE, MCI_WA
            (DWORD) (LPMCI_GENERIC_PARMS) & mciGeneric);

SetDlgItemText (hwnd, IDC_PLAY_PAUSE, TEXT ("Resume")
          Paused = TRUE ;
                    }
                    else
                                        // Begin playing ag
                    {
               mciPlay.dwCallback          = (DWORD) hwnd ;
                mciPlay.dwFrom                    = 0 ;
                mciPlay.dwTo                     = 0 ;

                mciSendCommand (wDeviceID, MCI_PLAY, MCI_
                 (DWORD) (LPMCI_PLAY_PARMS) &mciPlay) ;
```

```
SetDlgItemText (hwnd, IDC_PLAY_PAUSE, TEXT ("Pause")) ;
                bPaused = FALSE ;
                        }

                        return TRUE ;

        case   IDC_PLAY_END:
                                // Stop and close

        mciGeneric.dwCallback = 0 ;

        mciSendCommand (wDeviceID, MCI_STOP, MCI_WAIT
    (DWORD) (LPMCI_GENERIC_PARMS) &mciGeneric) ;

        mciSendCommand (wDeviceID, MCI_CLOSE, MCI_WAI
    (DWORD) (LPMCI_GENERIC_PARMS) &mciGeneric) ;

     // Enable and disable buttons
```

```
        EnableWindow (GetDlgItem   (hwnd, IDC_RECORD_BE
        EnableWindow (GetDlgItem   (hwnd, IDC_RECORD_EN
        EnableWindow (GetDlgItem   (hwnd, IDC_PLAY_BEG),
        EnableWindow (GetDlgItem   (hwnd, IDC_PLAY_PAUSE
        EnableWindow (GetDlgItem   (hwnd, IDC_PLAY_END),
        SetFocus (GetDlgItem (hwnd, IDC_PLAY_BEG)) ;

        bPlaying = FALSE ;
        bPaused  = FALSE ;
        return TRUE ;
              }
              break ;

    case   MM_MCINOTIFY:
              switch (wParam)
              {
              case   MCI_NOTIFY_SUCCESSFUL:
```

```
                    if (bPlaying)
            SendMessage (hwnd, WM_COMMAND, IDC_PLAY_END

                    if (bRecording)
            SendMessage (hwnd, WM_COMMAND, IDC_RECORD_

                    return TRUE ;
                }
                break ;

        case   WM_SYSCOMMAND:
                switch (wParam)
                {
                case   SC_CLOSE:
                    if (bRecording)
                SendMessage (hwnd, WM_COMMAND, IDC_RECO
                    if (bPlaying)
                SendMessage (hwnd, WM_COMMAND, IDC_PLAY
```

```
                                EndDialog (hwnd, 0) ;
                                return TRUE ;
                    }
                    break ;
    }
    return FALSE ;
}
```

RECORD2MCI

```
error = mciSendCommand (wDeviceID, message, dwFlags, dwPa
```

IDIDIDmciSendCommandMCIMCIRECORD2
MCI_OPENMCI_RECORDMCI_STOPMCI_SAVEMCI_PLAY
MCI_PAUSEMCI_CLOSE

dwFlags0CORdwParamMCI

mciSendCommand0

```
mciGetErrorString (error, szBuffer, dwLength)
```

TESTMCI

RecordRECORD2WM_COMMANDwParam
IDC_RECORD_BEGRECORD2MCI_OPEN_PARMSMCI_OPEN
mciSendCommandlpstrDeviceTypewaveaudiolpstrElementName0

MCIMCI_SETMCI

RECORD2mciGetErrorStringMessageBoxmciSendCommand
MCI_OPEN_PARMSwDeviceIDID

RECORD2mciSendCommandMCI_RECORD
MCI_WAVE_RECORD_PARMSdwFromzdwTo
MCI_OPEN_PARMSlpstrElementName

RECORD2MCI_WAVE_RECORD_PARMSdwCallbackmciSendCommand
MCI_NOTIFY

EndWM_COMMANDwParamIDC_RECORD_END
mciSendCommandMCI_STOPMCI_SAVEMCI_SAVE_PARMS
record2.wavMCI_CLOSE

MCI_OPEN_PARMSlpstrElementNamerecord2.wav
mciSendCommandMCI_OPEN_ELEMENTlpstrElementName.WAV
MCIMCI_OPEN_PARMSlpstrDeviceType

MCI_PLAYMCI_PLAY_PARMSmciSendCommandRECORD2

RECORD2PauseWM_COMMANDwParamIDC_PLAY_PAUSE
mciSendCommandMCI_PAUSEMCI_GENERIC_PARMS
MCI_GENERIC_PARMSMCI_PLAYmciSendCommand

EndwParamIDC_PLAY_ENDWM_COMMAND
mciSendCommandMCI_STOPMCI_CLOSE

EndMCI

MCI_RECORDMCI_PLAYmciSendCommandRECORD2MCI_NOTIFY
dwCallbackMM_MCINOTIFYwParamlParamID

MCI_STOPMCI_PAUSEmciSendCommandMM_MCINOTIFYwParam
MCI_NOTIFY_ABORTEDPauseEndRECORD2
MM_MCINOTIFYwParamMCI_NOTIFY_SUCCESSFUL
WM_COMMANDwParamIDC_PLAY_ENDEnd

MM_MCINOTIFYwParamMCI_NOTIFY_SUCCESSFUL
WM_COMMANDwParamIDC_RECORD_END

MCI

WindowsmciExecute

```
bSuccess = mciExecute (szCommand) ;
```

MCI00mciExecuteNULL0mciSendStringTESTMCIMCI
mciGetErrorStringMessageBox

mciExecuteAPIRECORD3RECORD2RECORD3RECORD1
RECORD.RCRESOURCE.H22-5

22-5　　RECORD3

```
RECORD3.C

/*------------------------------------------------------------------------

  RECORD3.C -- Waveform Audio Recorder

                                              (c) Charles Petzold, 1998

------------------------------------------------------------------------*/

#include <windows.h>

#include "..\\record1\\resource.h"


BOOL CALLBACK DlgProc (HWND, UINT, WPARAM, LPARAM) ;

TCHAR szAppName [] = TEXT ("Record3") ;
```

```
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                        PSTR szCmdLine, int iCr
{
        if (-1 == DialogBox (hInstance, TEXT ("Record"), NULL, DI
        {
             MessageBox (  NULL, TEXT ("This program requires W
                                                szAppName, MB_IC
  }
        return 0 ;
}

BOOL mciExecute (LPCTSTR szCommand)
{
        MCIERROR error ;
        TCHAR    szErrorStr [1024] ;

        if (error = mciSendString (szCommand, NULL, 0, NULL))
        {
             mciGetErrorString (error, szErrorStr, sizeof (szErrorSt
             MessageBeep (MB_ICONEXCLAMATION) ;
```

```
            MessageBox (  NULL, szErrorStr, TEXT ("MCI Error"),
                                                    MB_OK | MB_ICONE
        }
        return error == 0 ;
}

BOOL CALLBACK DlgProc (    HWND hwnd, UINT message, WPARA
{
        static BOOL bRecording, bPlaying, bPaused ;
        switch (message)
        {
       case   WM_COMMAND:
                    switch (wParam)
                   {
                    case   IDC_RECORD_BEG:
                                          // Delete existing waveform


                                          DeleteFile (TEXT ("record3.w
```

```
                // Open waveform audio and

            if (!mciExecute (TEXT ("open new

                return TRUE ;

            mciExecute (TEXT ("record mysou

                // Enable and disable button

            EnableWindow (GetDlgItem (hwn
            EnableWindow (GetDlgItem (hwn
            EnableWindow (GetDlgItem (hwn
            EnableWindow (GetDlgItem (hwn
           EnableWindow (GetDlgItem (hwnd
            SetFocus (GetDlgItem (hwnd, IDC

            bRecording = TRUE ;
            return TRUE ;
```

```
                case   IDC_RECORD_END:
                                        // Stop, save, and close recor


                                       mciExecute (TEXT ("stop mysoun
                                       mciExecute (TEXT ("save mysoun
                                       mciExecute (TEXT ("close mysour


                                        // Enable and disable button


                                       EnableWindow (GetDlgItem (hwn
                                       EnableWindow (GetDlgItem (hwn
                                       EnableWindow (GetDlgItem (hwn
                                      EnableWindow (GetDlgItem (hwnd
                                       EnableWindow (GetDlgItem (hwn
                                       SetFocus (GetDlgItem (hwnd, IDC


                                      bRecording = FALSE ;
                                     return TRUE ;
```

```
        case   IDC_PLAY_BEG:
                                    // Open waveform audio and

                                   if (!mciExecute (TEXT ("open recc

                                        return TRUE ;

                                  mciExecute (TEXT ("play mysounc

                                        // Enable and disable button

              EnableWindow (GetDlgItem      (hwnd, IDC_RECORD_
              EnableWindow (GetDlgItem      (hwnd, IDC_RECORD_
              EnableWindow (GetDlgItem      (hwnd, IDC_PLAY_BEG
              EnableWindow (GetDlgItem      (hwnd, IDC_PLAY_PAU
              EnableWindow (GetDlgItem      (hwnd, IDC_PLAY_END
              SetFocus (GetDlgItem (hwnd, IDC_PLAY_END)) ;
```

```
            bPlaying = TRUE ;
        return TRUE ;


                case   IDC_PLAY_PAUSE:
                                if (!bPaused)
                                        // Pause the play
                                {
                                mciExecute (TEXT ("pause mysou
                                SetDlgItemText (hwnd, IDC_PLAY_
                                bPaused = TRUE ;
                        }
                        else
                                        // Begin playing again
                        {
                              mciExecute (TEXT ("play mysounc
                              SetDlgItemText (hwnd, IDC_PLAY_I
                               bPaused = FALSE ;
                        }
```

```
                    return TRUE ;

               case   IDC_PLAY_END:
                                   // Stop and close

                    mciExecute (TEXT ("stop mysound")) ;
                    mciExecute (TEXT ("close mysound"))

                                    // Enable and disable button
                    EnableWindow (GetDlgItem (hwnd, IDC
                    EnableWindow (GetDlgItem (hwnd, IDC
                    EnableWindow (GetDlgItem (hwnd, IDC
                    EnableWindow (GetDlgItem (hwnd, IDC
                    EnableWindow (GetDlgItem (hwnd, IDC
                    SetFocus (GetDlgItem (hwnd, IDC_PLA

                    bPlaying = FALSE ;
                    bPaused  = FALSE ;
```

```
                    return TRUE ;
                }
                break ;

        case   WM_SYSCOMMAND:
                switch (wParam)
                {
                case   SC_CLOSE:
                                if (bRecording)
                SendMessage (hwnd, WM_COMMAND, IDC_REC

                                if (bPlaying)
                SendMessage (hwnd, WM_COMMAND, IDC_PLAY

                                EndDialog (hwnd, 0) ;
                                return TRUE ;
                }
                break ;
}
```

```
        return FALSE ;
}
```

MCIMCIRECORD3RECORD2MM_MCINOTIFYmciExecute
End

MCIopenaliasMCI

.WAVPCM22-1

22-1 .WAV

0000	4	RIFF
0004	4	8
0008	4	WAVE
000C	4	fmt
0010	4	16

0014	2	wf.wFormatTag = WAVE_FORMAT_PCM = 1
0016	2	wf.nChannels
0018	4	wf.nSamplesPerSec
001C	4	wf.nAvgBytesPerSec
0020	2	wf.nBlockAlign
0022	2	wf.wBitsPerSample
0024	4	data
0028	4	
002C		

RIFFResource Interchange File FormatRIFF4 ASCII4328

RIFFRIFF328

WAVEfmt4fmt16WAVEFORMATEX16 WAVEFORMATPCMWAVEFORMAT

nChannels12nSamplesPerSec11,02522,05044 nAvgBytesPerSec8816nBlockAlign8 wBitsPerSample

data3281291624

8880x8090

fmtdataINFO

20Hz20,000Hz27.5Hz4186Hz

Fourier

246013511/31/511/21/3
1/4

Hermann Helmholtz1821-1894On the
1954Dover PressHelmholtz

1968Wendy CarlosSwitched on BachMoog
envelope00

824

Fourier

FourierHelmholtz

19771978Computer Music JournalPeo
MIT PressJames A. MoorerJohn GreyJohn Strawn SomeLexi
Analyzed TonesCE202112Computer
Music JournalVolume IINumber 219789

Windows20ADDSYNTH22-6

22-6 ADDSYNTH

ADDSYNTH.C

```
/*------------------------------------------------------------------------
  ADDSYNTH.C -- Additive Synthesis Sound Generation
                                        (c) Charles Petzold, 1998
------------------------------------------------------------------------*/
#include <windows.h>
#include <math.h>
#include "addsynth.h"
#include "resource.h"

#define          ID_TIMER          1
#define          SAMPLE_RATE       22050
#define          MAX_PARTIALS      21
```

```
#define          PI                3.14159

BOOL CALLBACK DlgProc (HWND, UINT, WPARAM, LPARAM) ;

TCHAR szAppName [] = TEXT ("AddSynth") ;

// Sine wave generator

// ------------------

double SineGenerator (double dFreq, double * pdAngle)

{

        double dAmp ;

        dAmp = sin (* pdAngle) ;

        * pdAngle += 2 * PI * dFreq / SAMPLE_RATE ;


        if (* pdAngle >= 2 * PI)

                * pdAngle -= 2 * PI ;


        return dAmp ;

}

// Fill a buffer with composite waveform
```

```
// -----------------------------------

VOID FillBuffer (INS ins, PBYTE pBuffer, int iNumSamples)

{

        static double        dAngle [MAX_PARTIALS] ;

        double                          dAmp, dFrq, dComp, dFrac ;

        int                                 i, iPrt, iMsecTime, iCompMax

                        // Calculate the composite maximum ampli


        iCompMaxAmp = 0 ;

        for (iPrt = 0 ; iPrt < ins.iNumPartials ; iPrt++)

        {

                iMaxAmp = 0 ;

                for (i = 0 ; i < ins.pprt[iPrt].iNumAmp ; i++)

                                iMaxAmp = max (iMaxAmp, ins.p

                iCompMaxAmp += iMaxAmp ;

   }


                        // Loop through each sample
```

```
 for (iSmp = 0 ; iSmp < iNumSamples ; iSmp++)
 {
              dComp = 0 ;
              iMsecTime = (int) (1000 * iSmp / SAMPLE_RATE)


                        // Loop through each partial
             for (iPrt = 0 ; iPrt < ins.iNumPartials ; iPrt++)
              {
                         dAmp = 0 ;
                         dFrq = 0 ;


                         for (i = 0 ; i < ins.pprt[iPrt].iNumAmp -
                        {
if (iMsecTime >= ins.pprt[iPrt].pEnvAmp[i  ].iTime &&
iMsecTime <= ins.pprt[iPrt].pEnvAmp[i+1].iTime)
              {
dFrac = (double) (iMsecTime -
ins.pprt[iPrt].pEnvAmp[i  ].iTime) /
(ins.pprt[iPrt].pEnvAmp[i+1].iTime -
```

```
        ins.pprt[iPrt].pEnvAmp[i  ].iTime) ;

        dAmp = dFrac  * ins.pprt[iPrt].pEnvAmp[i+1].iValue +
        (1-dFrac) * ins.pprt[iPrt].pEnvAmp[i  ].iValue ;
        break ;
        }
        }

        for (i = 0 ; i < ins.pprt[iPrt].iNumFrq - 1 ; i++)
        {
       if (iMsecTime >= ins.pprt[iPrt].pEnvFrq[i  ].iTime &&
          iMsecTime <= ins.pprt[iPrt].pEnvFrq[i+1].iTime)
        {
          dFrac = (double) (iMsecTime -ins.pprt[iPrt].pEnvFrq[i  ].i
                                    (ins.pprt[iPrt].pEnvFrq[i+1].iTi
                                    ins.pprt[iPrt].pEnvFrq[i  ].iTime
          dFrq = dFrac  * ins.pprt[iPrt].pEnvFrq[i+1].iValue + (1-dF
ins.pprt[iPrt].pEnvFrq[i  ].iValue ;
```

```
            break ;
             }
             }
            dComp += dAmp * SineGenerator (dFrq, dAngle + iPrt) ;
                }
            pBuffer[iSmp] = (BYTE) (127 + 127 * dComp / iCompMax
            }
}

// Make a waveform file
// -------------------------------------------------------------------

BOOL MakeWaveFile (INS ins, TCHAR * szFileName)
{
            DWORD                       dwWritten ;
            HANDLE                      hFile ;
            int                         iChunkSize, iPcmSize, iNumSamp
            PBYTE                       pBuffer ;
            WAVEFORMATEX                waveform ;
```

```
        hFile = CreateFile (szFileName, GENERIC_WRITE, 0, NULL,
           CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL) ;
    if (hFile == NULL)
                    return FALSE ;
        iNumSamples = ((long) ins.iMsecTime * SAMPLE_RATE / 1
        iPcmSize    = sizeof (PCMWAVEFORMAT) ;
        iChunkSize    = 12 + iPcmSize + 8 + iNumSamples ;

        if (NULL == (pBuffer = malloc (iNumSamples)))
        {
             CloseHandle (hFile) ;
             return FALSE ;
       }

        FillBuffer (ins, pBuffer, iNumSamples) ;
        waveform.wFormatTag                 = WAVE_FORMAT_
        waveform.nChannels                  = 1 ;
        waveform.nSamplesPerSec             = SAMPLE_RATE ;
        waveform.nAvgBytesPerSec            = SAMPLE_RATE ;
```

```
waveform.nBlockAlign                    = 1 ;
waveform.wBitsPerSample  = 8 ;
waveform.cbSize          = 0 ;

WriteFile     (hFile, "RIFF", 4, &dwWritten, NULL) ;
WriteFile     (hFile, &iChunkSize, 4, &dwWritten, NULL) ;
WriteFile     (hFile, "WAVEfmt ", 8, &dwWritten, NULL) ;
WriteFile     (hFile, &iPcmSize, 4, &dwWritten, NULL) ;
WriteFile     (hFile, &waveform, sizeof (WAVEFORMATEX) -
WriteFile     (hFile, "data", 4, &dwWritten, NULL) ;
WriteFile     (hFile, &iNumSamples, 4, &dwWritten, NULL)
WriteFile (hFile, pBuffer,              iNumSamples,  &dwW

CloseHandle (hFile) ;
free (pBuffer) ;

if ((int) dwWritten != iNumSamples)
{
```

```
                DeleteFile (szFileName) ;
                return FALSE ;
          }
          return TRUE ;
}

void TestAndCreateFile (   HWND hwnd, INS ins, TCHAR * szFileNa
                                                  int idButton)
{
          TCHAR szMessage [64] ;
          if (-1 != GetFileAttributes (szFileName))
                    EnableWindow (GetDlgItem (hwnd, idButton), TR
          else
          {
                if (MakeWaveFile (ins, szFileName))
                              EnableWindow (GetDlgItem (hwnd, idB
                else
                {
                              wsprintf (szMessage, TEXT ("Could not
```

```
                    MessageBeep (MB_ICONEXCLAMATION)
                    MessageBox (hwnd,     szMessage, szA
               MB_OK | MB_ICONEXCLAMATION) ;
          }
     }
}

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                        PSTR szCmdLine, int iCr
{
     if (-1 == DialogBox (hInstance, szAppName, NULL, DlgPro
     {
          MessageBox (  NULL, TEXT ("This program requires W
                                        szAppName, MB_IC
     }
     return 0 ;
}

BOOL CALLBACK DlgProc (    HWND hwnd, UINT message, WPAR
{
```

```
 static TCHAR * szTrum = TEXT ("Trumpet.wav") ;
 static TCHAR * szOboe = TEXT ("Oboe.wav") ;
 static TCHAR * szClar = TEXT ("Clarinet.wav") ;

switch (message)
 {
 case   WM_INITDIALOG:
            SetTimer (hwnd, ID_TIMER, 1, NULL) ;
            return TRUE ;

 case   WM_TIMER:
            KillTimer (hwnd, ID_TIMER) ;
            SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
            ShowCursor (TRUE) ;

            TestAndCreateFile (hwnd, insTrum, szTrum, IDC_
            TestAndCreateFile (hwnd, insOboe, szOboe, IDC_
            TestAndCreateFile (hwnd, insClar, szClar, IDC_CL
```

```
                SetDlgItemText (hwnd, IDC_TEXT, TEXT (" ")) ;
                SetFocus (GetDlgItem (hwnd, IDC_TRUMPET)) ;

                ShowCursor (FALSE) ;
                SetCursor (LoadCursor (NULL, IDC_ARROW)) ;
                return TRUE ;

        case   WM_COMMAND:
                switch (LOWORD (wParam))
                {
                case   IDC_TRUMPET:
                                PlaySound (szTrum, NULL, SND_F
                                return TRUE ;

                case   IDC_OBOE:
                                PlaySound (szOboe, NULL, SND_F
                                return TRUE ;
```

```
                case   IDC_CLARINET:
                                PlaySound (szClar, NULL, SND_FIL
                                return TRUE ;
                }
                break ;

        case   WM_SYSCOMMAND:
                switch (LOWORD (wParam))
                {
                case SC_CLOSE:
                                EndDialog (hwnd, 0) ;
                                return TRUE ;
                }
                break ;
        }
        return FALSE ;
}
```

ADDSYNTH.RC

```
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
/////////////////////////////////////////////////////////////////////////////
// Dialog
ADDSYNTH          DIALOG DISCARDABLE  100, 100, 176, 49
STYLE                   WS_MINIMIZEBOX | WS_CAPTION | WS_SYSM
CAPTION           "Additive Synthesis"
FONT 8,                     "MS Sans Serif"
BEGIN
    PUSHBUTTON                      "Trumpet",IDC_TRUMPET,8,8,48
   PUSHBUTTON                      "Oboe",IDC_OBOE,64,8,48,16
   PUSHBUTTON                      "Clarinet",IDC_CLARINET,120,8,
   LTEXT                         "Preparing Data...",IDC_TEXT,8,32,1(
END
```

```
RESOURCE.H
// Microsoft Developer Studio generated include file.
// Used by AddSynth.rc
```

```
#define         IDC_TRUMPET                             1000
#define         IDC_OBOE                              1001
#define         IDC_CLARINET                            1002
#define         IDC_TEXT                            1003
```

ADDSYNTH.HADDSYNTH.HENV
/614

1221PRTPRTENVINSPRT

ADDSYNTHTrumpetOboeClarinetPCADDSYNTH
TRUMPET.WAVOBOE.WAVCLARINET.WAVPlaySound

ADDSYNTHFillBufferFillBuffer8

FillBuffer22.05

SineGeneratorSineGenerator

WAKEUP22-7WAKEUP

22-7 WAKEUP

```
WAKEUP.C

/*------------------------------------------------------------------------
  WAKEUP.C --   Alarm Clock Program
                                                (c) Charles Petzold, 1998
```

```
------------------------------------------------------------------------*/
#include <windows.h>
#include <commctrl.h>

              // ID values for 3 child windows
#define           ID_TIMEPICK          0
#define           ID_CHECKBOX           1
#define           ID_PUSHBTN           2

              // Timer ID

#define           ID_TIMER                    1

              // Number of 100-nanosecond increments (ie FILETIM
#define FTTICKSPERHOUR (60 * 60 * (LONGLONG) 10000000)
              // Defines and structure for waveform "file"
#define           SAMPRATE             11025
#define           NUMSAMPS              (3 * SAMPRATE)
#define           HALFSAMPS     (NUMSAMPS / 2)

typedef struct
```

```
{
        char        chRiff[4] ;
        DWORD       dwRiffSize ;
        char        chWave[4] ;
        char        chFmt [4] ;
        DWORD       dwFmtSize ;
        PCMWAVEFORMAT pwf ;
        char        chData[4] ;
        DWORD       dwDataSize ;
        BYTE        byData[0] ;
}
WAVEFORM ;
        // The window proc and the subclass proc
LRESULT    CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM
LRESULT    CALLBACK SubProc (HWND, UINT, WPARAM, LPARAM)

        // Original window procedure addresses for the subclasse

WNDPROC SubbedProc [3] ;
```

```
        // The current child window with the input focus

HWND hwndFocus ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                                PSTR szCmdLine, int iCr
{
        static TCHAR  szAppName [] = TEXT ("WakeUp") ;
        HWND                        hwnd ;
        MSG                         msg ;
        WNDCLASS                    wndclass ;

        wndclass.style                      = 0 ;
        wndclass.lpfnWndProc                    = WndProc ;
        wndclass.cbClsExtra                     = 0 ;
        wndclass.cbWndExtra                     = 0 ;
        wndclass.hInstance                      = hInstance ;
        wndclass.hIcon                      = LoadIcon (NULL, IDI_
        wndclass.hCursor                        = LoadCursor (NULL,
```

```
     wndclass.hbrBackground         = (HBRUSH) (1 + COLO
     wndclass.lpszMenuName          = NULL ;
     wndclass.lpszClassName         = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (  NULL, TEXT ("This program requires W
                                          szAppName, MB_IC
          return 0 ;
     }

     hwnd = CreateWindow ( szAppName, szAppName,
                         WS_OVERLAPPED | WS_CAPTION |
                       WS_SYSMENU | WS_MINIMIZEBOX,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                        CW_USEDEFAULT, CW_USEDEFAULT,
                         NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;
```

```
        while (GetMessage (&msg, NULL, 0, 0))
   {
                    TranslateMessage (&msg) ;
                   DispatchMessage (&msg) ;
        }
        return msg.wParam ;
}

LRESULT CALLBACK WndProc ( HWND hwnd, UINT message, WPA
{
        static HWND                hwndDTP, hwndCheck, hwndPush
        static WAVEFORM            waveform = { "RIFF", NUMSAM
"WAVE", "fmt ",
                                   sizeof (PCMWAVEFORMAT), 1, 1, S
                          SAMPRATE, 1, 8, "data", NUMSAMPS } ;
        static WAVEFORM                  * pwaveform ;
        FILETIME                   ft ;
        HINSTANCE                  hInstance ;
        INITCOMMONCONTROLSEX       icex ;
```

```
int                     i, cxChar, cyChar ;
LARGE_INTEGER             li ;
SYSTEMTIME                st ;

switch (message)
{
case   WM_CREATE:
                              // Some initialization stuff

          hInstance = (HINSTANCE) GetWindowLong (hwn

          icex.dwSize = sizeof (icex) ;
          icex.dwICC  = ICC_DATE_CLASSES ;
          InitCommonControlsEx (&icex) ;

          // Create the waveform file with alternating squa

          pwaveform = malloc (sizeof (WAVEFORM) + NUI
          * pwaveform = waveform ;

          for (i = 0 ; i < HALFSAMPS ; i++)
```

```
                                        if (i % 600 < 300)

                                if (i % 16 < 8)

                pwaveform->byData[i] = 25 ;

                else

                pwaveform->byData[i] = 230 ;

                      else

                if (i % 8 < 4)

                pwaveform->byData[i] = 25 ;

                else

               pwaveform->byData[i] = 230 ;

                      // Get character size and set a fixed window size

                      cxChar = LOWORD (GetDialogBaseUnits ()) ;

                      cyChar = HIWORD (GetDialogBaseUnits ()) ;

                SetWindowPos (hwnd, NULL, 0, 0, 42 * cxChar, 10 *
GetSystemMetrics (SM_CYBORDER) +GetSystemMetrics (SM_CY
,SWP_NOMOVE | SWP_NOZORDER | SWP_NOACTIVATE) ;

                                        // Create the three child windows
```

```
hwndDTP = CreateWindow (DATETIMEPICK_CLASS, TEXT (
        WS_BORDER | WS_CHILD | WS_VISIBLE | DTS_TIMEF
        2 * cxChar, cyChar, 12 * cxChar, 4 * cyChar / 3,
        hwnd, (HMENU) ID_TIMEPICK, hInstance, NULL) ;
hwndCheck = CreateWindow (TEXT ("Button"), TEXT ("Se
        WS_CHILD | WS_VISIBLE | BS_AUTOCHECKBOX,
        16 * cxChar, cyChar, 12 * cxChar, 4 * cyChar / 3,
        hwnd, (HMENU) ID_CHECKBOX, hInstance, NULL) ;

hwndPush = CreateWindow (TEXT ("Button"), TEXT ("Turn
        WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON | WS_D
        28 * cxChar, cyChar, 12 * cxChar, 4 * cyChar / 3,
        hwnd, (HMENU) ID_PUSHBTN, hInstance, NULL) ;

        hwndFocus = hwndDTP ;

                                        // Subclass the three chi

            SubbedProc [ID_TIMEPICK] = (WNDPROC)
        SetWindowLong (hwndDTP, GWL_WNDPROC, (LONG)
            SubbedProc [ID_CHECKBOX] = (WNDPROC)
```

```
            SetWindowLong (hwndCheck, GWL_WNDPROC, (LON
                SubbedProc [ID_PUSHBTN] = (WNDPROC)
            SetWindowLong (hwndPush, GWL_WNDPROC, (LONG


                    // Set the date and time picker control to th
                     // plus 9 hours, rounded down to next lowe


                GetLocalTime (&st) ;
                SystemTimeToFileTime (&st, &ft) ;
                li = * (LARGE_INTEGER *) &ft ;
               li.QuadPart += 9 * FTTICKSPERHOUR ;
                ft = * (FILETIME *) &li ;
                FileTimeToSystemTime (&ft, &st) ;
                st.wMinute = st.wSecond = st.wMilliseconds = (
                SendMessage (hwndDTP, DTM_SETSYSTEMTIME
                return 0 ;

        case   WM_SETFOCUS:
                SetFocus (hwndFocus) ;
```

```
            return 0 ;

        case   WM_COMMAND:
            switch (LOWORD (wParam))      // control ID
            {
            case   ID_CHECKBOX:

                        // When the user checks the "Set
                        // time in the date and time contr
                        // it the current PC time.

                    if (SendMessage (hwndCheck, BM_GETCHE
                        {
                             SendMessage (hwndDTP, DT
                            SystemTimeToFileTime (&st,
                        li = * (LARGE_INTEGER *) &ft ;

                        GetLocalTime (&st) ;
                        SystemTimeToFileTime (&st, &ft)
                        li.QuadPart -= ((LARGE_INTEGER
```

```
                    // Make sure the time is
                    // These little adjustmer
                    // the date part of the SY

               while  (      li.QuadPart < 0)
                              li.QuadPart += 2

               li.QuadPart %= 24 * FTTICKSPERH

                    // Set a one-shot timer!

               SetTimer (hwnd, ID_TIMER, (int) (

          }
                    // If button is being uncheck

     else
               KillTimer (hwnd, ID_TIMER) ;

     return 0 ;

          // The "Turn Off" button turns off the ri
```

```
                        // unchecks the "Set Alarm" button and

        case   ID_PUSHBTN:
        PlaySound (NULL, NULL, 0) ;
        SendMessage (hwndCheck, BM_SETCHECK, 0, 0) ;
        EnableWindow (hwndDTP, TRUE) ;
        EnableWindow (hwndCheck, TRUE) ;
      EnableWindow (hwndPush, FALSE) ;
        SetFocus (hwndDTP) ;
        return 0 ;
       }
 return 0 ;

                    // The WM_NOTIFY message comes from th
                    // If the user has checked "Set Alarm" and t
                    // change the alarm time, there might be a
                    // the displayed time and the one-shot time
                   // unchecks "Set Alarm" and kills any outsta

 case   WM_NOTIFY:
```

```
                switch (wParam)                         // control ID
                {
                case   ID_TIMEPICK:
                switch (((NMHDR *) lParam)->code) // notificatic
                                {
                                case   DTN_DATETIMECHANGE:
               if (SendMessage (hwndCheck, BM_GETCHECK, 0,
               {
                KillTimer (hwnd, ID_TIMER) ;
                SendMessage (hwndCheck, BM_SETCHECK, 0, 0
                }
                return 0 ;
                            }
                }
                return 0 ;

                // The WM_COMMAND message comes from the

      case   WM_TIMER:
```

```
            // When the timer message comes, kill the
            // want a one-shot) and start the annoying

        KillTimer (   hwnd, ID_TIMER) ;
        PlaySound (   (PTSTR) pwaveform,  NULL,
                                        SND_MEMORY |

            // Let the sleepy user turn off the timer by
        // space bar. If the window is minimized, it's
            // it's brought to the forefront; then the pus
            // and given the input focus.

        EnableWindow (hwndDTP, FALSE) ;
        EnableWindow (hwndCheck, FALSE) ;
        EnableWindow (hwndPush, TRUE) ;

        hwndFocus = hwndPush ;
        ShowWindow (hwnd, SW_RESTORE) ;
        SetForegroundWindow (hwnd) ;
        return 0 ;
```

```
                    // Clean up if the alarm is ringing or the timer is

        case   WM_DESTROY:
                    free (pwaveform) ;

                    if (IsWindowEnabled (hwndPush))
                                PlaySound (NULL, NULL, 0) ;

                    if (SendMessage (hwndCheck, BM_GETCHECK, 0
                                KillTimer (hwnd, ID_TIMER) ;

                    PostQuitMessage (0) ;
                    return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam)
}

LRESULT CALLBACK SubProc ( HWND hwnd, UINT message, WPA
{
        int idNext, id = GetWindowLong (hwnd, GWL_ID) ;
        switch (message)
```

```
        {
        case   WM_CHAR:
                   if (wParam == '\t')
                   {
                              idNext = id ;

                              do
                                   idNext = (idNext +
                                                 (GetKeyState (VK_SHIFT
                              while (!IsWindowEnabled (GetDlgItem

                              SetFocus (GetDlgItem (GetParent (hwr
                              return 0 ;
                   }
                   break ;

        case   WM_SETFOCUS:
                   hwndFocus = hwnd ;
                   break ;
        }
```

```
        return CallWindowProc (      SubbedProc [id], hwnd, messa

}
```

WAKEUPWndProcWM_CREATEPlaySound
SND_MEMORYSND_LOOPSND_ASYNC

WAKEUPDate-Time PickerWAKEUPSYSTEMTIMEPC
Date-Time PickerDTS

WM_CREATE8

GetLocalTimeSYSTEMTIME24

/Platform SDK/Windows Base Services/General Library/T
Reference/Time Structures/SYSTEMTIMESYSTEMTIMEFILETIME
SystemTimeToFileTimeFILETIMELARGE_INTEGERFILETIME
SYSTEMTIMEFileTimeToSystemTime

FILETIME

```
type struct _FILETIME                    // ft

{

        DWORD dwLowDateTime ;

        DWORD dwHighDateTime ;

}

FILETIME ;
```

160111100064

Microsoft C/C++64ANSI C__int64__int64WindowsWINNT.

```
typedef __int64 LONGLONG ;

typedef unsigned __int64 DWORDLONG ;
```

Windowsunion

```
typedef union _LARGE_INTEGER
{
        struct
        {
             DWORD LowPart ;
             LONG  HighPart ;
  } ;
        LONGLONG QuadPart ;
}
LARGE_INTEGER ;
```

/Platform SDK/Windows Base Services/General Library/Large Integer Operationsunion3264

MIDI

1980MIDIMusical
MIDIMIDI Manufacturers AssociationMMA http://www

MIDI

MIDIMIDI5DINMIDI31,2503,125

MIDI123

MIDIMIDIMIDIMIDIMIDI
In

C3MIDI Out

```
90 3C 40
```

90Note On3CC11273MIDIMidi

3MIDI Out

```
90 3C 00
```

Note On00Note Off

Note

MIDIMIDIMIDIMIDIMIDMIDIMIDIMIDI3
Note OnNote Off

Note OnNote
MIDIMIDIMIDI

sound moduletone generatorMIDI

MIDIMIDI OutMIDIMIDI
ThruMIDI InMIDIMIDI OutMIDIMII
InMIDI ThruMIDI ThruMIDI Out

MIDIMIDI OutMIDI InMIDI ThruMI

ROM

MIDIMIDI

```
C0 pp
```

pp0127MIDIProgram

MIDIYamaha DX7Warm Strings
GuitarYamaha TX81ZGrand PianoUpright PianoDeep
MT-32Acoustic Piano 1Acoustic Piano 2Acoustic Piano 3

Program ChangeMIDIGeneral
General MIDIGeneral MIDIGeneral MIDI

MIDI

MIDINote On

```
90 kk vv
```

kk0127v v01270Note OffProgram Chan

```
C0 pp
```

pp0127MIDI01210

Note On

```
9n kk vv
```

Program Change

```
Cn pp
```

n015MIDI1n01

16MIDI16MIDIProgram
Change

12MIDIProgram

```
C0 01

C1 05
```

122610Note

```
90 kk vv

91 kk vv
```

1Note On2Note

PCMIDI161616MIDIMIDI

MIDI

Note OnProgram ChangeMIDIMIDI22-2MIDIMIDI00x80
0xFF00x7F

22-2 MIDIn =015

MIDI		
Note Off	8n kk vv	kk = 0-127 vv = 0-127
Note On	9n kk vv	kk = 0-127 vv = 1-127, 0 = note off
Polyphonic After Touch	An kk tt	kk = 0-127 tt = 0-127
Control Change	Bn cc xx	cc = 0-121 xx = 0-127
Channel Mode Local Control	Bn 7A xx	xx = 0127
All Notes Off	Bn 7B 00	
Omni Mode Off	Bn 7C 00	
Omni Mode On	Bn 7D 00	
Mono Mode On	Bn 7E cc	cc =

Poly Mode On	Bn 7F 00	
Program Change	Cn pp	pp = 0-127
Channel After Touch	Dn tt	tt = 0-127
Pitch Wheel Change	En ll hh	ll = 70-127 hh = 70-127

60CMIDI882119

0Note OnNote OffNote

0xDn0xAn

0xBn01210xBnChannel

MIDI0xEn

22-2F0FFMIDI

MIDI0xFEActive SensingMIDI

0xF0200xF7

MIDI OutMI

MIDI.MDIMIDIMCIMIDImidiOut

MIDI

MIDIAPImidiInmidiOutMIDIMIDI

MIDImidiOutOpen

```
error = midiOutOpen (&hMidiOut, wDeviceID, dwCallBack,
                                                    dwCallBackData
```

0MIDI

HMIDIOUTMIDIMIDIIDMIDI0midiOutGetNumDevs
MIDIMAPPERMMSYSTEM.H-1NULL0

MIDIMIDI

```
error = midiOutShortMsg (hMidiOut, dwMessage) ;
```

midiOutOpen32DWORD123MIDI012dwMessage
0

MIDI50x7FC0x3C3Note C

```
0x95 0x3C 0x7F
```

midiOutShortMsgdwMessage0x007F3C95

MIDIProgram ChangeNote OnNote
Note OnNote Off

MIDI

```
midiOutReset (hMidiOut) ;
```

```
midiOutClose (hMidiOut) ;
```

MIDIAPImidiOutOpenmidiOutShortMsgmidiOutResetmidiOutClose

BACHTOCC22-8J. S. BachDToccata

22-8 BACHTOCC

```
BACHTOCC.C
/*------------------------------------------------------------------------
  BACHTOCC.C --        Bach Toccata in D Minor (First Bar)
                                                (c) Charles Petzold, 199
------------------------------------------------------------------------*/
#include <windows.h>
#define ID_TIMER    1
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
TCHAR szAppName[] = TEXT ("BachTocc") ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                                PSTR szCmdLine, int iCr
{
        HWND                 hwnd ;
        MSG                  msg ;
```

```
     WNDCLASS         wndclass ;

     wndclass.style                    = CS_HREDRAW | CS_V
     wndclass.lpfnWndProc                = WndProc ;
     wndclass.cbClsExtra                 = 0 ;
     wndclass.cbWndExtra                 = 0 ;
     wndclass.hInstance                  = hInstance ;
     wndclass.hIcon                      = LoadIcon (NULL, IDI_
     wndclass.hCursor                    = LoadCursor (NULL,
     wndclass.hbrBackground              = GetStockObject (WH
     wndclass.lpszMenuName               = NULL ;
     wndclass.lpszClassName              = szAppName ;
     if (!RegisterClass (&wndclass))
     {
          MessageBox (  NULL, TEXT ("This program requires W
                                        szAppName, MB_IC
          return 0 ;
}
```

```
  hwnd = CreateWindow ( szAppName, TEXT ("Bach Toccata in D
                        WS_OVERLAPPEDWINDOW,
                        CW_USEDEFAULT, CW_USEDEFAULT,
                       CW_USEDEFAULT, CW_USEDEFAULT,
                        NULL, NULL, hInstance, NULL) ;

          if (!hwnd)
                              return 0 ;
          ShowWindow (hwnd, iCmdShow) ;
          UpdateWindow (hwnd) ;

          while (GetMessage (&msg, NULL, 0, 0))
          {
               TranslateMessage (&msg) ;
               DispatchMessage (&msg) ;
          }
          return msg.wParam ;
}

DWORD MidiOutMessage (     HMIDIOUT hMidi, int iStatus, int iCh
```

```
                                        int iData1,  int iData2)
{
        DWORD dwMessage = iStatus | iChannel | (iData1 << 8)
        return midiOutShortMsg (hMidi, dwMessage) ;
}

LRESULT CALLBACK WndProc ( HWND hwnd, UINT message, WPA
{
        static struct
        {
             int iDur ;
              int iNote [2] ;
        }
        noteseq [] = { 110, 69, 81,  110, 67, 79,  990, 69, 81,  22
              110, 67, 79,  110, 65, 77,  110, 64, 76,  110, 62, 74,
              220, 61, 73,  440, 62, 74, 1980, -1, -1,  110, 57, 69,
              110, 55, 67,  990, 57, 69,  220, -1, -1,  220, 52, 64,
              220, 53, 65,  220, 49, 61,  440, 50, 62, 1980, -1, -1 }
```

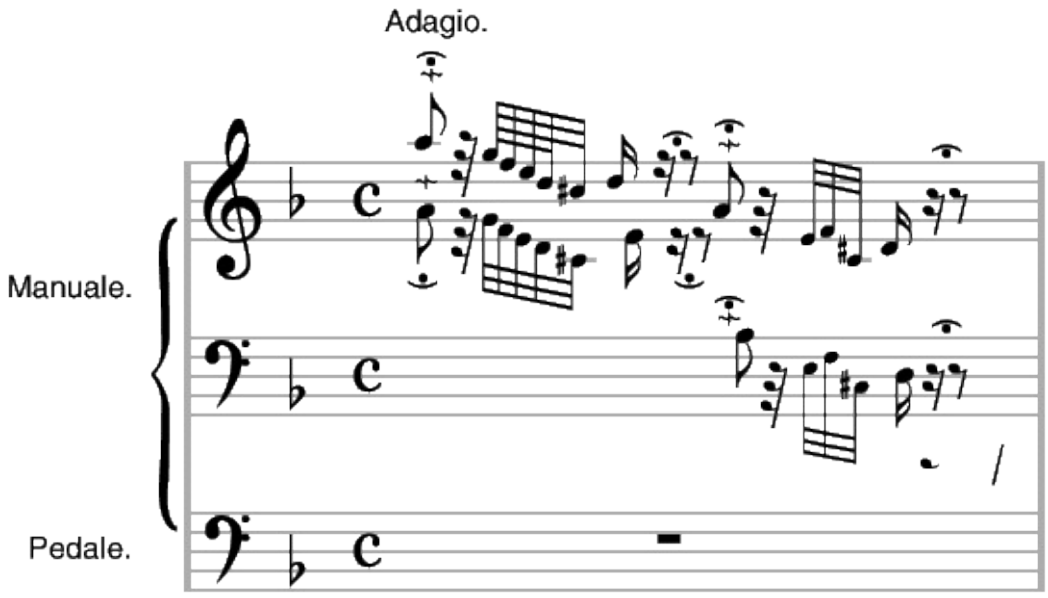
```
static HMIDIOUT             hMidiOut ;
static int                        iIndex ;
int                                    i ;

switch (message)
{
case   WM_CREATE:
                    // Open MIDIMAPPER device

          if (midiOutOpen (&hMidiOut, MIDIMAPPER, 0, 0,
          {
                    MessageBeep   (MB_ICONEXCLAMATIO
                   MessageBox    (      hwnd, TEXT ("Cann
          szAppName, MB_ICONEXCLAMATION | MB_OK) ;
                    return -1 ;
          }
                    // Send Program Change messages for
```

```
                MidiOutMessage (hMidiOut, 0xC0,  0, 19, 0) ;
                MidiOutMessage (hMidiOut, 0xC0, 12, 19, 0) ;

                SetTimer (hwnd, ID_TIMER, 1000, NULL) ;
                return 0 ;

        case   WM_TIMER:
                                // Loop for 2-note polyphony

                for (i = 0 ; i < 2 ; i++)
                {
                                        // Note Off messages for previous

             if     (iIndex != 0 && noteseq[iIndex - 1].iNote[i] != -1
        {
        MidiOutMessage (hMidiOut, 0x80,  0, noteseq[iIndex - 1].

        MidiOutMessage (hMidiOut, 0x80, 12, noteseq[iIndex - 1].
        }
```

```
 // Note On messages for new note


if (iIndex != sizeof (noteseq) / sizeof (noteseq[0]) &&
    noteseq[iIndex].iNote[i] != -1)
  {
MidiOutMessage (hMidiOut, 0x90,  0, noteseq[iIndex].iNot


MidiOutMessage (hMidiOut, 0x90, 12,noteseq[iIndex].iNot
                              }
             }


             if (iIndex != sizeof (noteseq) / sizeof (noteseq[0
            {
             SetTimer (hwnd, ID_TIMER, noteseq[iIndex++].i
             }
             else
             {
                              KillTimer (hwnd, ID_TIMER) ;
                              DestroyWindow (hwnd) ;
```

```
                }
                return 0 ;

        case   WM_DESTROY:
                midiOutReset (hMidiOut) ;
                midiOutClose (hMidiOut) ;
                PostQuitMessage (0) ;
                return 0 ;
   }
        return DefWindowProc (hwnd, message, wParam, lParam)
}
```

22-1BachDToccata



22-1 BachDToccata and Fugue

Note

BACHTOCCnoteseq1760880440
220110

AAGA110

50%

DToccata

noteseqMIDIA110CMIDI60CA69A81noteseq110
6981-1

WM_CREATEBACHTOCCWindows10001MIDIMAPPERID
midiOutOpen

BACHTOCCMIDIBACHTOCCMidiOutMessageMIDI
32midiOutShortMsg

WM_CREATEBACHTOCCProgram ChangeGener

Program Change19WM_TIMERBACHTOCCNote
012Note OnWindowsnoteseq

BACHTOCCWM_DESTROYmidiOutResetmidiOutClose

BACHTOCCWindowsWindowsPCWindows
WM_TIMERWM_TIMER

BACHTOCCMIDIWindowsWindowsMIDItime1
DRUM

MIDI

PCMIDI22-9KBMIDIPCMIDI
MIDIWindowsMIDI

22-9 KBMIDI

```
KBMIDI.C

/*------------------------------------------------------------------------

  KBMIDI.C --   Keyboard MIDI Player

                                        (c) Charles Petzold, 1998

------------------------------------------------------------------------*/

#include <windows.h>

// Defines for Menu IDs

// -------------------

#define          IDM_OPEN            0x100
```

```
#define          IDM_CLOSE            0x101
#define          IDM_DEVICE           0x200
#define          IDM_CHANNEL          0x300
#define          IDM_VOICE            0x400

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM);
TCHAR                  szAppName [] = TEXT ("KBMidi") ;
HMIDIOUT          hMidiOut ;
int                    iDevice = MIDIMAPPER, iChannel = 0, iVoice =
int                     cxCaps, cyChar, xOffset, yOffset ;

          // Structures and data for showing families and instrumen
          // ------------------------------------------------------------------

typedef struct
{
   TCHAR * szInst ;
          int    iVoice ;
}
INSTRUMENT ;
```

```
typedef     struct
{
          TCHAR      * szFam ;
          INSTRUMENT   inst [8] ;
}
FAMILY ;
FAMILY      fam [16] = {

          TEXT ("Piano"),
                TEXT ("Acoustic Grand Piano"), 0,
                TEXT ("Bright Acoustic Piano"),1,
                TEXT ("Electric Grand Piano"), 2,
                TEXT ("Honky-tonk Piano"),     3,
                TEXT ("Rhodes Piano"),         4,
                TEXT ("Chorused Piano"),       5,
                TEXT ("Harpsichord"),          6,
                TEXT ("Clavinet"),             7,
                TEXT ("Chromatic Percussion"),
```

```
        TEXT ("Celesta"),              8,
        TEXT ("Glockenspiel"),         9,
        TEXT ("Music Box"),           10,
        TEXT ("Vibraphone"),          11,
        TEXT ("Marimba"),             12,
        TEXT ("Xylophone"),           13,
        TEXT ("Tubular Bells"),       14,
        TEXT ("Dulcimer"),            15,
        TEXT ("Organ"),
        TEXT ("Hammond Organ"),       16,
        TEXT ("Percussive Organ"),    17,
        TEXT ("Rock Organ"),          18,
        TEXT ("Church Organ"),        19,
        TEXT ("Reed Organ"),          20,
        TEXT ("Accordian"),           21,
        TEXT ("Harmonica"),           22,
        TEXT ("Tango Accordian"),     23,
        TEXT ("Guitar"),
        TEXT ("Acoustic Guitar (nylon)"), 24,
```

```
 TEXT ("Acoustic Guitar (steel)"), 25,
TEXT ("Electric Guitar (jazz)"),  26,
 TEXT ("Electric Guitar (clean)"), 27,
 TEXT ("Electric Guitar (muted)"), 28,
 TEXT ("Overdriven Guitar"),       29,
 TEXT ("Distortion Guitar"),       30,
TEXT ("Guitar Harmonics"),        31,
 TEXT ("Bass"),
 TEXT ("Acoustic Bass"),           32,
 TEXT ("Electric Bass (finger)"),  33,
 TEXT ("Electric Bass (pick)"),    34,
 TEXT ("Fretless Bass"),           35,
 TEXT ("Slap Bass 1"),             36,
 TEXT ("Slap Bass 2"),             37,
 TEXT ("Synth Bass 1"),            38,
 TEXT ("Synth Bass 2"),            39,
        TEXT ("Strings"),
 TEXT ("Violin"),                40,
```

```
                TEXT ("Viola"),                 41,
                TEXT ("Cello"),                 42,
                TEXT ("Contrabass"),            43,
                TEXT ("Tremolo Strings"),       44,
                TEXT ("Pizzicato Strings"),     45,
                TEXT ("Orchestral Harp"),       46,
                TEXT ("Timpani"),               47,
TEXT ("Ensemble"),
                TEXT ("String Ensemble 1"),     48,
                TEXT ("String Ensemble 2"),     49,
               TEXT ("Synth Strings 1"),        50,
                TEXT ("Synth Strings 2"),       51,
                TEXT ("Choir Aahs"),            52,
                TEXT ("Voice Oohs"),            53,
                TEXT ("Synth Voice"),           54,
               TEXT ("Orchestra Hit"),          55,
                TEXT ("Brass"),
                TEXT ("Trumpet"),               56,
                TEXT ("Trombone"),              57,
```

```
TEXT ("Tuba"),                  58,
TEXT ("Muted Trumpet"),         59,
TEXT ("French Horn"),           60,
TEXT ("Brass Section"),         61,
TEXT ("Synth Brass 1"),         62,
TEXT ("Synth Brass 2"),         63,
TEXT ("Reed"),
TEXT ("Soprano Sax"),           64,
TEXT ("Alto Sax"),              65,
TEXT ("Tenor Sax"),             66,
TEXT ("Baritone Sax"),          67,
TEXT ("Oboe"),                  68,
TEXT ("English Horn"),          69,
TEXT ("Bassoon"),               70,
TEXT ("Clarinet"),              71,
TEXT ("Pipe"),
TEXT ("Piccolo"),               72,
TEXT ("Flute "),                73,
```

```
        TEXT ("Recorder"),              74,
      TEXT ("Pan Flute"),             75,
        TEXT ("Bottle Blow"),           76,
        TEXT ("Shakuhachi"),            77,
        TEXT ("Whistle"),               78,
      TEXT ("Ocarina"),               79,
TEXT ("Synth Lead"),
        TEXT ("Lead 1 (square)"),       80,
        TEXT ("Lead 2 (sawtooth)"),     81,
        TEXT ("Lead 3 (caliope lead)"), 82,
        TEXT ("Lead 4 (chiff lead)"),   83,
        TEXT ("Lead 5 (charang)"),      84,
        TEXT ("Lead 6 (voice)"),        85,
        TEXT ("Lead 7 (fifths)"),       86,
        TEXT ("Lead 8 (brass + lead)"), 87,
        TEXT ("Synth Pad"),
      TEXT ("Pad 1 (new age)"),       88,
        TEXT ("Pad 2 (warm)"),          89,
        TEXT ("Pad 3 (polysynth)"),     90,
```

```
TEXT ("Pad 4 (choir)"),          91,
TEXT ("Pad 5 (bowed)"),          92,
TEXT ("Pad 6 (metallic)"),       93,
TEXT ("Pad 7 (halo)"),           94,
TEXT ("Pad 8 (sweep)"),          95,
TEXT ("Synth Effects"),
TEXT ("FX 1 (rain)"),            96,
TEXT ("FX 2 (soundtrack)"),      97,
TEXT ("FX 3 (crystal)"),         98,
TEXT ("FX 4 (atmosphere)"),      99,
TEXT ("FX 5 (brightness)"),      100,
TEXT ("FX 6 (goblins)"),         101,
TEXT ("FX 7 (echoes)"),          102,
TEXT ("FX 8 (sci-fi)"),          103,
TEXT ("Ethnic"),
TEXT ("Sitar"),                  104,
TEXT ("Banjo"),                  105,
TEXT ("Shamisen"),               106,
```

```
 TEXT ("Koto"),                 107,
 TEXT ("Kalimba"),               108,
TEXT ("Bagpipe"),               109,
 TEXT ("Fiddle"),               110,
 TEXT ("Shanai"),               111,
 TEXT ("Percussive"),
 TEXT ("Tinkle Bell"),          112,
 TEXT ("Agogo"),                113,
 TEXT ("Steel Drums"),           114,
 TEXT ("Woodblock"),             115,
 TEXT ("Taiko Drum"),           116,
 TEXT ("Melodic Tom"),           117,
 TEXT ("Synth Drum"),           118,
 TEXT ("Reverse Cymbal"),        119,
 TEXT ("Sound Effects"),
 TEXT ("Guitar Fret Noise"),     120,
TEXT ("Breath Noise"),          121,
 TEXT ("Seashore"),             122,
 TEXT ("Bird Tweet"),           123,
```

```
                TEXT ("Telephone Ring"),        124,
                TEXT ("Helicopter"),            125,
                TEXT ("Applause"),              126,
                TEXT ("Gunshot"),               127
} ;

          // Data for translating scan codes to octaves and notes
          // ------------------------------------------------

#define NUMSCANS    (sizeof key / sizeof key[0])
struct
{
    int                 iOctave ;
          int                   iNote ;
          int                   yPos ;
          int                   xPos ;
         TCHAR *            szKey ;
}
key [] =
```

```
{
                        // Scan  Char  Oct  Note
                        // ----  ----  ---  ----
  -1,   -1,   1,    -1,   NULL,  //    0     None
  -1,   -1,   -1,   -1,   NULL,  //    1     Esc
  -1,   -1,   0,    0,    TEXT (""),   //    2    1
  5,    1,    0,    2,    TEXT ("C#"),  //    3     2  5       C#
  5,    3,    0,    4,    TEXT ("D#"), //    4     3     5     D#
  -1,   -1,   0,    6,    TEXT (""),   //    5    4
  5,    6,    0,    8,    TEXT ("F#"), //    6     5     5     F#
  5,    8,    0,    10,   TEXT ("G#"),  //    7     6     5     G#
  5,    10,   0,    12,   TEXT ("A#"),  //    8     7     5     A#
  -1,   -1,   0,    14,   TEXT (""),   //    9     8
  6,    1,    0,    16,   TEXT ("C#"),  //    10    9     6     C#
  6,    3,    0,    18,   TEXT ("D#"),  //    11    0     6     D#
  -1,   -1,   0,    20,   TEXT (""),   //    12    -
  6,    6,    0,    22,   TEXT ("F#"), //    13    =     6     F#
  -1,   -1,   -1,   -1,   NULL,  //    14    Back
```

```
-1,   -1,   -1,   -1,   NULL,  //    15    Tab
5,    0,    1,    1,    TEXT ("C"),   //    16    q     5     C
5,    2,    1,    3,    TEXT ("D"),   //    17    w     5     D
      5,    4,    1,    5,    TEXT ("E"),         //    18    e     5
      5,    5,    1,    7,    TEXT ("F"),         //    19    r     5
      5,    7,    1,    9,    TEXT ("G"),   //    20    t     5     G
     5,    9,    1,    11,   TEXT ("A"),         //    21    y     5
      5,    11,   1,    13,   TEXT ("B"),         //    22    u     5
      6,    0,    1,    15,   TEXT ("C"),   //    23    i     6     C
      6,    2,    1,    17,   TEXT ("D"),         //    24    o     6
      6,    4,    1,    19,   TEXT ("E"),         //    25    p     6
      6,    5,    1,    21,   TEXT ("F"),         //    26    [     6
      6,    7,    1,    23,   TEXT ("G"),         //    27    ]     6
      -1,   -1,   -1,   -1,   NULL,               //    28    Ent
      -1,   -1,   -1,   -1,   NULL,               //    29    Ctrl
      3,    8,    2,    2,    TEXT ("G#"),        //    30    a     3
      3,    10,   2,    4,    TEXT ("A#"),        //    31    s     3
      -1,   -1,   2,    6,    TEXT (""),          //    32    d
```

```
        4,    1,    2,    8,    TEXT ("C#"),        //    33    f    4
        4,    3,    2,    10,   TEXT ("D#"),        //    34    g    4
        -1,   -1,   2,    12,   TEXT (""),          //    35    h
        4,    6,    2,    14,   TEXT ("F#"),        //    36    j    4
        4,    8,    2,    16,   TEXT ("G#"),        //    37    k    4
        4,    10,   2,    18,   TEXT ("A#"),        //    38    l    4
        -1,   -1,   2,    20,   TEXT (""),          //    39    ;
        5,    1,    2,    22,   TEXT ("C#"),        //    40    '    5
        -1,   -1,   -1,   -1,   NULL,               //    41    `
        -1,   -1,   -1,   -1,   NULL,               //    42    Shift
        -1,   -1,   -1,   -1,   NULL,               //    43    \    (not l
        3,    9,    3,    3,    TEXT ("A"),   //    44    z    3    A
        3,    11,   3,    5,    TEXT ("B"),         //    45    x    3
       4,    0,    3,    7,    TEXT ("C"),         //    46    c    4
  4,    2,    3,    9,    TEXT ("D"),         //    47    v    4    D
        4,    4,    3,    11,   TEXT ("E"),         //    48    b    4
        4,    5,    3,    13,   TEXT ("F"),         //    49    n    4
        4,    7,    3,    15,   TEXT ("G"),         //    50    m    4
  4,    9,    3,    17,   TEXT ("A"),         //    51    ,    4    A
```

```
        4,    11,   3,    19,   TEXT ("B"),        //    52    .    4
        5,    0,    3,    21,   TEXT ("C")         //    53    /    5
} ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                                PSTR szCmdLine, int iCr
{
        MSG                     msg;
        HWND                     hwnd ;
        WNDCLASS                wndclass ;

        wndclass.style                          = CS_HREDRA
        wndclass.lpfnWndProc                    = WndProc ;
        wndclass.cbClsExtra                     = 0 ;
        wndclass.cbWndExtra                     = 0 ;
        wndclass.hInstance                      = hInstance ;
        wndclass.hIcon                          = LoadIcon (
        wndclass.hCursor                        = LoadCursor (N
        wndclass.hbrBackground          = GetStockObject (WH
```

```
     wndclass.lpszMenuName  = NULL ;
     wndclass.lpszClassName             = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Wi
                                                  szAppName, MB_IC
          return 0 ;
}

     hwnd = CreateWindow (szAppName, TEXT ("Keyboard MI
       WS_OVERLAPPEDWINDOW | WS_HSCROLL | WS_VSCROL
       CW_USEDEFAULT, CW_USEDEFAULT,
       CW_USEDEFAULT, CW_USEDEFAULT,
       NULL, NULL, hInstance, NULL) ;

     if (!hwnd)
               return 0 ;
     ShowWindow (hwnd, iCmdShow) ;
```

```
        UpdateWindow (hwnd);


        while (GetMessage (&msg, NULL, 0, 0))
        {
             TranslateMessage (&msg) ;
             DispatchMessage (&msg) ;
        }
        return msg.wParam ;
}

// Create the program's menu (called from WndProc, WM_CREATE
// ------------------------------------------------------------------

HMENU CreateTheMenu (int iNumDevs)
{
        TCHAR                   szBuffer [32] ;
        HMENU                    hMenu, hMenuPopup, hMenuS
        int                     i, iFam, iIns ;
        MIDIOUTCAPS          moc ;
```

```
hMenu = CreateMenu () ;
                    // Create "On/Off" popup menu
hMenuPopup = CreateMenu () ;
AppendMenu (hMenuPopup, MF_STRING, IDM_OPEN, TEXT
AppendMenu (hMenuPopup, MF_STRING | MF_CHECKED, I
      TEXT ("&Closed")) ;
AppendMenu (hMenu,    MF_STRING | MF_POPUP, (UINT) h
      TEXT ("&Status")) ;

          // Create "Device" popup menu

hMenuPopup = CreateMenu () ;
          // Put MIDI Mapper on menu if it's installed
if (!midiOutGetDevCaps (MIDIMAPPER, &moc, sizeof (moc
          AppendMenu (hMenuPopup, MF_STRING, IDM_D
                                        moc.szPname) ;
else
          iDevice = 0 ;
```

```
                    // Add the rest of the MIDI devices
        for (i = 0 ; i < iNumDevs ; i++)
        {
             midiOutGetDevCaps (i, &moc, sizeof (moc)) ;
              AppendMenu (hMenuPopup, MF_STRING, IDM_DEVICE
  }

        CheckMenuItem (hMenuPopup, 0, MF_BYPOSITION | MF_C
        AppendMenu (hMenu, MF_STRING | MF_POPUP, (UINT) hM
                                                TEXT ("&Device")) ;
                    // Create "Channel" popup menu
hMenuPopup = CreateMenu () ;
for (i = 0 ; i < 16 ; i++)
        {
              wsprintf (szBuffer, TEXT ("%d"), i + 1) ;
              AppendMenu (hMenuPopup, MF_STRING | (i ? MF_UNC
                                  IDM_CHANNEL + i, szBuffer) ;
        }
```

```
        AppendMenu (hMenu, MF_STRING | MF_POPUP, (UINT) hM
                                                TEXT ("&Channel")) ;
                        // Create "Voice" popup menu
hMenuPopup = CreateMenu () ;
        for (iFam = 0 ; iFam < 16 ; iFam++)
        {
        hMenuSubPopup = CreateMenu () ;
        for (iIns = 0 ; iIns < 8 ; iIns++)
        {
                        wsprintf (szBuffer, TEXT ("&%d.\t%s"), iIns
                                                fam[iFam].inst[iI
                        AppendMenu (hMenuSubPopup,
                MF_STRING | (fam[iFam].inst[iIns].iVoice ?
                MF_UNCHECKED : MF_CHECKED),
                fam[iFam].inst[iIns].iVoice + IDM_VOICE,
                szBuffer) ;
                   }
```

```
    wsprintf (szBuffer, TEXT ("&%c.\t%s"), 'A' + iFam,
    fam[iFam].szFam) ;
    AppendMenu (hMenuPopup, MF_STRING | MF_POPUP, (UINT) h
    szBuffer) ;
   }
        AppendMenu (hMenu,    MF_STRING | MF_POPUP, (UINT) h
     TEXT ("&Voice")) ;
  return hMenu ;
}

// Routines for simplifying MIDI output
// ------------------------------------

DWORD MidiOutMessage (     HMIDIOUT hMidi, int iStatus, int iCh
                                                    int iData1,  in
{
        DWORD dwMessage ;
        dwMessage = iStatus | iChannel | (iData1 << 8) | (iData2
        return midiOutShortMsg (hMidi, dwMessage) ;
```

```
}

DWORD MidiNoteOff (HMIDIOUT hMidi, int iChannel, int iOct, int

{

        return MidiOutMessage (hMidi, 0x080, iChannel, 12 * iOct

}

DWORD MidiNoteOn (HMIDIOUT hMidi, int iChannel, int iOct, int

{

        return MidiOutMessage (       hMidi, 0x090, iChannel, 12 *

}

DWORD MidiSetPatch (HMIDIOUT hMidi, int iChannel, int iVoice)

{

        return MidiOutMessage (hMidi, 0x0C0, iChannel, iVoice, 0

}

DWORD MidiPitchBend (HMIDIOUT hMidi, int iChannel, int iBend)

{

        return MidiOutMessage (hMidi, 0x0E0, iChannel, iBend &

}
```

```
// Draw a single key on window
// -------------------------------

VOID DrawKey (HDC hdc, int iScanCode, BOOL fInvert)
{
          RECT rc ;
          rc.left              = 3 * cxCaps * key[iScanCode].xPos / 2 +
          rc.top               = 3 * cyChar * key[iScanCode].yPos / 2 +
          rc.right             = rc.left + 3 * cxCaps ;
          rc.bottom     = rc.top  + 3 * cyChar / 2 ;

          SetTextColor  (hdc, fInvert ? 0x00FFFFFFul : 0x00000000u
          SetBkColor           (hdc, fInvert ? 0x00000000ul : 0x00FFF

          FillRect (hdc, &rc, GetStockObject (fInvert ? BLACK_BRUSH
          DrawText (hdc, key[iScanCode].szKey, -1, &rc,
                                    DT_SINGLELINE | DT_CENTER | DT_
          FrameRect (hdc, &rc, GetStockObject (BLACK_BRUSH)) ;
```

```
}

// Process a Key Up or Key Down message
// ------------------------------------

VOID ProcessKey (HDC hdc, UINT message, LPARAM lParam)
{
          int iScanCode, iOctave, iNote ;
          iScanCode = 0x0FF & HIWORD (lParam) ;
          if (iScanCode >= NUMSCANS)          // No scan codes ove
                    return ;

    if ((iOctave = key[iScanCode].iOctave) == -1)              // Non-
                    return ;

          if (GetKeyState (VK_SHIFT) < 0)
                    iOctave += 0x20000000 & lParam ? 2 : 1 ;
          if (GetKeyState (VK_CONTROL) < 0)
                    iOctave -= 0x20000000 & lParam ? 2 : 1 ;
          iNote = key[iScanCode].iNote ;
```

```
        if (message == WM_KEYUP)            // For key up
   {
            MidiNoteOff (hMidiOut, iChannel, iOctave, iNote, 0) ;
                                        DrawKey (hdc, iScanCode, FA
                                         return ;
   }

        if (0x40000000 & lParam)            // ignore typematics
            return ;

        MidiNoteOn (hMidiOut, iChannel, iOctave, iNote, iVelocity
        DrawKey (hdc, iScanCode, TRUE) ;                    // Draw th
}

// Window Procedure
// --------------------

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
```

```
        static BOOL bOpened = FALSE ;
        HDC                         hdc ;
        HMENU                       hMenu ;
        int                         i, iNumDevs, iPitchBend, cxClient
        MIDIOUTCAPS         moc ;
        PAINTSTRUCT         ps ;
        SIZE                        size ;
        TCHAR                       szBuffer [16] ;

        switch (message)
        {
        case   WM_CREATE:
                            // Get size of capital letters in system f

                    hdc = GetDC (hwnd) ;

                    GetTextExtentPoint (hdc, TEXT ("M"), 1, &size) ;
                    cxCaps = size.cx ;
                    cyChar = size.cy ;
```

```
     ReleaseDC (hwnd, hdc) ;

                    // Initialize "Volume" scroll bar

    SetScrollRange       (hwnd, SB_HORZ, 1, 127, FALS
     SetScrollPos        (hwnd, SB_HORZ, iVelocity, TI

                    // Initialize "Pitch Bend" scroll bar

     SetScrollRange       (hwnd, SB_VERT, 0, 16383,
     SetScrollPos        (hwnd, SB_VERT, 8192, TRUE)

                    // Get number of MIDI output devices a

     if (0 == (iNumDevs = midiOutGetNumDevs ()))
     {
                         MessageBeep (MB_ICONSTOP) ;
```

```
                                MessageBox (  hwnd, TEXT ("No M
                    szAppName, MB_OK | MB_ICONSTOP) ;
                                return -1 ;
                }
                SetMenu (hwnd, CreateTheMenu (iNumDevs)) ;
                return 0 ;

        case   WM_SIZE:
                cxClient = LOWORD (lParam) ;
                cyClient = HIWORD (lParam) ;

                xOffset = (cxClient - 25 * 3 * cxCaps / 2) / 2 ;
                yOffset = (cyClient - 11 * cyChar) / 2 + 5 * cyCha
                return 0 ;

        case   WM_COMMAND:
                hMenu = GetMenu (hwnd) ;

                        // "Open" menu command
```

```
            if (LOWORD (wParam) == IDM_OPEN && !bOpen
            {
                  if (midiOutOpen (&hMidiOut, iDevice, 0
                  {
                        MessageBeep (MB_ICONEXCLAMA
                        MessageBox (hwnd, TEXT ("Cann
                  szAppName, MB_OK | MB_ICONEXCLAM
                  }
                  else
                  {
                        CheckMenuItem (hMenu, IDM_OP
                       CheckMenuItem (hMenu, IDM_CLO

                        MidiSetPatch (hMidiOut, iChannel
                        bOpened = TRUE ;
      }
                  }
```

```
                    // "Close" menu command
            else if (LOWORD (wParam) == IDM_CLOSE && b
            {
                    CheckMenuItem (hMenu, IDM_OPEN,  M
                    CheckMenuItem (hMenu, IDM_CLOSE,

                        // Turn all keys off and close devic
                    for (i = 0 ; i < 16 ; i++)
                            MidiOutMessage (hMidiOut,
                    midiOutClose (hMidiOut) ;
                    bOpened = FALSE ;
            }

                    // Change MIDI "Device" menu co
            else if (     LOWORD (wParam) >= IDM_DEVICE -
                                LOWORD (wParam) <  ID
            {
                    CheckMenuItem (hMenu, IDM_DEVICE
```

```
                    iDevice = LOWORD (wParam) - IDM_DI
                    CheckMenuItem (hMenu, IDM_DEVICE

                         // Close and reopen MIDI device

                    if (bOpened)
                    {
                         SendMessage (hwnd, WM_COMMA
                         SendMessage (hwnd, WM_COMM
                    }
               }

                    // Change MIDI "Channel" menu comm

                else if (     LOWORD (wParam) >= IDM_CHANNE
                                        LOWORD (wParam) <  I
                {
                    CheckMenuItem (hMenu, IDM_CHANNE
```

```
                    iChannel = LOWORD (wParam) - IDM_C
                    CheckMenuItem (hMenu, IDM_CHANNE

                    if (bOpened)
                            MidiSetPatch (hMidiOut, iChan
          }

                    // Change MIDI "Voice" menu comman

          else if (LOWORD (wParam) >= IDM_VOICE)
          {
                    CheckMenuItem (hMenu, IDM_VOICE +
                    iVoice = LOWORD (wParam) - IDM_VOI
                    CheckMenuItem (hMenu, IDM_VOICE +

                    if (bOpened)
                            MidiSetPatch (hMidiOut, iCha
          }
```

```
                InvalidateRect (hwnd, NULL, TRUE) ;
                return 0 ;

                    // Process a Key Up or Key Down message

        case   WM_KEYUP:
        case   WM_KEYDOWN:
                hdc = GetDC (hwnd) ;

                if (bOpened)
                                ProcessKey (hdc, message, lParar

                ReleaseDC (hwnd, hdc) ;
                return 0 ;

                    // For Escape, turn off all notes and repaint

        case   WM_CHAR:
```

```
                if (bOpened && wParam == 27)
                {
                        for (i = 0 ; i < 16 ; i++)
                                MidiOutMessage (hMidiO


                        InvalidateRect (hwnd, NULL, TRUE) ;
                }
                return 0 ;

                // Horizontal scroll: Velocity

        case   WM_HSCROLL:
                switch (LOWORD (wParam))
                {
               case SB_LINEUP:                          iVelocity
                case SB_LINEDOWN:                      iVelocity -
                case SB_PAGEUP:                          iVeloci
                case SB_PAGEDOWN:                      iVelocity
                case SB_THUMBPOSITION:       iVelocity = HIWO
```

```
                default:                                                retur
        }
                iVelocity = max (1, min (iVelocity, 127)) ;
               SetScrollPos (hwnd, SB_HORZ, iVelocity, TRUE) ;
                return 0 ;

                            // Vertical scroll:  Pitch Bend
         case   WM_VSCROLL:
                switch (LOWORD (wParam))
                {
         case SB_THUMBTRACK:   iPitchBend = 16383 - HIWORD (
         case SB_THUMBPOSITION: iPitchBend = 8191 ;  break ;
         default:                                              return 0 ;
         }
                iPitchBend = max (0, min (iPitchBend, 16383)) ;
                SetScrollPos (hwnd, SB_VERT, 16383 - iPitchBen

                if (bOpened)
```

```
                                MidiPitchBend (hMidiOut, iChanne

                return 0 ;


        case   WM_PAINT:

                hdc = BeginPaint (hwnd, &ps) ;


                for (i = 0 ; i < NUMSCANS ; i++)

                                if (key[i].xPos != -1)

                                                DrawKey (hdc, i, FALSE)


                midiOutGetDevCaps (iDevice, &moc, sizeof (MID

                wsprintf (szBuffer, TEXT ("Channel %i"), iChannel


                TextOut (     hdc, cxCaps, 1 * cyChar,

        Opened ? TEXT ("Open") : TEXT ("Closed"),

        bOpened ? 4 : 6) ;

                TextOut (     hdc, cxCaps, 2 * cyChar, moc.szPna

        lstrlen (moc.szPname)) ;

                TextOut       (hdc, cxCaps, 3 * cyChar, szBuffer, l
```

```
                TextOut(hdc, cxCaps, 4 * cyChar,
            fam[iVoice / 8].inst[iVoice % 8].szInst,
                lstrlen (fam[iVoice / 8].inst[iVoice % 8].szInst)) ;

                EndPaint (hwnd, &ps) ;
                return 0 ;

        case   WM_DESTROY :
                SendMessage (hwnd, WM_COMMAND, IDM_CLOS
                PostQuitMessage (0) ;
                return 0 ;
   }
        return DefWindowProc (hwnd, message, wParam, lParam)
}
```

KBMIDIZ110

StatusOpenMIDIMIDI

StatusCloseMIDIKBMIDIWindowsMIDI

DeviceMIDImidiOutGetDevCapsMIDI
MIDI

Channel116MIDI1KBMIDIMIDI

KBMIDIVoice128General
128

General MIDIChannel10VoiceAcoustic
MIDI35CB81CA47DRUM

KBMIDIPCNote

Pitch BendMIDI

CtrlShift

EscMIDI1616All

KBMIDImidiOutGetDevCaps

KBMIDIMIDIPitch BendPitch
0x3FFF

StatusOpenKBMIDImidiOutOpenMidiSetPatch
KBMIDIMIDIMIDIKBMIDIMidiSetPatch

KBMIDIWM_KEYUPWM_KEYDOWNKBMIDIZ4439
AKBMIDIMidiNoteOnMIDI451239

SB_THUMBTRACKSB_THUMBPOSITION
SB_THUMBPOSITIONKBMIDIMidiPitchBend8192

MIDI

KBMIDIVoice

General

DRUM22-104732

DRUM.C

```
/*------------------------------------------------------------------------
  DRUM.C -- MIDI Drum Machine
                                        (c) Charles Petzold, 1998
------------------------------------------------------------------------*/
#include <windows.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "drumtime.h"
#include "drumfile.h"
#include "resource.h"

LRESULT     CALLBACK WndProc            (HWND, UINT, WPARAM,
BOOL             CALLBACK AboutProc    (HWND, UINT, WPARAM, L


void             DrawRectangle        (HDC, int, int, DWORD *, DWO
void             ErrorMessage         (HWND, TCHAR *, TCHAR *) ;
```

```
void            DoCaption           (HWND, TCHAR *) ;
int             AskAboutSave        (HWND, TCHAR *) ;

TCHAR * szPerc [NUM_PERC] =
{
        TEXT ("Acoustic Bass Drum"), TEXT ("Bass Drum 1"),
        TEXT ("Side Stick"),                TEXT ("Acoustic Snare
        TEXT ("Hand Clap"),             TEXT ("Electric Snare"),
        TEXT ("Low Floor Tom"),         TEXT ("Closed High Hat"
        TEXT ("High Floor Tom"),        TEXT ("Pedal High Hat"),
        TEXT ("Low Tom"),               TEXT ("Open High Hat"),
        TEXT ("Low-Mid Tom"),           TEXT ("High-Mid Tom"),
        TEXT ("Crash Cymbal 1"),        TEXT ("High Tom"),
        TEXT ("Ride Cymbal 1"),         TEXT ("Chinese Cymbal"
        TEXT ("Ride Bell"),             TEXT ("Tambourine"),
        TEXT ("Splash Cymbal"),         TEXT ("Cowbell"),
        TEXT ("Crash Cymbal 2"),        TEXT ("Vibraslap"),
        TEXT ("Ride Cymbal 2"),         TEXT ("High Bongo"),
        TEXT ("Low Bongo"),             TEXT ("Mute High Conga"
```

```
        TEXT ("Open High Conga"),         TEXT ("Low Conga"),
        TEXT ("High Timbale"),            TEXT ("Low Timbale"),
        TEXT ("High Agogo"),              TEXT ("Low Agogo"),
        TEXT ("Cabasa"),                                    TEXT ("
        TEXT ("Short Whistle"),                 TEXT ("Long Whistle
        TEXT ("Short Guiro"),             TEXT ("Long Guiro"),
        TEXT ("Claves"),                                    TEXT ("H
        TEXT ("Low Wood Block"),          TEXT ("Mute Cuica"),
        TEXT ("Open Cuica"),              TEXT ("Mute Triangle"),
        TEXT ("Open Triangle")
} ;

TCHAR           szAppName     []     = TEXT ("Drum") ;
TCHAR           szUntitled    []     = TEXT ("(Untitled)") ;
TCHAR           szBuffer [80 + MAX_PATH] ;
HANDLE      hInst ;
int             cxChar, cyChar ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
```

```
                                        PSTR szCmdLine, int iCr
{
        HWND                        hwnd ;
        MSG                         msg ;
        WNDCLASS                    wndclass ;

        hInst = hInstance ;
        wndclass.style              = CS_HREDRAW | CS_VR
        wndclass.lpfnWndProc            = WndProc ;
        wndclass.cbClsExtra             = 0 ;
        wndclass.cbWndExtra             = 0 ;
        wndclass.hInstance              = hInstance ;
        wndclass.hIcon                  = LoadIcon (hInstance
        wndclass.hCursor                = LoadCursor (NULL,
        wndclass.hbrBackground          = GetStockObject (WH
        wndclass.lpszMenuName           = szAppName ;
        wndclass.lpszClassName          = szAppName ;

        if (!RegisterClass (&wndclass))
```

```
        {
             MessageBox (  NULL, TEXT ("This program requires W
                                 szAppName, MB_ICONERROR) ;
             return 0 ;
   }

        hwnd = CreateWindow (szAppName, NULL,
WS_OVERLAPPED | WS_CAPTION | WS_SYSMENU |
                     WS_MINIMIZEBOX | WS_HSCROLL | WS_VSCRO
                   CW_USEDEFAULT, CW_USEDEFAULT,
                   CW_USEDEFAULT, CW_USEDEFAULT,
                  NULL, NULL, hInstance, szCmdLine) ;

        ShowWindow (hwnd, iCmdShow) ;
        UpdateWindow (hwnd) ;

        while (GetMessage (&msg, NULL, 0, 0))
        {
             TranslateMessage (&msg) ;
```

```
            DispatchMessage (&msg) ;
        }
        return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPA
{
        staticBOOL         bNeedSave ;
        static DRUM         drum ;
        static HMENU        hMenu ;
        staticint          iTempo = 50, iIndexLast ;
        static TCHAR        szFileName  [MAX_PATH], szTitleName
        HDC                                 hdc ;
        int                                 i, x, y ;
        PAINTSTRUCT                 ps ;
        POINT                               point ;
        RECT                                rect ;
        TCHAR                       *      szError ;
```

```
        switch (message)
{
        case   WM_CREATE:
                                // Initialize DRUM structure


                        drum.iMsecPerBeat = 100 ;
                        drum.iVelocity    =  64 ;
                        drum.iNumBeats    =  32 ;


                        DrumSetParams (&drum) ;


                                // Other initialization


                        cxChar = LOWORD (GetDialogBaseUnits ()) ;
                        cyChar = HIWORD (GetDialogBaseUnits ()) ;

                        GetWindowRect (hwnd, &rect) ;
                        MoveWindow (hwnd,     rect.left, rect.top,
                    77 * cxChar, 29 * cyChar, FALSE) ;
```

```
                hMenu = GetMenu (hwnd) ;

                        // Initialize "Volume" scroll bar

                SetScrollRange       (hwnd, SB_HORZ, 1, 127, FA
                SetScrollPos         (hwnd, SB_HORZ, drum.iVeloc

                        // Initialize "Tempo" scroll bar

                SetScrollRange       (hwnd, SB_VERT, 0, 100, FAl
                SetScrollPos         (hwnd, SB_VERT, iTempo, TRU

                DoCaption (hwnd, szTitleName) ;
                return 0 ;

        case   WM_COMMAND:
                switch (LOWORD (wParam))
```

```
                {
                case   IDM_FILE_NEW:
                                if     ( bNeedSave && IDCANCEL =
                                       return 0 ;

                                // Clear drum pattern

                                for (i = 0 ; i < NUM_PERC ; i++)
                                {
                                       drum.dwSeqPerc [i] = 0 ;
                                       drum.dwSeqPian [i] = 0 ;
                                }

                                InvalidateRect (hwnd, NULL, FALS
                                DrumSetParams (&drum) ;
                                bNeedSave = FALSE ;
                                return 0 ;

                case   IDM_FILE_OPEN:
```

```
                    // Save previous file

          if (bNeedSave && IDCANCEL ==
                    AskAboutSave (hwnd, sz
               return 0 ;

                    // Open the selected file

          if (DrumFileOpenDlg (hwnd, szFile
          {
               szError = DrumFileRead (&dru

               if (szError != NULL)
               {
                         ErrorMessage (hwn
                        szTitleName [0] = '
               }
              else
```

```
                                        {
                                                                // Set new p

                                        Tempo = (int) (50 *
                                        (log10 (drum.iMsecPerBeat)

            SetScrollPos (hwnd, SB_VERT, iTempo, TRUE) ;
            SetScrollPos (hwnd, SB_HORZ, drum.iVelocity, TR

            DrumSetParams (&drum) ;
            InvalidateRect (hwnd, NULL, FALSE) ;
            bNeedSave = FALSE ;
        }

        DoCaption (hwnd, szTitleName) ;
                        }
                        return 0 ;
    case   IDM_FILE_SAVE:
    case   IDM_FILE_SAVE_AS:
```

```
                    // Save the selected file

          if ((LOWORD (wParam) == IDM_FILE_S
 DrumFileSaveDlg (hwnd, szFileName, szTitleNam
          {
 szError = DrumFileWrite (&drum, szFileName) ;

if (szError != NULL)
 {
 ErrorMessage (hwnd, szError, szTitleName) ;
               szTitleName [0] = '\0' ;
                    }
                    else
                         bNeedSave = FALSE

                    DoCaption (hwnd, szTitleNam
          }
          return 0 ;
```

```
        case   IDM_APP_EXIT:
                        SendMessage (hwnd, WM_SYSCOMMA
                        return 0 ;

              case   IDM_SEQUENCE_RUNNING:
                        // Begin sequence

                        if (!DrumBeginSequence (hwnd))
                        {
                              ErrorMessage (hwnd,
                                          TEXT ("Could not start M
                                         TEXT ("MIDI Mapper devi
                                          szTitleName) ;
                         }
                         else
                         {
                               CheckMenuItem (hMenu, IDM_SE
                               CheckMenuItem (hMenu, IDM_SE
```

```
                    }
                    return 0 ;

               case   IDM_SEQUENCE_STOPPED:
                         // Finish at end of sequence

                         DrumEndSequence (FALSE) ;
                         return 0 ;

               case   IDM_APP_ABOUT:
                         DialogBox (hInst, TEXT ("AboutBo
                         return 0 ;
               }
               return 0 ;

     case   WM_LBUTTONDOWN:
     case   WM_RBUTTONDOWN:
               hdc = GetDC (hwnd) ;
```

```
                // Convert mouse coordinates to grid c

x =                 LOWORD (lParam) / cxChar - 40
y = 2 *HIWORD (lParam) / cyChar -  2 ;
                // Set a new number of beats of seque

if (x > 0 && x <= 32 && y < 0)
{
                SetTextColor (hdc, RGB (255, 255, 255
                TextOut (hdc, (40 + drum.iNumBeats) *
                SetTextColor (hdc, RGB (0, 0, 0)) ;

                if (drum.iNumBeats % 4 == 0)
                                TextOut       (      hdc, (40 + c
                                                                TEXT

                drum.iNumBeats = x ;
```

```
                TextOut (hdc, (40 + drum.iNumBeats) *

                bNeedSave = TRUE ;
            }

                // Set or reset a percussion instrument

            if (x >= 0 && x < 32 && y >= 0 && y < NUM_P
            {
                if (message == WM_LBUTTONDOWN)
                        drum.dwSeqPerc[y] ^= (1 <
                else
                        drum.dwSeqPian[y] ^= (1 <

                DrawRectangle (hdc, x, y, drum.dwSeq

                bNeedSave = TRUE ;
            }
```

```
                ReleaseDC (hwnd, hdc) ;
                DrumSetParams (&drum) ;
                return 0 ;

        case   WM_HSCROLL:
                        // Change the note velocity

                switch (LOWORD (wParam))
                {
                case SB_LINEUP:                         drum.iVel
                case SB_LINEDOWN:                       drum.iVel
                case SB_PAGEUP:                          drum.iVe
                case SB_PAGEDOWN:                        drum.iVe
                case SB_THUMBPOSITION:
                                drum.iVelocity = HIWORD (wParar
                                 break ;

                default:
```

```
                    return 0 ;
               }

               drum.iVelocity = max (1, min (drum.iVelocity, 1
               SetScrollPos (hwnd, SB_HORZ, drum.iVelocity, T
               DrumSetParams (&drum) ;
               bNeedSave = TRUE ;
               return 0 ;

     case   WM_VSCROLL:
                    // Change the tempo

               switch (LOWORD (wParam))
               {
               case SB_LINEUP:                    iTempo
               case SB_LINEDOWN:                  iTempo +
               case SB_PAGEUP:                    iTempo
              case SB_PAGEDOWN:                   iTempo +=
```

```
                case SB_THUMBPOSITION:
                                iTempo = HIWORD (wParam) ;
                                break ;

                default:
                        return 0 ;
                }

                iTempo = max (0, min (iTempo, 100)) ;
                SetScrollPos (hwnd, SB_VERT, iTempo, TRUE) ;

                drum.iMsecPerBeat = (WORD) (10 * pow (100, i

                DrumSetParams (&drum) ;
                bNeedSave = TRUE ;
               return 0 ;

        case   WM_PAINT:
                hdc = BeginPaint (hwnd, &ps) ;
```

```
SetTextAlign (hdc, TA_UPDATECP) ;
SetBkMode (hdc, TRANSPARENT) ;

          // Draw the text strings and horizontal
for (i = 0 ; i < NUM_PERC ; i++)
{
          MoveToEx (hdc, i & 1 ? 20 * cxChar : cx
 (2 * i + 3) * cyChar / 4, NULL) ;

          TextOut (hdc, 0, 0, szPerc [i], lstrlen (s

          GetCurrentPositionEx (hdc, &point) ;

          MoveToEx     (hdc,  point.x + cxChar, p
          LineTo (hdc,        39 * cxChar, point.y
}
```

```
                SetTextAlign (hdc, 0) ;

                        // Draw rectangular grid, repeat mark,

                for (x = 0 ; x < 32 ; x++)
                {
                        for (y = 0 ; y < NUM_PERC ; y++)
 DrawRectangle (hdc, x, y, drum.dwSeqPerc, drum.dwS

    SetTextColor (hdc, x == drum.iNumBeats - 1 ?
RGB (0, 0, 0) : RGB (255, 255, 255)) ;

    TextOut (hdc, (41 + x) * cxChar, 0, TEXT (":|"), 2) ;

    SetTextColor (hdc, RGB (0, 0, 0)) ;

    if (x % 4 == 0)
 TextOut (hdc, (40 + x) * cxChar, 0, TEXT ("."), 1) ;
    }
```

```
        EndPaint (hwnd, &ps) ;
        return 0 ;

    case   WM_USER_NOTIFY:
                                // Draw the "bouncing ball"

                hdc = GetDC (hwnd) ;

                SelectObject (hdc, GetStockObject (NULL_PEN))
                SelectObject (hdc, GetStockObject (WHITE_BRUS

                for (i = 0 ; i < 2 ; i++)
                {
                            x = iIndexLast ;
                            y = NUM_PERC + 1 ;

                            Ellipse (hdc, (x + 40) * cxChar, (2 * y -
```

```
                                        (x + 41) * cxChar, (2 * y + 5

                        iIndexLast = wParam ;

                        SelectObject (hdc, GetStockObject (BL

                }


                ReleaseDC (hwnd, hdc) ;

                return 0 ;


        case   WM_USER_ERROR:

                ErrorMessage (hwnd, TEXT ("Can't set timer eve

                        szTitleName) ;

              // fall through

         case   WM_USER_FINISHED:

                DrumEndSequence (TRUE) ;

                CheckMenuItem (hMenu, IDM_SEQUENCE_RUNN

                CheckMenuItem (hMenu, IDM_SEQUENCE_STOP

                return 0 ;
```

```
        case   WM_CLOSE:
                    if (!bNeedSave || IDCANCEL != AskAboutSave (h
                          DestroyWindow (hwnd) ;

                    return 0 ;

        case   WM_QUERYENDSESSION:
                    if (!bNeedSave || IDCANCEL != AskAboutSave (h
                           return 1L ;

                  return 0 ;

        case   WM_DESTROY:
                    DrumEndSequence (TRUE) ;
                    PostQuitMessage (0) ;
                    return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam)
```

```
}

BOOL CALLBACK AboutProc (  HWND hDlg, UINT message, WPAR
{
        switch (message)
        {
        case   WM_INITDIALOG:
                   return TRUE ;


        case   WM_COMMAND:
                   switch (LOWORD (wParam))
                   {
                   case   IDOK:
                                       EndDialog (hDlg, 0) ;
                                       return TRUE ;
                   }
                   break ;
        }
        return FALSE ;
```

```
}

void DrawRectangle (HDC hdc, int x, int y, DWORD * dwSeqPerc,
                                    DWORD * dwSeqPian)
{
          int iBrush ;
          if (dwSeqPerc [y] & dwSeqPian [y] & (1L << x))
                    iBrush = BLACK_BRUSH ;
          else if (dwSeqPerc [y] & (1L << x))
                              iBrush = DKGRAY_BRUSH ;
          else if (dwSeqPian [y] & (1L << x))
                              iBrush = LTGRAY_BRUSH ;
          else
                    iBrush = WHITE_BRUSH ;
          SelectObject (hdc, GetStockObject (iBrush)) ;
         Rectangle (hdc, (x + 40) * cxChar , (2 * y + 4) * cyChar / 4
                                          (x + 41) * cxChar + 1, (2 * y
}

void ErrorMessage (HWND hwnd, TCHAR * szError, TCHAR * szTit
```

```
{
        wsprintf (szBuffer, szError,
                                (LPSTR) (szTitleName [0] ? szTitleNam
        MessageBeep (MB_ICONEXCLAMATION) ;
        MessageBox (hwnd, szBuffer, szAppName, MB_OK | MB_IC
}

void DoCaption (HWND hwnd, TCHAR * szTitleName)
{
        wsprintf (szBuffer, TEXT ("MIDI Drum Machine - %s"),
                                    (LPSTR) (szTitleName [0] ? szTitleN
        SetWindowText (hwnd, szBuffer) ;
}

int AskAboutSave (HWND hwnd, TCHAR * szTitleName)
{
        int iReturn ;
        wsprintf (szBuffer, TEXT ("Save current changes in %s?"),
                                    (LPSTR) (szTitleName [0] ? szTitle
```

```
        iReturn = MessageBox (hwnd, szBuffer, szAppName,
                              MB_YESNOCANCEL | MB_ICONQUESTION) ;

        if (iReturn == IDYES)
                    if (!SendMessage (hwnd, WM_COMMAND, IDM_F
                                         iReturn = IDCANCEL ;
        return iReturn ;
}
```

DRUMTIME.H

```
/*------------------------------------------------------------------------
  DRUMTIME.H Header File for Time Functions for DRUM Program
------------------------------------------------------------------------*/

#define NUM_PERC                                47
#define WM_USER_NOTIFY            (WM_USER + 1)
#define WM_USER_FINISHED           (WM_USER + 2)
#define WM_USER_ERROR                   (WM_USER + 3)

#pragma pack(push, 2)
typedef struct
```

```
{
        short iMsecPerBeat ;
        short iVelocity ;
        short iNumBeats ;
        DWORD dwSeqPerc [NUM_PERC] ;
        DWORD dwSeqPian [NUM_PERC] ;
}
DRUM, * PDRUM ;
#pragma pack(pop)
void DrumSetParams                      (PDRUM) ;
BOOL DrumBeginSequence          (HWND)  ;
void DrumEndSequence                (BOOL)  ;
```

DRUMTIME.C

```
/*------------------------------------------------------------------------
  DRUMFILE.C --Timer Routines for DRUM
                                        (c) Charles Petzold, 1998
------------------------------------------------------------------------*/
#include <windows.h>
```

```
#include "drumtime.h"

#define minmax(a,x,b) (min (max (x, a), b))
#define TIMER_RES   5
void CALLBACK DrumTimerFunc (UINT, UINT, DWORD, DWORD, D
BOOL                    bSequenceGoing, bEndSequence ;
DRUM                     drum ;
HMIDIOUT           hMidiOut ;
HWND                     hwndNotify ;
int                      iIndex ;
UINT                     uTimerRes, uTimerID ;

DWORD MidiOutMessage (     HMIDIOUT hMidi, int iStatus, int iCh
                                                        int iData1, int
{
   DWORD dwMessage ;
         dwMessage = iStatus | iChannel | (iData1 << 8) | (iData2
         return midiOutShortMsg (hMidi, dwMessage) ;
}
```

```
void DrumSetParams (PDRUM pdrum)
{
   CopyMemory (&drum, pdrum, sizeof (DRUM)) ;
}

BOOL DrumBeginSequence (HWND hwnd)
{
        TIMECAPS tc ;
        hwndNotify = hwnd ;                     // Save window hand
        DrumEndSequence (TRUE) ;     // Stop current sequence if

                    // Open the MIDI Mapper output port

        if (midiOutOpen (&hMidiOut, MIDIMAPPER, 0, 0, 0))
                    return FALSE ;
              // Send Program Change messages for channels 9 an
        MidiOutMessage (hMidiOut, 0xC0, 9, 0, 0) ;
        MidiOutMessage (hMidiOut, 0xC0, 0, 0, 0) ;
```

```
            // Begin sequence by setting a timer event
timeGetDevCaps (&tc, sizeof (TIMECAPS)) ;
        uTimerRes = minmax (tc.wPeriodMin, TIMER_RES, tc.wPer
        timeBeginPeriod (uTimerRes) ;

        uTimerID = timeSetEvent(max ((UINT) uTimerRes, (UINT)
             uTimerRes, DrumTimerFunc, 0, TIME_ONESHOT) ;

        if (uTimerID == 0)
 {
             timeEndPeriod (uTimerRes) ;
             midiOutClose (hMidiOut) ;
             return FALSE ;
        }

        iIndex = -1 ;
        bEndSequence = FALSE ;
        bSequenceGoing = TRUE ;
```

```
        return TRUE ;
}

void DrumEndSequence (BOOL bRightAway)
{
        if (bRightAway)
       {
              if (bSequenceGoing)
              {
                                    // stop the timer
                           if (uTimerID)
                                    timeKillEvent (uTimerID) ;
                                    timeEndPeriod (uTimerRes) ;

    // turn off all notes
                                   MidiOutMessage (hMidiOut, 0xB0,
                                MidiOutMessage (hMidiOut, 0xB0, 0, 1
   // close the MIDI port midiOutClose (hMidiOut) ; bSequenceGo
                        }
```

```
        }
        else
        bEndSequence = TRUE ;
}

void CALLBACK DrumTimerFunc (      UINT  uID, UINT uMsg, DWO
                                DWORD dw1, DWORD dw2)
{
   static DWORD dwSeqPercLast [NUM_PERC], dwSeqPianLast [N
   int                              i ;

                    // Note Off messages for channels 9 and 0

        if (iIndex != -1)
        {
              for (i = 0 ; i < NUM_PERC ; i++)
        {
                              if (dwSeqPercLast[i] & 1 << iIndex)
                                                MidiOutMessage (hMidiOut, 0x80,
```

```
                    if (dwSeqPianLast[i] & 1 << iIndex)
                        MidiOutMessage (hMidiOut, 0x80,
            }
}

            // Increment index and notify window to advance bou
        iIndex = (iIndex + 1) % drum.iNumBeats ;
        PostMessage (hwndNotify, WM_USER_NOTIFY, iIndex, tim

            // Check if ending the sequence
        if (bEndSequence && iIndex == 0)
        {
            PostMessage (hwndNotify, WM_USER_FINISHED, 0, 0L
            return ;
        }

            // Note On messages for channels 9 and 0
        for (i = 0 ; i < NUM_PERC ; i++)
```

```
        {
            if (drum.dwSeqPerc[i] & 1 << iIndex)
                        MidiOutMessage (hMidiOut, 0x90, 9, i
            if (drum.dwSeqPian[i] & 1 << iIndex)
                        MidiOutMessage (hMidiOut, 0x90, 0, i
                    dwSeqPercLast[i] = drum.dwSeqPerc[i] ;
                    dwSeqPianLast[i] = drum.dwSeqPian[i] ;
            }
                    // Set a new timer event
        uTimerID = timeSetEvent (max ((int) uTimerRes, drum.iM
            uTimerRes, DrumTimerFunc, 0, TIME_ONESHOT) ;
        if (uTimerID == 0)
        {
            PostMessage (hwndNotify, WM_USER_ERROR, 0, 0) ;
        }
}
```

DRUMFILE.H

```
/*------------------------------------------------------------------------
```

```
  DRUMFILE.H Header File for File I/O Routines for DRUM

-----------------------------------------------------------------------*/

BOOL            DrumFileOpenDlg      (HWND, TCHAR *, TCHAR *)

BOOL            DrumFileSaveDlg      (HWND, TCHAR *, TCHAR *)

TCHAR      *     DrumFileWrite              (DRUM *, TCHAR *) ;

TCHAR      *     DrumFileRead               (DRUM *, TCHAR *) ;
```

```
DRUMFILE.C

/*-----------------------------------------------------------------------

  DRUMFILE.C --        File I/O Routines for DRUM

                                                (c) Charles Petzold, 199

-----------------------------------------------------------------------*/

#include <windows.h>

#include <commdlg.h>

#include "drumtime.h"

#include "drumfile.h"

OPENFILENAME ofn = { sizeof (OPENFILENAME) } ;

TCHAR * szFilter[] = {     TEXT ("Drum Files (*.DRM)"),

                      TEXT ("*.drm"), TEXT ("") } ;
```

```
TCHAR szDrumID       []= TEXT ("DRUM") ;
TCHAR szListID        []    = TEXT ("LIST") ;
TCHAR szInfoID        []    = TEXT ("INFO") ;
TCHAR szSoftID        []    = TEXT ("ISFT") ;
TCHAR szDateID        []    = TEXT ("ISCD") ;
TCHAR szFmtID     []    = TEXT ("fmt ") ;
TCHAR szDataID        []    = TEXT ("data") ;
char  szSoftware     []= "DRUM by Charles Petzold, Programming

TCHAR szErrorNoCreate     []    =     TEXT   ("File %s could not b
TCHAR szErrorCannotWrite        []    =     TEXT   ("File %s coul
TCHAR szErrorNotFound    []    =     TEXT   ("File %s not found c
TCHAR szErrorNotDrum      []    =     TEXT   ("File %s is not a sta
TCHAR szErrorUnsupported        []    =     TEXT   ("File %s is n
TCHAR szErrorCannotRead         []    =     TEXT   ("File %s can

BOOL DrumFileOpenDlg (HWND hwnd, TCHAR * szFileName, TCH
{
          ofn.hwndOwner                  = hwnd ;
```

```
               ofn.lpstrFilter                 = szFilter [0] ;
               ofn.lpstrFile                   = szFileName ;
               ofn.nMaxFile                    = MAX_PATH ;
               ofn.lpstrFileTitle              = szTitleName ;
               ofn.nMaxFileTitle               = MAX_PATH ;
               ofn.Flags                               = OFN_CRE
               ofn.lpstrDefExt                 = TEXT ("drm") ;

          return GetOpenFileName (&ofn) ;
}

BOOL DrumFileSaveDlg (     HWND hwnd, TCHAR * szFileName,
                                        TCHAR * szTitleName)
{
               ofn.hwndOwner                   = hwnd ;
               ofn.lpstrFilter                 = szFilter [0] ;
               ofn.lpstrFile                   = szFileName ;
               ofn.nMaxFile                    = MAX_PATH ;
               ofn.lpstrFileTitle              = szTitleName ;
```

```
        ofn.nMaxFileTitle       = MAX_PATH ;
        ofn.Flags               = OFN_OVE
        ofn.lpstrDefExt         = TEXT ("drm") ;

    return GetSaveFileName (&ofn) ;
}

TCHAR * DrumFileWrite (DRUM * pdrum, TCHAR * szFileName)
{
    char            szDateBuf [16] ;
    HMMIO           hmmio ;
    int             iFormat = 2 ;
    MMCKINFO        mmckinfo [3] ;
    SYSTEMTIME      st ;
    WORD            wError = 0 ;

    memset (mmckinfo, 0, 3 * sizeof (MMCKINFO)) ;
                    // Recreate the file for writing
  if ((hmmio = mmioOpen (szFileName, NULL,
```

```
                    MMIO_CREATE | MMIO_WRITE | MMIO_ALLO
            return szErrorNoCreate ;
                    // Create a "RIFF" chunk with a "CPDR" typ
mmckinfo[0].fccType = mmioStringToFOURCC (szDrumID,
wError |= mmioCreateChunk (hmmio, &mmckinfo[0], MM
                    // Create "LIST" sub-chunk with an "INFO" t
mmckinfo[1].fccType = mmioStringToFOURCC (szInfoID, 0
wError |= mmioCreateChunk (hmmio, &mmckinfo[1], MM
                    // Create "ISFT" sub-sub-chunk
mmckinfo[2].ckid = mmioStringToFOURCC (szSoftID, 0) ;
wError |= mmioCreateChunk (hmmio, &mmckinfo[2], 0) ;
wError |= (mmioWrite (hmmio, szSoftware,     sizeof (szSo
              sizeof (szSoftware)) ;
wError |= mmioAscend (hmmio, &mmckinfo[2], 0) ;
                        // Create a time string
GetLocalTime (&st) ;
wsprintfA (szDateBuf, "%04d-%02d-%02d", st.wYear, st.w
                        // Create "ISCD" sub-sub-chunk
```

```
        mmckinfo[2].ckid = mmioStringToFOURCC (szDateID, 0) ;
        wError |= mmioCreateChunk (hmmio, &mmckinfo[2], 0) ;
        wError |= (mmioWrite (hmmio, szDateBuf, (strlen (szDate
                                                (int) (strlen (szDat
        wError |= mmioAscend (hmmio, &mmckinfo[2], 0) ;
        wError |= mmioAscend (hmmio, &mmckinfo[1], 0) ;


                            // Create "fmt " sub-chunk
        mmckinfo[1].ckid = mmioStringToFOURCC (szFmtID, 0) ;
        wError |= mmioCreateChunk (hmmio, &mmckinfo[1], 0) ;
        wError |= (mmioWrite (hmmio, (PSTR) &iFormat,       size
             sizeof (int)) ;
        wError |= mmioAscend (hmmio, &mmckinfo[1], 0) ;
                            // Create the "data" sub-chunk
        mmckinfo[1].ckid = mmioStringToFOURCC (szDataID, 0) ;
        wError |= mmioCreateChunk (hmmio, &mmckinfo[1], 0) ;
        wError |= (mmioWrite (hmmio, (PSTR) pdrum, sizeof (DRU
             sizeof (DRUM)) ;
        wError |= mmioAscend (hmmio, &mmckinfo[1], 0) ;
```

```
        wError |= mmioAscend (hmmio, &mmckinfo[0], 0) ;


                                // Clean up and return
  wError |= mmioClose (hmmio, 0) ;
        if (wError)
        {
              mmioOpen (szFileName, NULL, MMIO_DELETE) ;
              return szErrorCannotWrite ;
        }
        return NULL ;
}

TCHAR * DrumFileRead (DRUM * pdrum, TCHAR * szFileName)
{
        DRUM                    drum ;
        HMMIO                    hmmio ;
        int                     i, iFormat ;
        MMCKINFO            mmckinfo [3] ;
```

```
        ZeroMemory    (mmckinfo, 2 * sizeof (MMCKINFO)) ;

                        // Open the file

        if ((hmmio = mmioOpen (szFileName, NULL, MMIO_READ)
             return szErrorNotFound ;
                        // Locate a "RIFF" chunk with a "DRUM" for
        mmckinfo[0].ckid = mmioStringToFOURCC (szDrumID, 0)
        if (mmioDescend (hmmio, &mmckinfo[0], NULL, MMIO_FI
        {
             mmioClose (hmmio, 0) ;
        return szErrorNotDrum ;
 }

             // Locate, read, and verify the "fmt " sub-chunk
        mmckinfo[1].ckid = mmioStringToFOURCC (szFmtID, 0) ;
if (mmioDescend (hmmio, &mmckinfo[1], &mmckinfo[0], MMIO
        {
```

```
            mmioClose (hmmio, 0) ;
            return szErrorNotDrum ;
        }

if (mmckinfo[1].cksize != sizeof (int))
        {
            mmioClose (hmmio, 0) ;
        return szErrorUnsupported ;
  }

        if (mmioRead (hmmio, (PSTR) &iFormat, sizeof (int)) != si
  {
            mmioClose (hmmio, 0) ;
            return szErrorCannotRead ;
  }

        if (iFormat != 1 && iFormat != 2)
  {
            mmioClose (hmmio, 0) ;
```

```
            return szErrorUnsupported ;
}


            // Go to end of "fmt " sub-chunk
      mmioAscend (hmmio, &mmckinfo[1], 0) ;
            // Locate, read, and verify the "data" sub-chunk
      mmckinfo[1].ckid = mmioStringToFOURCC (szDataID, 0) ;
      if (mmioDescend (hmmio, &mmckinfo[1], &mmckinfo[0],
{
            mmioClose (hmmio, 0) ;
            return szErrorNotDrum ;
}


      if (mmckinfo[1].cksize != sizeof (DRUM))
      {
            mmioClose (hmmio, 0) ;
            return szErrorUnsupported ;
}
```

```
        if (mmioRead (hmmio, (LPSTR) &drum, sizeof (DRUM)) !=
        {
                mmioClose (hmmio, 0) ;
                return szErrorCannotRead ;
}


                        // Close the file
        mmioClose (hmmio, 0) ;
                // Convert format 1 to format 2 and copy the DRUM s
        if (iFormat == 1)
        {
                for (i = 0 ; i < NUM_PERC ; i++)
                {
                                drum.dwSeqPerc [i] = drum.dwSeqPian [i]
                                drum.dwSeqPian [i] = 0 ;
                }
        }
```

```
            memcpy (pdrum, &drum, sizeof (DRUM)) ;
            return NULL ;
}
```

```
DRUM.RC
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
//////////////////////////////////////////////////////////////////////////////
// Menu
DRUM        MENU DISCARDABLE
BEGIN
  POPUP "&File"
  BEGIN
    MENUITEM "&New",        IDM_FILE_NEW
     MENUITEM "&Open...",    IDM_FILE_OPEN
     MENUITEM "&Save",         IDM_FILE_SAVE
     MENUITEM "Save &As...", IDM_FILE_SAVE_AS
     MENUITEM SEPARATOR
```

```
    MENUITEM "E&xit",       IDM_APP_EXIT
  END
  POPUP "&Sequence"
  BEGIN
    MENUITEM "&Running",    IDM_SEQUENCE_RUNNING
    MENUITEM "&Stopped",    IDM_SEQUENCE_STOPPED
    , CHECKED
  END
  POPUP "&Help"
  BEGIN
    MENUITEM "&About...",   IDM_APP_ABOUT
  END
END

//////////////////////////////////////////////////////////////////////////
// Icon
DRUM         ICON         DISCARDABLE         "drum.ico"

//////////////////////////////////////////////////////////////////////////
```

```
// Dialog

ABOUTBOX DIALOG DISCARDABLE  20, 20, 160, 164

STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSM

CAPTION "Dialog"

FONT 8, "MS Sans Serif"

BEGIN

DEFPUSHBUTTON  "OK",IDOK,54,143,50,14

ICON        "DRUM",IDC_STATIC,8,8,21,20

CTEXT       "DRUM",IDC_STATIC,34,12,90,8

CTEXT      "MIDI Drum Machine",IDC_STATIC,7,36,144,8

CONTROL     "",IDC_STATIC,"Static",SS_BLACKFRAME,8,88,144,46

LTEXT      "Left Button:\t\tDrum sounds",IDC_STATIC,12,92,136,8

LTEXT      "Right Button:\t\tPiano sounds",IDC_STATIC,12,102,13

LTEXT            "Horizontal Scroll:\t\tVelocity",IDC_STATIC,12,112,

LTEXT            "Vertical Scroll:\t\tTempo",IDC_STATIC,12,122,136

CTEXT      "Copyright (c) Charles Petzold, 1998",IDC_STATIC,8,48

CTEXT      """Programming Windows,"" 5th Edition",IDC_STATIC,8

END
```

RESOURCE.H

```
// Microsoft Developer Studio generated include file.
// Used by Drum.rc
#define IDM_FILE_NEW 4
#define IDM_FILE_OPEN 4
#define IDM_FILE_SAVE 4
#define IDM_FILE_SAVE_AS
#define IDM_APP_EXIT 4(
#define IDM_SEQUENCE_RUNNING
#define IDM_SEQUENCE_STOPPED
#define IDM_APP_ABOUT
```

DRUM473232324/441

SequenceRunningMIDI Mapper

4:|3

MIDI Note On1101001/10

File.DRMRIFFHelpAbout

SequenceStoppedMIDI N

time

DRUM.CDRUMTIME

WindowsBACHTOCCWindowsPCMIDIAPI7time
DRUMTIME6callback

1001090110

```
timeGetDevCaps (&timecaps, uSize) ;
```

TIMECAPSTIMECAPSwPeriodMinwPeriodMax
timeGetDevCapswPeriodMin1wPeriodMax65535

```
timeBeginPeriod (uResolution) ;
```

TIMECAPStimeBeginPeriodtimeEndPeriodtimeEndPeriod

```
idTimer = timeSetEvent (   uDelay, uResolution, CallBackFunc, d
```

idTimer0WindowsuDelayCallBackFuncuResolution
uResolutiontimeBeginPerioddwDataCallBackFunc
TIME_ONESHOTTIME_PERIODICuDelayCallBackFuncuDelay
CallBackFunc

CallBackFunc

```
timeKillEvent (idTimer) ;
```

CallBackFunc

```
timeEndPeriod (wResolution) ;
```

timeBeginPeriod

time

```
dwSysTime = timeGetTime () ;
```

Windows

```
timeGetSystemTime (&mmtime, uSize) ;
```

MMTIMEMMTIMEtimeGetSystemTime

CallbackWindowsCallbackPostMessagePostMessagetimeSetEvent
timeKillEventtimeGetTimetimeGetSystemTimeMIDImidiOutShortMsg
midiOutLongMsgOutputDebugStr

MIDIPostMessagecallback

CallbacktimeSetEventIDtimeSetEventdwData

DRUM.CDRUMTIME.CDrumSetParamsDRUM.DRM
DrumSetParamsDRUMDRUMTIME.H473232
DRUM.CDRUMDrumSetParamsDrumSetParams

DRUMDRUMTIMEDrumBeginSequenceDrumBeginSequence
MIDI MapperProgram ChangeMIDI0909MIDI10
DrumBeginSequencetimeGetDevCapstimeBeginPeriodTIMER_RES5
minmaxtimeGetDevCaps

timeSetEventcallbackDrumTimerFuncTIME_ONESHOT
DRUMTIMEtimeSetEventDrumTimerFunc

DrumTimerFunccallbackDRUMTIME.CDRUMTIME.CiIndex

CallbackMIDI Note OffiIndex-1

iIndexWM_USER_NOTIFYDRUMwParamiIndexDRUM.C
WndProc

DrumTimerFuncNote On09timeSetEvent

DRUMDrumEndSequenceTRUEFALSETRUEDrumEndSequence
timeEndPeriodMIDIall notes offMIDIDRUMTRUE
DrumEndSequence

DRUMSequenceStopFALSEDrumEndSequence
DrumEndSequencebEndSequenceNULLbEndSequenceTRUE0
DrumTimerFuncWM_USER_FINISHEDWndProcWndProcTRUE
DrumEndSequenceMIDI

RIFFI/O

DRUMDRUMRIFFResource
RIFFmmio/

.WAVRIFF4ASCII32

WindowsRIFF4ASCII43208

RIFFLIST4ASCIILISTRIFF4ASCIIRIFFRIFFLIST

RIFFRIFFRIFFRIFF832

API16mmioRIFFDRUMFILE.CDRUM

mmiommioOpenmmioCreateChunkMMCKINFOmmioWrite
mmioAscendmmioAscendMMCKINFOmmioCreateChunk
MMCKINFOdwDataOffsetmmioAscend2mmioAscend

RIFFmmioAscendMMCKINFODRUMDRUMFILE.C
DrumFileWriteMMCKINFOmmckinfo[0]mmckinfo[1]mmckinfo[2]
mmioCreateChunkmmckinfo[0]DRUMRIFFmmioCreateChunk
mmckinfo[1]INFOLIST

mmioCreateChunkmmckinfo[2]ISFTmmioWriteszSoftware
mmioAscentmmckinfo[2]LISTmmioCreateChunkISCDcreation
datammckinfo[2]mmioWritemmckinfo[2]mmioAscend
LISTLISTmmioAscendmmckinfo[1]LIST

fmtdatammioCreateChunkmmckinfo[1]mmioWritemmckinfo[1]
mmioAscendRIFFmmckinfo[0]mmioAscendmmioClose

mmioAscend1

mmioOpenwErrormmioCreateChunkmmioWritemmioAscend
mmioClosemmioOpenMMIO_DELETE

RIFFRIFFmmioReadmmioWritemmioDescendmmioCreateChunk
descendRIFFLISTmmioDescend
mmioAscend

DRUM1992PC MagazineWindowsMIDI1
DRUM 2DrumFileRead

Internet

InternetInternetID

InternetWorld

WindowsInternetMicrosoft
(Winsock) APIWindows InternetWinInetAPIFTPFile Transfer

Windows Sockets

SocketUniversity of CaliforniaBerkeleyUNIXAPIBerkeley
interface

SocketsTCP/IP

SocketInternet/TCP/IPTransmission
ProtocolIPInternet ProtocolTCP/IPdatagram
TCPTransmission Control ProtocolIP

TCP/IPIPIP4InternetIP255209.86.105.231

SocketTCP/IPSocketTCP/IPSocketIP

Time ProtocolInternetPC

National Institute of Standards and TechnologyNation
StandardsInternet
bldrdocBoulderColoradoNIST TimeFrequency Division

NIST Network Time Service
http://www.bldrdoc.gov/timefreq/service/nts.htmNISTtime-

a.timefreq.bldrdoc.govIP132.163.135.130

Internet NISTPC Magazine
http://www.zdnet.com/pcmag/pctech/content/16/20/ut1620.001.html
Windows Telephony API

InternetRequest for CommentRFCInternetDay
RFC-867ASCIIASCIIRFC-86832190011UTC
Coordinated Universal TimeGreenwich
Network Time ProtocolRFC-1305

SocketPCRFC-868RFC-868TCP

1. 37
2. 32
3.

Socket

NETTIME

Windows Sockets APIWinSockBerkeley Sockets APIUNIX
WindowsWindowsBerkeley SocketWSAWinSock
/Platform SDK/Networking and Distributed Services/Windows Sockets
Version 2

NETTIME23-1WinSock API

23-1 NETTIME

```
NETTIME.C

/*------------------------------------------------------------------------
  NETTIME.C -- Sets System Clock from Internet Services
                                        (c) Charles Petzold, 1998
```

```
------------------------------------------------------------------------*/
#include <windows.h>
#include "resource.h"

#define     WM_SOCKET_NOTIFY (WM_USER + 1)
#define     ID_TIMER         1

LRESULT     CALLBACK WndProc   (HWND, UINT, WPARAM, LPARA
BOOL             CALLBACK MainDlg   (HWND, UINT, WPARAM, LPA
BOOL             CALLBACK ServerDlg          (HWND, UINT, WPAR

void        ChangeSystemTime           (HWND hwndEdit, ULONG
void        FormatUpdatedTime          (HWND hwndEdit, SYSTEM
                        SYSTEMTIME * pstNew) ;
void EditPrintf (HWND hwndEdit, TCHAR * szFormat, ...) ;
HINSTANCE hInst ;
HWND             hwndModeless ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                PSTR szCmdLine, int iCmdShow)
```

```
{
        static TCHAR szAppName[] = TEXT ("NetTime") ;
        HWND                    hwnd ;
        MSG                     msg ;
        RECT                    rect ;
        WNDCLASS                wndclass ;

        hInst = hInstance ;
        wndclass.style                  = 0 ;
        wndclass.lpfnWndProc                = WndProc ;
        wndclass.cbClsExtra                 = 0 ;
        wndclass.cbWndExtra                 = 0 ;
        wndclass.hInstance                  = hInstance ;
        wndclass.hIcon                  = LoadIcon (NULL, IDI_
        wndclass.hCursor                    = NULL ;
        wndclass.hbrBackground          = NULL ;
        wndclass.lpszMenuName           = NULL ;
        wndclass.lpszClassName          = szAppName ;

       if (!RegisterClass (&wndclass))
```

```
     {
   MessageBox (NULL, TEXT ("This program requires Windows
              szAppName, MB_ICONERROR) ;
            return 0 ;
}


     hwnd = CreateWindow (szAppName, TEXT ("Set System C
                WS_OVERLAPPED | WS_CAPTION | WS_SYSMENU
                 WS_BORDER | WS_MINIMIZEBOX,
                           CW_USEDEFAULT, CW_USEDEFAULT,
                CW_USEDEFAULT, CW_USEDEFAULT,
               NULL, NULL, hInstance, NULL) ;

     // Create the modeless dialog box to go on top of the win
     hwndModeless = CreateDialog (hInstance, szAppName, h
     // Size the main parent window to the size of the dialog b
     //   Show both windows.

     GetWindowRect (hwndModeless, &rect) ;
```

```
        AdjustWindowRect (&rect, WS_CAPTION | WS_BORDER, FA

        SetWindowPos (hwnd, NULL, 0, 0, rect.right - rect.left,
                        rect.bottom - rect.top, SWP_NOMOVE) ;

        ShowWindow (hwndModeless, SW_SHOW) ;
        ShowWindow (hwnd, iCmdShow) ;
        UpdateWindow (hwnd) ;

        // Normal message loop when a modeless dialog box is us
        while (GetMessage (&msg, NULL, 0, 0))
        {
             if (hwndModeless == 0 || !IsDialogMessage (hwndMo
             {
                        TranslateMessage (&msg) ;
                        DispatchMessage (&msg) ;
             }
        }
        return msg.wParam ;
}
```

```
LRESULT CALLBACK WndProc ( HWND hwnd, UINT message, WPA
{
        switch (message)
   {
        case   WM_SETFOCUS:
                    SetFocus (hwndModeless) ;
                    return 0 ;

        case   WM_DESTROY:
                    PostQuitMessage (0) ;
                   return 0 ;
   }
        return DefWindowProc (hwnd, message, wParam, lParam)
}

BOOL CALLBACK MainDlg (    HWND hwnd, UINT message, WPAR
                                                LPARAM lParam)
{
        static char          szIPAddr[32] = { "132.163.135.130" } ;
```

```
        static HWND          hwndButton, hwndEdit ;
        static SOCKET        sock ;
        static struct        sockaddr_in sa ;
        static TCHAR         szOKLabel[32] ;
        int                              iError, iSize ;
        unsigned long        ulTime ;
        WORD                             wEvent, wError ;
        WSADATA                          WSAData ;

        switch (message)
        {
        case   WM_INITDIALOG:
                    hwndButton = GetDlgItem (hwnd, IDOK) ;
                   hwndEdit = GetDlgItem (hwnd, IDC_TEXTOUT) ;
                    return TRUE ;

        case   WM_COMMAND:
                   switch (LOWORD (wParam))
             {
                    case   IDC_SERVER:
```

```
        DialogBoxParam (hInst, TEXT ("Servers"), hwnd,
        return TRUE ;

        case   IDOK:
// Call "WSAStartup" and display description text

        if (iError = WSAStartup (MAKEWORD(2,0), &WSA
        {
EditPrintf (hwndEdit, TEXT ("Startup error #%i.\r\n"),
return TRUE ;
              }
        EditPrintf (hwndEdit, TEXT ("Started up %hs\r\n"
            WSAData.szDescription);

// Call "socket"

                          sock = socket (AF_INET, SOCK_ST

                        if (sock == INVALID_SOCKET)
                          {
EditPrintf (hwndEdit, TEXT ("Socket creation error #%
```

```
  WSACleanup () ;
   return TRUE ;
    }
EditPrintf (hwndEdit, TEXT ("Socket %i created.\r\n"), so

// Call "WSAAsyncSelect"

if (SOCKET_ERROR == WSAAsyncSelect (sock, hwnd, W
                                {
EditPrintf ( hwndEdit, TEXT ("WSAAsyncSelect error #%i.
   closesocket (sock) ;
   WSACleanup () ;
   return TRUE ;
    }

   // Call "connect" with IP address and time-server port

    sa.sin_family           = AF_INET ;
    sa.sin_port             = htons (IPPORT_TIMESERVER) ;
    sa.sin_addr.S_un.S_addr = inet_addr (szIPAddr) ;
```

```
    connect(sock, (SOCKADDR *) &sa, sizeof (sa)) ;

// "connect" will return SOCKET_ERROR because even if
// succeeds, it will require blocking. The following only
// reports unexpected errors.

if (WSAEWOULDBLOCK != (iError = WSAGetLastError ())
    {
EditPrintf (hwndEdit, TEXT ("Connect error #%i.\r\n"), iE
          closesocket (sock) ;
          WSACleanup () ;
         return TRUE ;
}
EditPrintf (hwndEdit, TEXT ("Connecting to %hs..."), szIP

// The result of the "connect" call will be reported
// through the WM_SOCKET_NOTIFY message.
// Set timer and change the button to "Cancel"

SetTimer (hwnd, ID_TIMER, 1000, NULL) ;
```

```
 GetWindowText (hwndButton, szOKLabel, sizeof (szOKLa
 SetWindowText (hwndButton, TEXT ("Cancel")) ;
 SetWindowLong (hwndButton, GWL_ID, IDCANCEL) ;
return TRUE ;

            case   IDCANCEL:
                              closesocket (sock) ;
                              sock = 0 ;
                              WSACleanup () ;
                              SetWindowText (hwndButton, szO
                             SetWindowLong (hwndButton, GW

                              KillTimer (hwnd, ID_TIMER) ;
                              EditPrintf (hwndEdit, TEXT ("\r\nS
                              return TRUE ;

            case   IDC_CLOSE:
                        if (sock)
            SendMessage (hwnd, WM_COMMAND, IDCANCE
```

```
                                DestroyWindow (GetParent (hwnd
                                return TRUE ;
                    }
                    return FALSE ;

          case   WM_TIMER:
                    EditPrintf (hwndEdit, TEXT (".")) ;
                    return TRUE ;

          case   WM_SOCKET_NOTIFY:
                    wEvent = WSAGETSELECTEVENT (lParam) ;   // i
                    wError = WSAGETSELECTERROR (lParam) ;   // ie

                              // Process two events specified in WSA

                    switch (wEvent)
                    {
     // This event occurs as a result of the "connect" call

                    case   FD_CONNECT:
                                EditPrintf (hwndEdit, TEXT ("\r\n")
```

```
                                    if (wError)
                                      {
 EditPrintf (hwndEdit, TEXT ("Connect error #%i."), wError) ;
 SendMessage (hwnd, WM_COMMAND, IDCANCEL, 0) ;
return TRUE ;
     }
  EditPrintf (hwndEdit, TEXT ("Connected to %hs.\r\n"), szIPAd

   // Try to receive data. The call will generate an error
   // of WSAEWOULDBLOCK and an event of FD_READ

   recv (sock, (char *) &ulTime, 4, MSG_PEEK) ;
   EditPrintf (hwndEdit, TEXT ("Waiting to receive...")) ;
  return TRUE ;

   // This even occurs when the "recv" call can be made

            case   FD_READ:
                                    KillTimer (hwnd, ID_TIMER) ;
                                    EditPrintf (hwndEdit, TEXT ("\r\n")
```

```
                                    if (wError)
                                    {
EditPrintf (hwndEdit, TEXT ("FD_READ error #%i."), wError) ;
SendMessage (hwnd, WM_COMMAND, IDCANCEL, 0) ;
return TRUE ;
}
                                    // Get the time and swap the byte

  iSize = recv (sock, (char *) &ulTime, 4, 0) ;
  ulTime = ntohl (ulTime) ;
  EditPrintf (hwndEdit,
  TEXT ("Received current time of %u seconds ")
  TEXT ("since Jan. 1 1900.\r\n"), ulTime) ;

  // Change the system time

                                     ChangeSystemTime (hwndEdit, u
                                    SendMessage (hwnd, WM_COMMA
                                     return TRUE ;
```

```
                }
                return FALSE ;
        }
        return FALSE ;
}

BOOL CALLBACK ServerDlg (  HWND hwnd, UINT message, WPAR
{
        static char * szServer ;
        static WORD          wServer = IDC_SERVER1 ;
        char                                szLabel [64] ;

        switch (message)
        {
        case   WM_INITDIALOG:
                   szServer = (char *) lParam ;
                    CheckRadioButton (hwnd, IDC_SERVER1, IDC_SE
                    return TRUE ;

        case   WM_COMMAND:
```

```
                switch (LOWORD (wParam))
                {
                case   IDC_SERVER1:
                case   IDC_SERVER2:
                case   IDC_SERVER3:
                case   IDC_SERVER4:
                case   IDC_SERVER5:
                case   IDC_SERVER6:
                case   IDC_SERVER7:
                case   IDC_SERVER8:
                case   IDC_SERVER9:
                case   IDC_SERVER10:
                                wServer = LOWORD (wParam) ;
                                return TRUE ;

               case   IDOK:
                                GetDlgItemTextA (hwnd, wServer
                                strtok (szLabel, "(") ;
                                strcpy (szServer, strtok (NULL, ")'
```

```
                                EndDialog (hwnd, TRUE) ;
                                return TRUE ;

                        case   IDCANCEL:
                                EndDialog (hwnd, FALSE) ;
                                return TRUE ;
                        }
                        break ;
    }
        return FALSE ;
}

void ChangeSystemTime (HWND hwndEdit, ULONG ulTime)
{
        FILETIME                ftNew ;
        LARGE_INTEGER        li ;
        SYSTEMTIME              stOld, stNew ;

        GetLocalTime (&stOld) ;
        stNew.wYear                                = 1900 ;
```

```
        stNew.wMonth                                    = 1 ;
        stNew.wDay                                     = 1 ;
        stNew.wHour                                    = 0 ;
        stNew.wMinute                                   = 0 ;
        stNew.wSecond                                   = 0 ;
        stNew.wMilliseconds                 = 0 ;

        SystemTimeToFileTime (&stNew, &ftNew) ;
        li = * (LARGE_INTEGER *) &ftNew ;
        li.QuadPart += (LONGLONG) 10000000 * ulTime ;
        ftNew = * (FILETIME *) &li ;
        FileTimeToSystemTime (&ftNew, &stNew) ;

        if (SetSystemTime (&stNew))
        {
             GetLocalTime (&stNew) ;
              FormatUpdatedTime (hwndEdit, &stOld, &stNew) ;
}
        else
```

```
            EditPrintf (hwndEdit, TEXT ("Could NOT set new date
}

void FormatUpdatedTime (   HWND hwndEdit, SYSTEMTIME * pst
{
        TCHAR szDateOld [64], szTimeOld [64], szDateNew [64],
        GetDateFormat (LOCALE_USER_DEFAULT, LOCALE_NOUSE
                                        pstOld, NULL, szDateOld, siz
        GetTimeFormat (LOCALE_USER_DEFAULT, LOCALE_NOUSE
                TIME_NOTIMEMARKER | TIME_FORCE24HOURFORM
              pstOld, NULL, szTimeOld, sizeof (szTimeOld)) ;

        GetDateFormat (LOCALE_USER_DEFAULT, LOCALE_NOUSE
              pstNew, NULL, szDateNew, sizeof (szDateNew)) ;
        GetTimeFormat (LOCALE_USER_DEFAULT, LOCALE_NOUSE
                TIME_NOTIMEMARKER | TIME_FORCE24HOURFORM
              pstNew, NULL, szTimeNew, sizeof (szTimeNew)) ;

        EditPrintf (hwndEdit, TEXT ("System date and time succes
                        TEXT ("from\r\n\t%s, %s.%03i to\r\n\t%s, %
```

```
                szDateOld, szTimeOld, pstOld->wMilliseconds,
                szDateNew, szTimeNew, pstNew->wMilliseconds) ;
}

void EditPrintf (HWND hwndEdit, TCHAR * szFormat, ...)
{
        TCHAR        szBuffer [1024] ;
        va_listpArgList ;

        va_start (pArgList, szFormat) ;
        wvsprintf (szBuffer, szFormat, pArgList) ;
        va_end (pArgList) ;
        SendMessage (hwndEdit, EM_SETSEL, (WPARAM) -1, (LPAI
        SendMessage (hwndEdit, EM_REPLACESEL, FALSE, (LPARA
        SendMessage (hwn dEdit, EM_SCROLLCARET, 0, 0) ;
}
```

NETTIME.RC

```
//Microsoft Developer Studio generated resource script.
#include "resource.h"
```

```
#include "afxres.h"

//////////////////////////////////////////////////////////////////////////

// Dialog

SERVERS DIALOG DISCARDABLE  20, 20, 274, 202

STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSM

CAPTION "NIST Time Service Servers"

FONT 8, "MS Sans Serif"

BEGIN

  DEFPUSHBUTTON              "OK",IDOK,73,181,50,14

  PUSHBUTTON                   "Cancel",IDCANCEL,150,181,50

  CONTROL

            "time-a.timefreq.bldrdoc.gov (132.163.135.130) NIST

                  IDC_SERVER1,"Button",BS_AUTORADIOBUTTON,

  CONTROL

       "time-b.timefreq.bldrdoc.gov (132.163.135.131) NIST, Bou

             IDC_SERVER2,"Button",BS_AUTORADIOBUTTON,9,24,

  CONTROL

             "time-c.timefreq.bldrdoc.gov (132.163.135.132) Boul

                  IDC_SERVER3,"Button",BS_AUTORADIOBUTTON,
```

```
CONTROL
            "utcnist.colorado.edu (128.138.140.44) University of
                IDC_SERVER4,"Button",BS_AUTORADIOBUTTON,
CONTROL
                "time.nist.gov (192.43.244.18) NCAR, Boulder, C
               IDC_SERVER5,"Button",BS_AUTORADIOBUTTON,9
CONTROL
                "time-a.nist.gov (129.6.16.35) NIST, Gaithersbu
                IDC_SERVER6,"Button",BS_AUTORADIOBUTTON,
CONTROL
               "time-b.nist.gov (129.6.16.36) NIST, Gaithersbur
                IDC_SERVER7,"Button",BS_AUTORADIOBUTTON,
CONTROL
                "time-nw.nist.gov (131.107.1.10) Microsoft, Red
                IDC_SERVER8,"Button",BS_AUTORADIOBUTTON,
CONTROL
                "utcnist.reston.mci.net (204.70.131.13) MCI, Re
                IDC_SERVER9,"Button",BS_AUTORADIOBUTTON,
```

```
  CONTROL
                        "nist1.data.com (209.0.72.7) Datum, San J
                   IDC_SERVER10,"Button",BS_AUTORADIOBUTTON
END
NETTIME DIALOG DISCARDABLE  0, 0, 270, 150 STYLE WS_CHILD
BEGIN
  DEFPUSHBUTTON             "Set Correct Time",IDOK,95,129,8
  PUSHBUTTON                  "Close",IDC_CLOSE,183,129,80,
  PUSHBUTTON                  "Select Server...",IDC_SERVER,7
  EDITTEXT                      IDC_TEXTOUT,7,7,253,110,ES
                                      ES_READONLY | WS_VSC
END
```

```
RESOURCE.H
// Microsoft Developer Studio generated include file.
// Used by NetTime.rc
#define         IDC_TEXTOUT                                 1
#define         IDC_SERVER1                                 1
#define         IDC_SERVER2                                 1
```

```
#define         IDC_SERVER3                             10
#define         IDC_SERVER4                              1(
#define         IDC_SERVER5                              1(
#define         IDC_SERVER6                              1(
#define         IDC_SERVER7                              1(
#define         IDC_SERVER8                             10
#define         IDC_SERVER9                              1(
#define         IDC_SERVER10                              1
#define         IDC_SERVER                              10
#define         IDC_CLOSE
```

NETTIMENETTIME.RCNETTIMESelect
Set Correct TimeCloseClose

MainDlgszIPAddr132.163.135.130Select
SERVERSszIPAddrDialogBoxParamServer10NIST
ServerDlgIPszIPAddr

Set Correct TimeWM_COMMANDwParamIDOKMainDlgIDOK
Socket

Windows Sockets APIWindows

```
iError = WSAStartup (wVersion, &WSAData) ;
```

NETTIME0x02002.0WSADataWindows
szDescription

NETTIMEsocket

```
sock = socket (AF_INET, SOCK_STREAM, IPPROTO_TCP) ;
```

Internet4InternetTCPRFC-868socket
SOCKETSocket

NETTIMEWSAAsynchSelectWindowsSocketInternetWinSock
blockingWSAAsyncSelectWSAAsyncSelect

```
WSAAsyncSelect (sock, hwnd, message, iConditions) ;
```

NETTIMEWM_SOCKET_NOTIFYWSAAsyncSelect
FD_CONNECT　| FD_READ

NETTIMEWinSockconnectSocketSocketNETTIMETCP/IP

```
struct sockaddr_in
{
        short                   sin_family;
        u_short                 sin_port;
        struct in_addr sin_addr;
        char                    sin_zero[8];
} ;
```

in_addrInternet421

NETTIMEsin_familyAF_INETsin_portRFC-8683737
Internetbig endianIntellittle
network shortNETTIMEsockaddr_insin_port

```
htons (IPPORT_TIMESERVER)
```

WINSOCK2.H37NETTIMEinet_addrszIPAddrsin_addr

Windows 98connectWindowsInternetWindows
NTNETTIMEInternet

connectNETTIMEWSAAsyncSelectconnect
SOCKET_ERRORNETTIMEWSAGetLastError
WSAGetLastErrorWSAEWOULDBLOCKNETTIMESet
Correct TimeCancel1WM_TIMER

MainDlgWM_SOCKET_NOTIFYNETTIMEWSAAsyncSelectlParam
FD_CONNECTNETTIME9

NETTIMErecvreceive

```
recv (sock, (char *) &ulTime, 4, MSG_PEEK) ;
```

4ulTimeconnectrecv32recv
MSG_PEEK

connectrecvWM_SOCKET_NOTIFYFD_READNETTIMErecv
0ulTimeNETTIMEWM_COMMANDwParamIDCANCEL
closesocketWSACleanup

NETTIME32ulTime1990110:00 UTCntohlnet
IntelNETTIMEChangeSystemTime

ChangeSystemTimeSYSTEMTIME1900110SYSTEMTIME

SystemTimeToFileTimeFILETIMEFILETIME32DWORD64160111
100nanosecond

ChangeSystemTimeFILETIMELARGE_INTEGERunion6432__int6464
__int64MicrosoftANSI C160111190011100190011100
ulTime10,000,000

FileTimeToSystemTimeFILETIMESYSTEMTIMEUTCNETTIME
SetSystemTimeSetSystemTimeUTCGetLocalTime
FormatUpdatedTimeGetTimeFormatGetDateFormatASCII

Windows NTSetSystemTimeSetSystemTimeNETTIME

WinInet FTP

WinInetWindows InternetAPIInternetWorld
HTTPHypertext Transfer ProtocolFTPFile Transfer Proto
GopherWinInetWindowsWinInet
Intranet, Extranet Services/Internet Tools and Technologies/WinInet API

WinInet APIFTPFTPInternet
ftp://ftp.microsoft.comFTP
ftp://ftp.cpetzold.com/cpetzold.com/ProgWin/UpdDemoFTP

FTPWebFTPFTPUPDDEMOupdate

FTP API

WinInetWinInetWININET.HWININET.LIBMicrosoft
Project SettingsLinkWININET.DLL

WinInetUPDDEMOFTP

Windows Internet APIInternetOpenWinInetInternetOpenInternet
HINTERNETWinInet APIInternetCloseHandle

FTPInternetConnectInternetOpenInternetFTPFtpInternetConnect
FTPftp.cpetzold.comFTPNULLInternetConnectPC

InternetWindows　98FTPInternetCloseHandle

FtpWindowsI/OInternetFTP

```
fSuccess = FtpCreateDirectory          (hFtpSession, szDirectory
fSuccess = FtpRemoveDirectory          (hFtpSession, szDirector
fSuccess = FtpSetCurrentDirectory (hFtpSession, szDirectory) ;
fSuccess = FtpGetCurrentDirectory (hFtpSession, szDirectory, &
```

WindowsCreateDirectoryRemoveDirectorySetCurrentDirectory
GetCurrentDirectory

FTPFTPWindowsSetCurrentDirectory
GetCurrentDirectoryGetCurrentDirectoryMAX_PATH

FTP

```
fSuccess = FtpDeleteFile (hFtpSession, szFileName) ;
fSuccess = FtpRenameFile (hFtpSession, szOldName, szNewNam
```

FtpFindFirstFileFindFirstFileWIN32_FIND_DATA
InternetFindNextFileInternetCloseHandle

FtpFileOpenInternetReadFileInternetReadFileExInternetWrite
InternetSetFilePointerInternetCloseHandle

FtpGetFileFTPFtpFileOpenFileCreateInternetReadFile
WriteFileInternetCloseHandleCloseHandleFtpGetFile
FtpPutFileFTP

UPDDEMO23-2WinInet FTPFTP

23-2 UPDDEMO

```
UPDDEMO.C
/*------------------------------------------------------------------------
  UPDDEMO.C -- Demonstrates Anonymous FTP Access
                                        (c) Charles Petzold, 1998
------------------------------------------------------------------------*/
#include          <windows.h>
#include          <wininet.h>
#include          <process.h>
#include          "resource.h"

              // User-defined messages used in WndProc

#define     WM_USER_CHECKFILES    (WM_USER + 1)
#define     WM_USER_GETFILES            (WM_USER + 2)

              // Information for FTP download
```

```
#define     FTPSERVER     TEXT   ("ftp.cpetzold.com")
#define     DIRECTORY     TEXT   ("cpetzold.com/ProgWin/UpdDer
#define     TEMPLATE      TEXT   ("UD??????.TXT")

   // Structures used for storing filenames and contents
typedef struct
{
        TCHAR * szFilename ;
        char  * szContents ;
}
FILEINFO ;
typedef struct
{
        int                   iNum ;
        FILEINFO          info[1] ;
}
FILELIST ;
   // Structure used for second thread
typedef struct
```

```
{
        BOOL bContinue ;
        HWND hwnd ;
}
PARAMS ;
            // Declarations of all functions in program
LRESULT    CALLBACK          WndProc (HWND, UINT, WPARAM,
BOOL           CALLBACK          DlgProc (HWND, UINT, WPARA
VOID                                 FtpThread (PVOID) ;
VOID                                 ButtonSwitch (HWND, HW
FILELIST *                       GetFileList (VOID) ;
int                                  Compare (const FILEINFO *

            // A couple globals

HINSTANCE hInst ;
TCHAR                szAppName[] = TEXT ("UpdDemo") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevIns
                                        PSTR szCmdLine, int iCr
```

```
{
     HWND                    hwnd ;
     MSG                     msg ;
     WNDCLASS                wndclass ;

     hInst = hInstance ;
     wndclass.style                  = 0 ;
     wndclass.lpfnWndProc              = WndProc ;
     wndclass.cbClsExtra               = 0 ;
     wndclass.cbWndExtra               = 0 ;
     wndclass.hInstance                = hInstance ;
     wndclass.hIcon                    = LoadIcon (NULL, IDI_
     wndclass.hCursor                  = NULL ;
     wndclass.hbrBackground            = GetStockObject (WH
   wndclass.lpszMenuName             = NULL ;
     wndclass.lpszClassName            = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (  NULL, TEXT ("This program requires W
```

```
                                                    szAppName, MB_IC
          return 0 ;
}

     hwnd = CreateWindow ( szAppName, TEXT ("Update Dem
               WS_OVERLAPPEDWINDOW | WS_VSCROLL,
               CW_USEDEFAULT, CW_USEDEFAULT,
              CW_USEDEFAULT, CW_USEDEFAULT,
               NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

          // After window is displayed, check if the latest file ex
     SendMessage (hwnd, WM_USER_CHECKFILES, 0, 0) ;
     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
```

```
        }
        return msg.wParam ;
}

LRESULT CALLBACK WndProc ( HWND hwnd, UINT message, WPA
{
        static FILELIST *     plist ;
        static int                              cxClient, cyClient, cxChar,
        HDC                                             hdc ;
        int                                           i ;
        PAINTSTRUCT                             ps ;
        SCROLLINFO                            si ;
        SYSTEMTIME                             st ;
        TCHAR                                           szFilename [MAX

        switch (message)
   {
        case   WM_CREATE:
                    cxChar = LOWORD (GetDialogBaseUnits ()) ;
                    cyChar = HIWORD (GetDialogBaseUnits ()) ;
```

```
                return 0 ;

        case   WM_SIZE:
                cxClient            = LOWORD (lParam) ;
                cyClient            = HIWORD (lParam) ;

                si.cbSize     = sizeof (SCROLLINFO) ;
                si.fMask            = SIF_RANGE | SIF_PAGE ;
                si.nMin             = 0 ;
                si.nMax             = plist ? plist->iNum - 1 : 0 ;
                si.nPage            = cyClient / cyChar ;

                SetScrollInfo (hwnd, SB_VERT, &si, TRUE) ;
                return 0 ;

        case   WM_VSCROLL:
                si.cbSize     = sizeof (SCROLLINFO) ;
                si.fMask            = SIF_POS | SIF_RANGE | SIF_PA
                GetScrollInfo (hwnd, SB_VERT, &si) ;

                switch (LOWORD (wParam))
```

```
                {
                case SB_LINEDOWN: si.nPos += 1 ; break ;
                case SB_LINEUP:   si.nPos -= 1 ; break ;
                case SB_PAGEDOWN: si.nPos += si.nPage ; brea
                case SB_PAGEUP:   si.nPos -= si.nPage ; break ;
                case SB_THUMBPOSITION: si.nPos = HIWORD (w
                default:                  return 0 ;
                }
                si.fMask = SIF_POS ;
                SetScrollInfo (hwnd, SB_VERT, &si, TRUE) ;
                InvalidateRect (hwnd, NULL, TRUE) ;
                return 0 ;

        case   WM_USER_CHECKFILES:
                        // Get the system date & form filename fro

                GetSystemTime (&st) ;
                wsprintf (szFilename, TEXT ("UD%04i%02i.TXT")

                        // Check if the file exists; if so, read all the
```

```
                if (GetFileAttributes (szFilename) != (DWORD) -1
                {
                        SendMessage (hwnd, WM_USER_GETF
                        return 0 ;
                }
                        // Otherwise, get files from Internet.
                        // But first check so we don't try to copy file

                if (GetDriveType (NULL) == DRIVE_CDROM)
                {
MessageBox (hwnd, TEXT ("Cannot run this program from CD-R
          szAppName, MB_OK | MB_ICONEXCLAMATION) ;
                        return 0 ;
                }
                                // Ask user if an Internet connecti

                 if (IDYES == MessageBox (hwnd, TEXT ("Update
               szAppName, MB_YESNO | MB_ICONQUESTION))
```

```
                    // Invoke dialog box

               DialogBox (hInst, szAppName, hwnd, DlgProc) ;

                    // Update display

              SendMessage (hwnd, WM_USER_GETFILES, 0, 0)
               return 0 ;

     case   WM_USER_GETFILES:
               SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
               ShowCursor (TRUE) ;

                    // Read in all the disk files

               plist = GetFileList () ;

               ShowCursor (FALSE) ;
               SetCursor (LoadCursor (NULL, IDC_ARROW)) ;

                  // Simulate a WM_SIZE message to alte

               SendMessage (hwnd, WM_SIZE, 0, MAKELONG (
```

```
                InvalidateRect (hwnd, NULL, TRUE) ;
                return 0 ;

        case   WM_PAINT:
                hdc = BeginPaint (hwnd, &ps) ;
                SetTextAlign (hdc, TA_UPDATECP) ;

                si.cbSize = sizeof (SCROLLINFO) ;
                si.fMask = SIF_POS ;
                GetScrollInfo (hwnd, SB_VERT, &si) ;

                if (plist)
                {
                        for (i = 0 ; i < plist->iNum ; i++)
                        {
          MoveToEx       (hdc, cxChar, (i - si.nPos) * cyChar, NU
         TextOut        (hdc, 0, 0, plist->info[i].szFilename,
          lstrlen (plist->info[i].szFilename)) ;
          TextOut  (hdc, 0, 0, TEXT (": "), 2) ;
          TextOutA (hdc, 0, 0, plist->info[i].szContents,
```

```
                strlen (plist->info[i].szContents)) ;
                                }
                        }
                        EndPaint (hwnd, &ps) ;
                        return 0 ;

        case   WM_DESTROY:
                        PostQuitMessage (0) ;
                        return 0 ;
   }
        return DefWindowProc (hwnd, message, wParam, lParam)
}

BOOL CALLBACK DlgProc (    HWND hwnd, UINT message, WPARA
{
   static PARAMS params ;
        switch (message)
        {
        case   WM_INITDIALOG:
                        params.bContinue = TRUE ;
```

```
                params.hwnd = hwnd ;

                _beginthread (FtpThread, 0, s) ;
                return TRUE ;

        case   WM_COMMAND:
                switch (LOWORD (wParam))
                {
                case   IDCANCEL:   // button for user to abort do
                           params.bContinue = FALSE ;
                          return TRUE ;

                case   IDOK:      // button to make dialog box go
                           EndDialog (hwnd, 0) ;
                           return TRUE ;
                }
 }
        return FALSE ;
}
```

```
/*---------------------------------------------------------------------
  FtpThread: Reads files from FTP server and copies them to loca
---------------------------------------------------------------------*/

void FtpThread (PVOID parg)
{
        BOOL                                    bSuccess ;
        HINTERNET                       hIntSession, hFtpSessio
        HWND                                    hwndStatus, hw
        PARAMS                  *       pparams ;
        TCHAR                                   szBuffer [64] ;
        WIN32_FIND_DATA         finddata ;

        pparams             = parg ;
        hwndStatus    = GetDlgItem (pparams->hwnd, IDC_STATU
        hwndButton    = GetDlgItem (pparams->hwnd, IDCANCEL

                // Open an internet session

        hIntSession = InternetOpen (szAppName, INTERNET_OPE
```

```
                        NULL, NULL, INTERNET_FLAG_ASYNC) ;
        if (hIntSession == NULL)
        {
     wsprintf (szBuffer, TEXT ("InternetOpen error %i"), GetLastE
             ButtonSwitch (hwndStatus, hwndButton, szBuffer) ;
             _endthread () ;
        }

        SetWindowText (hwndStatus, TEXT ("Internet session ope
                    // Check if user has pressed Cancel
        if (!pparams->bContinue)
        {
             InternetCloseHandle (hIntSession) ;
             ButtonSwitch (hwndStatus, hwndButton, NULL) ;
             _endthread () ;
        }

             // Open an FTP session.
hFtpSession = InternetConnect (hIntSession, FTPSERVER, INTERI
                        NULL, NULL, INTERNET_SERVICE_FTP, 0, 0) ;
```

```
      if (hFtpSession == NULL)
      {
            InternetCloseHandle (hIntSession) ;
            wsprintf (szBuffer, TEXT ("InternetConnect error %i"),
                                                GetLastError ()) ;
      ButtonSwitch (hwndStatus, hwndButton, szBuffer) ;
      _endthread () ;
}

SetWindowText (hwndStatus, TEXT ("FTP Session opened..."))
                  // Check if user has pressed Cancel
if (!pparams->bContinue)
      {
            InternetCloseHandle (hFtpSession) ;
            InternetCloseHandle (hIntSession) ;
            ButtonSwitch (hwndStatus, hwndButton, NULL) ;
            _endthread () ;
      }
```

```
                // Set the directory
        bSuccess = FtpSetCurrentDirectory (hFtpSession, DIRECT
        if (!bSuccess)
        {
              InternetCloseHandle (hFtpSession) ;
              InternetCloseHandle (hIntSession) ;
              wsprintf (    szBuffer, TEXT ("Cannot set directory to "
                                                    DIRECTORY) ;
              ButtonSwitch (hwndStatus, hwndButton, szBuffer) ;
              _endthread () ;
        }

        SetWindowText (hwndStatus, TEXT ("Directory found..."))
              // Check if user has pressed Cancel
        if (!pparams->bContinue)
        {
              InternetCloseHandle (hFtpSession) ;
              InternetCloseHandle (hIntSession) ;
              ButtonSwitch (hwndStatus, hwndButton, NULL) ;
```

```
            _endthread () ;
        }

                    // Get the first file fitting the template
        hFind = FtpFindFirstFile (hFtpSession, TEMPLATE, &findda
        if (hFind == NULL)
        {
            InternetCloseHandle (hFtpSession) ;
            InternetCloseHandle (hIntSession) ;
            ButtonSwitch (hwndStatus, hwndButton, TEXT ("Canr
            _endthread () ;
        }

do
        {
                        // Check if user has pressed Cancel
         if (!pparams->bContinue)
            {
                        InternetCloseHandle (hFind) ;
                         InternetCloseHandle (hFtpSession) ;
```

```
                    InternetCloseHandle (hIntSession) ;
                    ButtonSwitch (hwndStatus, hwndButton, N
                    _endthread () ;
          }
                    // Copy file from internet to local hard disk,
                    // if the file already exists locally

          wsprintf (szBuffer, TEXT ("Reading file %s..."), findda
          SetWindowText (hwndStatus, szBuffer) ;

          FtpGetFile (  hFtpSession,
       finddata.cFileName, finddata.cFileName, TRUE,
       FILE_ATTRIBUTE_NORMAL, FTP_TRANSFER_TYPE_BINAR
     }
     while (InternetFindNextFile (hFind, &finddata)) ;
     InternetCloseHandle (hFind) ;
     InternetCloseHandle (hFtpSession) ;
     InternetCloseHandle (hIntSession) ;

     ButtonSwitch (hwndStatus, hwndButton, TEXT ("Internet
```

```
}

/*------------------------------------------------------------------------
 ButtonSwitch:  Displays final status message and changes Canc
------------------------------------------------------------------------*/
VOID ButtonSwitch (HWND hwndStatus, HWND hwndButton, TCH
{
          if (szText)
                    SetWindowText (hwndStatus, szText) ;
          else
                    SetWindowText (hwndStatus, TEXT ("Internet Se
          SetWindowText (hwndButton, TEXT ("OK")) ;
          SetWindowLong (hwndButton, GWL_ID, IDOK) ;
}

/*------------------------------------------------------------------------
  GetFileList: Reads files from disk and saves their names and co
------------------------------------------------------------------------*/

FILELIST * GetFileList (void)
```

```
{
        DWORD                                   dwRead ;
        FILELIST                        *       plist ;
        HANDLE                                  hFile, hFind ;
  int                                           iSize, iNum  ;
        WIN32_FIND_DATA             finddata ;

        hFind = FindFirstFile (TEMPLATE, &finddata) ;
        if (hFind == INVALID_HANDLE_VALUE)
                return NULL ;
        plist        = NULL ;
       iNum          = 0 ;

        do
        {
                                // Open the file and get the size
             hFile = CreateFile (finddata.cFileName, GENERIC_REA
                 NULL, OPEN_EXISTING, 0, NULL) ;

             if (hFile == INVALID_HANDLE_VALUE)
```

```
                    continue ;

        iSize = GetFileSize (hFile, NULL) ;

        if (iSize == (DWORD) -1)
        {
                    CloseHandle (hFile) ;
                    continue ;
        }
                   // Realloc the FILELIST structure for a n
        plist = realloc (plist, sizeof (FILELIST) + iNum * sizeof

                    // Allocate space and save the filenam

        plist->info[iNum].szFilename = malloc (lstrlen (findda
        lstrcpy (plist->info[iNum].szFilename, finddata.cFileN

                    // Allocate space and save the content

        plist->info[iNum].szContents = malloc (iSize + 1) ;
```

```
            ReadFile (hFile, plist->info[iNum].szContents, iSize, &
             plist->info[iNum].szContents[iSize] = 0 ;

             CloseHandle (hFile) ;
             iNum ++ ;
        }
        while (FindNextFile (hFind, &finddata)) ;
        FindClose (hFind) ;
                        // Sort the files by filename
        qsort (plist->info, iNum, sizeof (FILEINFO), Compare) ;
        plist->iNum = iNum ;
        return plist ;
}

/*--------------------------------------------------------------------------
  Compare function for qsort
--------------------------------------------------------------------------*/

int Compare (const FILEINFO * pinfo1, const FILEINFO * pinfo2)
{
```

```
    return lstrcmp (pinfo2->szFilename, pinfo1->szFilename) ;
}
```

UPDDEMO.RC

```
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"
//////////////////////////////////////////////////////////////////////////////
// Dialog
UPDDEMO DIALOG DISCARDABLE  20, 20, 186, 95
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMI
CAPTION "Internet Download"
FONT 8, "MS Sans Serif"
BEGIN
    PUSHBUTTON          Cancel",IDCANCEL,69,74,50,14
    CTEXT                                   "",IDC_STATUS,7,29,172,21
END
```

RESOURCE.H

```
// Microsoft Developer Studio generated include file.
```

```
// Used by UpdDemo.rc

#define IDC_STATUS     40001
```

UPDDEMOUDyyyymm.TXTyyyy42000mm2

WinMainShowWindowUpdateWindowUPDDEMOWndProc
WM_USER_CHECKFILESWndProcUDyyyymm.TXTUPDDEMO
UPDDEMOWM_USER_GETFILES
GetFileListUPDDEMO.CUDyyyymm.TXTFILELIST

UPDDEMOInternetOKCancelIDIDC_STATUS
DlgProc

DlgProcPARAMSbContinueBOOL_

FtpThreadInternetOpenInternetConnectFtpSetCurrentDirectory
FtpFindFirstFileInternetFindNextFileFtpGetFileInternetCloseHandle
FtpThreadhwndStatusSetWindowText

FtpThreadWinInetCancelButtonSwitch
ButtonSwitchCancelOKIDOK

FtpThreadInternet

DlgProcPARAMSbContinueFALSEFtpThreadbContinue
FALSENULLButtonSwitchInternet

UPDDEMOUPDDEMO