第 1 章　简介

1.1　计算机绘图及图象处理的应用

目前计算机绘图及图象处理的应用相当广泛，从电子游戏到太空探测，几乎在各行各业都可以见到，以下我们就介绍几个常见的应用领域：

娱乐事业

各式大型电动玩具以及计算机游戏，不论是角色扮演、动作冒险、战略、模拟、益智、射击……等，都涉及到颇复杂的图象处理。

刑事侦查

刑警在办案时，必须将照片、指纹、笔迹、人像等案情相关资料，作进一步的处理和鉴别，找到证据或线索。

医学研究

借助计算机绘图及图象处理技术，如断层扫描、超声波成象、X 射线成象等，可以让医生更快更准确地诊断疾病。

科学研究

诸如飞机航道显示、自然灾害预测、宇宙星体照片等，都是利用计算机绘图及图象处理来取得所需信息，当然每天天气预报中的卫星云图，也是应用的范畴。

工业生产

大家熟悉的计算机辅助设计(Computer Aided Design，简称 CAD)和计算机辅助制造(Computer Aided Manufacturing，简称 CAM)，在工业生产中广泛应用是众所周知的，越来越多的人已不再用手去画建筑设计图或汽车结构图。

除了上述应用之外，还有许许多多的应用实例出现在我们的生活之中，例如：电子书籍、电子相薄、电子暗房以及各式各样的多媒体应用系统。电子暗房就是图象处理软件。

最近几年由于 PC 的速度愈来愈快，价格愈来愈低，加上各种 Super VGA 卡的产生，当然最主要的原因还是由于 Windows 的出现且日趋成熟，使我们越来越难发现有 Macintosh 计算机提供、而 PC 却做不到的事情。过去 Macintosh 的图象处理软件遥遥领先，Adobe Photoshop for Mac 使得 PC 上的图象处理软件相形见拙。但随着 Photoshop 与 Fractal Design 公司的 Color Studio 相互配合且一起进入 Windows 后，现在的形势已有所逆转。还记得几年前，能够在 Windows 上面处理全彩图象的软件可以说是屈指可数，不过在图象处理软件厂商几年来的细心耕耘之下，已经呈现出崭新局面，而今角逐此间市场的各路英雄已不在少数，以下即为常见的图象处理软件开发厂商及其软件名称：

Adobe System Inc.	Adobe Photoshop for Windows
Aldus Corp.	PhotoStyler
Micrografx Inc.	Picture Publisher

	Photomagic for Windows
Computer Associates	CA-Cricket Paint
	CA-Cricket Image
Image-In	Image-in-Color

PC 的图象处理软件市场规模还不能与 Macintosh 环境的相提并论，但是它发展的相当迅速，我们相信 24 位(bit)图形及图象日渐普及于现在的计算机世界时，图象处理软件将会是主流产品之一，而在不久的将来，计算机绘图及图象处理的应用也会更深入我们的生活。

1.2 相关知识简介

介绍完目前有关图象处理方面的应用后，您是否也想要投入这众所瞩目的领域呢？首先我们必须先来了解有关图象文件的基本概念。

我们首先了解一下如何储存那些丰富多彩的计算机展示画面呢？原则上这些图象文件大致可分为三大类：

点阵型

采用位映射(bit-mapped)的方式储存图形图象。例如大家最熟悉的中文字型，它的 15 * 16 字型和 24 * 24 字型就是使用此种方式存储。一般我们所常见的点阵型图象文件常用 BMP、PCX 、GIF 等作后缀，而这些图象文件也就是本书所要详细探讨的内容。

向量型

采用直线(line)、曲线(curve)、矩形(rectangle)、圆形(circle)等向量(vector)的方式储存图形。市面上所谓的向量字型就是使用这种方式储存(当然有的向量图内也可以储存点阵型图)，而一般我们所常见的向量图文件有 Windows 的 .WMF、Lotus 的 .PIC、CorelDraw 的 .CDR……等。

动画型

所谓动画型图象其实可以说是由许多的点阵型图象所组成，但是有些更高级的动画型图象则可以分析出两个画面的差异，然后只储存其有差异部分。例如有一只海鸥正飞过一座高山，此时当作背景的高山是不会有任何改变的，因此只需储存画面间海鸥飞过时所改变的地方即可。一般我们常见的动画图象文件有 Animator 的 .FLI、MacroMind Director 的 .MMM 等，而目前最受众人瞩目的动画型图象则为 MPEG(Motion Picture Experts Group)数字电视图象。

要设计出一个好的图象格式并非易事，它至少应具备以下几个特点：

结构容易了解
编程简单
支持多种色彩
扩充能力强
压缩比例高

执行速度快

目前的图象格式各有千秋，每一种类型的图形图象文件都有很多种存储格式，这些图形图象文件储存格式的提出，可能是由于以下几种情形：

为配合软件或硬件而制定

如 ZSoft 的 Paintbrush 所使用的 .PCX、Micrsoft Windows 所使用的 .BMP 以及 Truevision Targe 全彩显示卡所使用的 .TGA。

计算机相关厂商制定

如由 Aldus、Microsoft、Hewlett-Packard、Dest、Datacopy、Microtek、Media Cybernetics、New Image、Technology 和 Software Publishing 八家公司共同发表的 TIFF。

国际性的组织或协会所制定

如由 ISO 和 CCITT 为静态图象所建立的第一个国际数字图象压缩标准——JPEG。

由于格式繁多，我们选择几种常见或重要的格式介绍，以下是本书将要探讨的图象文件格式：

1. BMP 图象文件
2. JPEG 图象文件
3. PCX 图象文件
4. TGA 图象文件
5. GIF 图象文件
6. TIFF 图象文件

其中本书上册所要研究的是 BMP 和 JPEG 图象文件，至于其它的图象文件则会在本书的下册为读者详细的说明。特别针对静态图象所建立的第一个国际数字压缩标准——JPEG 图象文件，进行广泛而深入地探讨。从理论说明到详解压缩、解压缩的编程方法，然后附上原文的说明。资料可以说是相当的丰富。

一般说来，一个图象文件除了本身的图象数据以外，都还会有一个文件头，用于说明如何使用文件中的图象数据，而处理图象文件真正麻烦的地方就在这里。因为这部分根本就是一人一个号，各吹各的调，再加上新旧版本的差异甚至不相容，所以在编程过程中，给程序设计师增添了不少的困扰，关于这一点以后读者就会慢慢了解。文件头(header)可以说是图象文件中最重要的部分，它包含了图象的长宽、大小、分辨率、图象色彩数、调色板数据……等相关信息，有的图象文件甚至还会将公司名称、软件名称或是作者名称一并储存。除这些文件头之后，剩下的部分就是图象数据，而图象文件通常为了节省空间，往往都会使用一些压缩方法对图象数据进行编码，一般常见的压缩方法有 Run length 法、LZW 法、霍夫曼编码法……等。原则上较简单的压缩方法其执行效率会稍高、但是压缩比例就会较低；反之较复杂的压缩方法其执行效率会稍低，但是压缩比例就会较高。各个图象文件的文件头及压缩方法，会在后面的章节内为您介绍，但是在正式介绍图象文件格式之前，我们必须知道，根据图象色彩数目不同，图象也有下列的分类：

每点所需位数	色彩数目
1	2

4	16
8	256
16	65535
24	全彩
32	全彩

大部分的图象文件格式都支持24位的真彩色图象，但是支持32位的图象文件格式的就不多了，以下为我们介绍的六种图象文件格式及所支持的位数，如下所示：

图象文件格式	1位	4位	8位	16位	24位	32位
BMP	*	*	*		*	
JPEG			*		*	
PCX	*	*	*		*	
TGA	*		*	*	*	*
GIF	*	*	*			
TIFF	*	*	*		*	*

第 2 章　了解 Windows 环境

目前选择在 Windows 上开发软件的厂商有越来越多的趋势，为了让各位读者也能了解其中奥秘，在这一章中，先简单介绍一下 Windows 的发展历史；而本书主要是以程序设计为导向，将会为大家介绍在 Windows 上开发软件的好处。

2.1　Windows 简介

在 80 年代，由微软公司（Microsoft）所推出的 MS-DOS（Disk Operation System）可以说是个人计算机（Personal Computer），简称 PC 上的宠儿，每个 PC 使用者手上几乎都有一套 MS-DOS。当初 Microsoft 在推出 MS-DOS 时，有很多的专家并不看好它，但是在行销策略和广告的带动下，MS-DOS 却脱颖而出，打败了当时的竞争者，成为全球数千万台 PC 上的操作系统。MS-DOS 是一个提供文字模式、只支持单任务操作、功能并不是很强大、其用户界面也不怎么友善的操作系统，它陪伴大家走过了 80 年代。从目前的迹象看来，广大的 DOS 用户在短期之内，应该还不会抛弃这个十几年的老朋友。

继 MS-DOS 之后，Microsoft（当时还不能算是老大）于 1983 年又对外发表即将推出新一代的操作环境——Windows（话说的太早，结果拖了两年后才正式发行），企图再创另一个高峰。但是最初的 Windows 因为受限于硬件、开发技术和规划上的不成熟，刚开始并未得到太多的掌声。一直等到 1990 年（上述问题获得解决后）Microsoft 推出了 Windows3.0，全球开始掀起一股 Windows 风潮，许多硬件厂商积极推出与 Windows 相容的机种，而软件公司也开始注意这块尚未开垦的土地，相继推出 Windows 版的应用软件，或是把以往的产品移植到 Windows 上。时至今日，Windows 上的软件可以说是一应俱全，文字、排版、图象、绘图、通信、电子表格等等，不论是专业领域或者是一般用途，使用者将会发现有越来越多且功能更加强大的应用软件可供选择。

1983 年 11 月　Microsoft 对外宣布即将发表 Windows
1985 年 11 月　发行 Windows 1.03
1987 年 10 月　发行 Windows 2.0 & Windows 386
1988 年
1989 年
1990 年 5 月　发行 Windows 3.0
1991 年 8 月　发行 MS-Windows 3.0 中文版
1992 年 4 月　发行 MS-Windows 3.1
1993 年 5 月　发行 MS-Windows 3.1 中文版

我们相信 90 年代掌握 Windows 就像 80 年代掌握 DOS 一样的重要，甚至更重要，所

以如果各位读者有心踏入 Windows 的领域，我们的建议是越快越好。

2.2 Windows 特点

如果您问别人 Windows 和 DOS 究竟有什么不同，得到的答案通常是："DOS 是文字操作环境，而 Windows 是图形操作环境"。其实除了这个差异之外，Windows 尚有许多令人欣赏的特性：

设备无关性

Windows 提供了一个设备无关性(device independent)的环境，使得在 Windows 上的应用软件不需要直接控制各式各样的外部设备，链接程序完全由 Windows 自行处理。因此，只要外部设备本身提供有 Windows 的驱动程序(driver)，那么在 Windows 上的应用软件就能支持这种外部设备。

多任务操作

Windows 是一个多任务(multitasking)的操作环境，它允许使用者同时执行数个不同的应用程序。例如：我们可以同时执行文字处理软件及图象处理软件，而当图象编辑工作告一个段落时，我们可以直接切换至文字处理软件去输入资料，并不需要先退出图象处理软件。

存储器管理

为了实现多任务的功能，Windows 提供了相当完善的存储器管理，使得同时执行的数个应用程序不至于将存储器用尽，而存储器的使用也不再受限于 640K。

一致性的使用者界面

由于在 Windows 上的应用软件都是由一些基本目标(object)模块所组成，例如：标题栏(caption bar)、功能表(menu)、按键(button)、滚动条(scroll bar)等，所以每个应用软件都有相同的基本外观，而其操作方式也大同小异。因此，使用者不需要再花长时间去学习如何使用一个新的应用软件。

动态链接

传统的静态链接方式是将编译完的目标(OBJ)模块与函数库中的函数模块链接成为一个可执行文件(.EXE)，也就是将函数模块拷贝至可执行文件内。如果在多任务的操作环境中同时执行两个应用程序，而它们都调用许多相同的函数，那么就会造成相同的函数码占用数块存储空间的情况，因此 Windows 采用了动态链接(dynamic linking)的方式。通过这种方式，应用程序在执行时才链接其调用的函数，这不仅节省了存储的空间，也可以达到资源共享的目的。

动态数据交换

Windows 支持了三种应用程序之间的数据交换方式：动态链接函数库(Dynamic Linking Library，简称 DLL)的共用存储器(shared memory in DLL)、剪贴簿(clipboard)和动态数据交换(Dynamic Data Exchange，简称 DDE)。由于前面两种数据交换方式都有其使用上的限制和不便，而通过 DDE 做数据交换的应用程序，其间的关系一旦被建立之后便不再需要任何外部的控制，因此 DDE 可以说是最佳的数据交换方式。

多重文件界面

多重文件界面(Multiple Document Interface,简称 MDI)是一个在 Windows 上可以处理多份文件的应用程序规格。通过 MDI,您可以在一个应用程序中同时处理数份文件,以取代同时执行数个应用程序来处理数份文件的方式。

第 3 章　在 Windows 上显示图象

3.1　操作环境的差异

如果各位读者想要在 DOS 上发展一套图象处理软件的话，首先面临的问题就是如何建立一个完整的图形操作环境。由于 DOS 本身是文字模式的操作环境，因此您必须先克服一些基本的问题：

显示卡种类

Hercules、EGA、VGA、Super VGA…。

显示卡厂牌

S3、ATI、Tseng、Trident…。

显示存储器

0xA0000、0xA8000、0xB0000、0xB800…。

显示分辨率

640 * 480、720 * 348、800 * 600、1024 * 768、1280 * 1024…。

显示色彩数

2 colors、16 colors、256 colors、True color…。

打印机种类

单色、彩色、点阵、喷墨、激光、PostScript…。

打印机品牌

Epson、Panasonic、HP、Canon…。

界面标准

TWAIN、VESA、EMS、XMS、DPMI、VCPI…。

绘图基本函数

Pixel()、Line()、Circle()、Rectangle()…。

为了建立一个完整的图形操作环境，光是克服上述这些“基本问题”就必须花上个一年半载，而且结果也不见得令您满意，更何况您的应用软件都还没开始动工！或许您会觉得把问题说得太严重了，但是想要在 DOS 上建立一个完整的图形操作环境的确不是一件容易的事情。如果各位读者选择在 Windows 上开发一套图象处理软件的话，那么由于 Windows 本身就是一个极佳的图形操作环境，所以我们只需要专心于应用软件的开发，不必为建立一个完整的图形操作环境来伤脑筋。除此之外，Windows 又具备了设备无关特性，使得原本牵涉复杂的外部设备控制，例如：图象显示、图象打印等问题，都可以通过图形设备接口(Graphics Device Interface，简称 GDI)来处理。

GDI 本身是一个包含各种绘图函数的函数库(GDI.EXE)，所有与外部设备驱动程序(driver)沟通的工作均由它负责，应用软件并不需要直接控制各式各样的外部设备。事实

上 Windows 采用设备无关性的作法对于硬件厂商也有莫大的益处，因为他们只需要在新产品推出的时候顺便附上 Windows 的驱动程序，那么由于在 Windows 上所有的应用软件都能够使用该产品，所以他们可以大肆宣传该产品已经有数百种应用软件的支持，而不必像过去一样担心支持应用软件太少。当然他们也专心于开发能充分展现该产品特性的驱动程序，故以往因为由软件公司各自开发驱动程序而导致互不兼容的质量问题也迎刃而解了。

通过 GDI 我们可以达到设备无关的目的，但是在编写程序之前，我们还必须先对设备环境(Device Context，简称 DC)有一清楚的认识。DC 是一个由 Windows 负责维护的数据结构，它包含了该外部设备外部属性(我们可以使用 GetDeviceCaps 函数取得相关信息)，而我们准备对该外部设备输出数据之前，必须先取得 DC 的句柄(handle)，然后才可以进行输出。

```
⋮
hDC = Getdc(hWnd);
Rectangle(hDC, 100, 100, 300, 300)
ReleaseDC(hWnd, hDC);
⋮
```

以上面的例子来说，由于我们的输出过程是针对 DC 处理而不是显示卡本身，所以即使是将显示卡由 EGA 换至 VGA、16 色换至 256 色……，对我们的程序都没有丝毫的影响，依旧可以画出矩形的图形。

3.2 位图函数介绍

Windows 提供的一系列位图函数如下：

函数名称	函数功能
BitBlt	将位图由一个 DC 拷贝到另一个 DC
CreateBitmap	建立一个与设备相关的位图
CreateBitmapIndirect	依照 BITMAP 数据结构建立一个位图
CreateCompatibleBitmap	建立一个与 DC 相容的位图
CreateDIBitmap	依照 DIB 的数据建立一个位图
CreateDIBPatternBrush	依照 DIB 的数据建立一个图形画笔
CreateDiscardableBitmap	建立一个可抛弃的位图
GetBitmapBits	取得位图的位数据
GetBitmapDimension	取得位图的宽度及长度
GetDIBits	取得 DIB 的位数据
GetStretchBltMode	取得位图放大缩小的调整模式
LoadBitmap	载入一个位图资源
PatBlt	将图形式画笔输出到 device
SetBitmapBits	设定位图的位数据

SetBitmapDimension	设定位图的宽度及长度
SetDIBits	设定 DIB 的位数据
SetDIBitsToDevice	将 DIB 直接输出到 device
SetStchBltMode	设定位图放大缩小的调整模式
StretchBlt	与 BitBlt 相同,但有放大缩小的功能
StretchDIBits	与 StretchBlt 相同,但针对 DIB 处理

由于在本书的程序编写中,仅使用了 CreateCompatibleBitmap()、SetDIBits()以及 BitBlt()三个函数,因此我们只针对这三个函数作详细的介绍。

CreateCompatibleBitmap

函数语法:HBITMAP CreateCompatibleBitmap(HDC hDC, int iWidth, int iHeight);

函数功能:建立一个与 DC 相容的位图。

函数参数说明:

hDC	DC 的句柄(handle)。
iWidth	位图的宽度(以象素为单位)。
iHeight	位图的高度(以象素为单位)。

函数返回值:

成功	该位图的句柄。
失败	NULL。

SetDIBits

函数语法:int SetDIBits (HDC hDC, HANDLE hBitmap,
WORD wStartScan, WORD wNumScans,
LPSTR lpBits,
LPBITMAPINFO lpBitsInfo,
WORD wUsage);

函数功能:设定 DIB 的位数据。

函数参数说明:

hDC	DC 的句柄。
hBitmap	位图的句柄。
wStartScan	起始扫描线的编号。
wNumScans	扫描线的数目。
lpBits	指向 DIB 位数据的指针。
lpBitsInfo	指向 BITMAPINFO 数据结构的指针。
wUsage	DIB_PAL_COLORS:lpBitsInfo 中的 bmiColors[],其内容为指向目前调色板的索引值。 DIB_PAL_COLORS:lpBitsInfo 中 bmiColors[],其内容为实际的 RGB 值。

函数返回值:

成功	被改变的扫描线数目。

失败　　0。

BitBlt

函数语法：BOOL BitBlt(HDC hDstDC, int iDstX, int iDstY,
int iWidth, int iHeight, HDC hSrcDC,
int iSrcX, int iSrcY, DWORD dwRop);

函数功能：将位图由一个 DC 拷贝到另一个 DC。

函数参数说明：

hDstDC　　目的(destination)DC 的句柄。

iDstX　　目的 DC 左上点的 X 逻辑坐标值。

iDstY　　目的 DC 左上点的 Y 逻辑坐标值。

iWidth　　位图的宽度(逻辑单位)。

iHeight　　位图的长度(逻辑单位)。

hSrcDC　　图源(source)DC 的句柄。

iSrcX　　图源 DC 左上点的 X 逻辑坐标值。

iSrcY　　图源 DC 左上点的 Y 逻辑坐标值。

dwRop　　光栅操作码(Raster Operation Codes, 简称 ROP)。表 3-1 为操作码的具体名称和布尔表达式。

表 3-1

名称	ROP 码	布尔表示式
BLACKNESS	000042	0
DSTINVERT	550009	~
MERGECOPY	C000CA	P&S
MERGEPAINT	BB0226	~S|D
NOTSRCCOPY	330008	~S
NOTSRCERASE	1100A6	~(S|D)
PATCOPY	F00021	P
PATINVERT	5A0049	P^D
PATPAINT	FB0A09	P|~S|D
SRCAND	8800C6	S&D
SRCCOPY	CC0020	S
SRCERASE	440328	S&~D
SRCINVERT	660046	S^D
SRCPAINT	EE0086	S|D
WHITENESS	FF0062	1

S——Source bitmap

D——Destination bitmap

P——Pattern brush

函数返回值：

成功　　非 0 的值。
失败　　0。

3.3 编程

＜SHOWIMG＞

```
all: showimg.exe

ASM       = masm -v -ML -Mx
WINCC     = cl -c -Gsw -W3 -AM -Zp -Ow
WINLINK   = link /NOD /align: 16

.c.obj:
   $(WINCC) $ *.c

OBJS = showimg.obj   \
       wndproc.obj   \
       imprtbmp.obj  \
       mcsutil.obj   \
       fileopen.obj

showimg.res   :  showimg.rc         sys.h    mcsdef.h
         rc   -r  showimg.rc

showimg.obj   :  showimg.c          sys.h    mcsdef.h
      $(WINCC)  showimg.c

wndproc.obj   :  wndproc.c    sysextrn.h    mcsdef.h
      $(WINCC)  wndproc.c

imprtbmp.obj  :  imprtbmp.c   sysextrn.h    mcsdef.h
      $(WINCC)  imprtbmp.c

mcsutil.obj   :  mcsutil.c    sysextrn.h    mcsdef.h
      $(WINCC)  mcsutil.c

fileopen.obj  :  fileopen.c   sysextrn.h    mcsdef.h
      $(WINCC)  fileopen.c

showimg.exe   :  $(OBJS) showimg.def showimg.res
                 $(WINLINK) $(OBJS),,,libw mlibcew,showimg
                 rc showimg.res
```

<FILEOPEN.C>

```
#include "sysextrn.h"
#include <string.h>

BOOL FAR PASCAL OpenDialog(HWND hDlg, WORD wMsg, WORD wParam, LONG lParam);
LPSTR lstrchr(LPSTR lpszStr, char cChar);
LPSTR lstrrchr(LPSTR lpszStr, char cChar);

BOOL FAR PASCAL OpenDialog(HWND hDlg, WORD wMsg, WORD wParam, LONG lParam)
{
   HWND   hWndOK;
   LPSTR  lpszPtr1;
   LPSTR  lpszPtr2;
   BYTE   szDefSpec[6] = "*.BMP";
   BYTE   szTmpStr[65];

   switch(wMsg)
   {
      case WM_INITDIALOG:
      {
         lstrcpy((LPSTR)gszOpenFile, (LPSTR)szDefSpec);
         DlgDirList(hDlg, (LPSTR)gszOpenFile, IDD_DIRLIST , IDD_PATHNAME , 0xC010);
         DlgDirList(hDlg, (LPSTR)gszOpenFile, IDD_FILELIST, NULL          , 0x8020);
         SetDlgItemText(hDlg, IDD_FILENAME, (LPSTR)gszOpenFile);
         EnableWindow(GetDlgItem(hDlg, IDD_OK), FALSE);
         SendDlgItemMessage(hDlg, IDD_FILENAME, EM_SETSEL, 0, MAKELONG(0, 0x7FFF));
         SetFocus( GetDlgItem(hDlg, IDD_FILENAME) );
         return FALSE;
      }
      case WM_COMMAND:
      {
         switch(wParam)
         {
            case IDD_FILENAME:
            {
               if ( HIWORD(lParam) == EN_CHANGE )
               {
                  hWndOK = GetDlgItem(hDlg, IDD_OK);
                  EnableWindow(hWndOK, (SendMessage(LOWORD(lParam),
                                       WM_GETTEXTLENGTH, 0, 0L)>0) ? TRUE :
                                                                     FALSE);
                  SendMessage(hWndOK, BM_SETSTYLE,
```

```
                (WORD)BS_DEFPUSHBUTTON, 1L);
        SendMessage(GetDlgItem(hDlg, IDD_CANCEL), BM_SETSTYLE,
                (WORD)BS_PUSHBUTTON, 1L);
    }
    return TRUE;
}
case IDD_DIRLIST:
{
    switch( HIWORD(lParam) )
    {
        case LBN_SELCHANGE:
        {
            DlgDirSelect(hDlg, (LPSTR)szTmpStr, IDD_DIRLIST);
            lstrcat((LPSTR)szTmpStr, (LPSTR)gszOpenFile);
            SetDlgItemText(hDlg, IDD_FILENAME, (LPSTR)szTmpStr);
            break;
        }
        case LBN_DBLCLK:
        {
            lpszPtr1 = (LPSTR)szTmpStr;
            GetDlgItemText(hDlg, IDD_FILENAME, lpszPtr1, 64);
            /* Separate filename and pathname */
            if ( (lpszPtr2 = lstrrchr(lpszPtr1, ':')) ||
                 (lpszPtr2 = lstrrchr(lpszPtr1, '\\')) )
            {
                lpszPtr2++;
                lstrcpy((LPSTR)gszOpenFile, lpszPtr2);
                lstrcpy((LPSTR)gszOpenPath, lpszPtr1);
                gszOpenPath[ (lpszPtr2 - lpszPtr1) ] = NULL;
            }
            lstrcpy((LPSTR)szTmpStr, (LPSTR)gszOpenPath);
            lstrcat((LPSTR)szTmpStr, (LPSTR)gszOpenFile);
            DlgDirList(hDlg, (LPSTR)szTmpStr, IDD_DIRLIST,
                                        IDD_PATHNAME, 0xC010);
            DlgDirList(hDlg, (LPSTR)gszOpenFile, IDD_FILELIST,
                                        NULL, 0x8020);
            gszOpenPath[0] = NULL;
            SetDlgItemText(hDlg, IDD_FILENAME, (LPSTR)gszOpenFile);
            break;
        }
    }
```

```
            return TRUE;
        }
        case IDD_FILELIST:
        {
            switch( HIWORD(lParam) )
            {
                case LBN_SELCHANGE:
                {
                    DlgDirSelect(hDlg,(LPSTR)szTmpStr,IDD_FILELIST);
                    SetDlgItemText(hDlg,IDD_FILENAME,(LPSTR)szTmpStr);
                    break;
                }
                case LBN_DBLCLK:
                {
                    goto PressOK;
                }
            }
            return TRUE;
        }
        case IDD_OK:
        {
PressOK:
            lpszPtr1 = (LPSTR)szTmpStr;
            GetDlgItemText(hDlg,IDD_FILENAME,lpszPtr1,64);
            if ( ! szTmpStr[0] )
            {
                MessageBox(hDlg,"未指定文件名",
                                NULL,MB_OK | MB_ICONHAND);
                return TRUE;
            }
            if ( lstrchr(lpszPtr1,'*') || lstrchr(lpszPtr1,'?') )
            {
                /* Separate filename and pathname */
                if ( ! (lpszPtr2 = lstrrchr(lpszPtr1,':')) &&
                    ! (lpszPtr2 = lstrrchr(lpszPtr1,'\\')) )
                {
                 lstrcpy((LPSTR)gszOpenFile,lpszPtr1);
                 gszOpenPath[0] = NULL;
                }
                else
                {
                    lpszPtr2 ++;
```

```
                lstrcpy((LPSTR)gszOpenFile, lpszPtr2);
                lstrcpy((LPSTR)gszOpenPath, lpszPtr1);
                gszOpenPath[ (lpszPtr2 - lpszPtr1) ] = NULL;
            }
            lstrcpy((LPSTR)szTmpStr, (LPSTR)gszOpenPath);
            lstrcat((LPSTR)szTmpStr, (LPSTR)gszOpenFile);
            DlgDirList(hDlg, (LPSTR)szTmpStr, IDD_DIRLIST,
                                              IDD_PATHNAME, 0xC010);
            DlgDirList(hDlg, (LPSTR)gszOpenFile, IDD_FILELIST, NULL, 0x8020);
            gszOpenPath[0] = NULL;
            SetDlgItemText(hDlg, IDD_FILENAME, (LPSTR)gszOpenFile);
            EnableWindow(GetDlgItem(hDlg, IDD_OK), FALSE);
            return TRUE;
        }
        GetDlgItemText(hDlg, IDD_FILENAME, (LPSTR)gszOpenFile, 64);
        GetDlgItemText(hDlg, IDD_PATHNAME, (LPSTR)gszOpenPath, 64);
        EndDialog(hDlg, TRUE);
        return TRUE;
      }
      case IDD_CANCEL:
      {
        EndDialog(hDlg, FALSE);
        return FALSE;
      }
    }
    break;
  }
}
return FALSE;
}
LPSTR lstrchr(LPSTR lpszStr, char cChar)
{
  int iChar;
  iChar = (int)( cChar & (char)0xFF );
  return _fstrchr(lpszStr, iChar);
}
LPSTR lstrrchr(LPSTR lpszStr, char cChar)
{
  LPSTR lpszTmp;
  lpszTmp = lpszStr + (LONG)lstrlen(lpszStr);
```

```
    while( lpszTmp > lpszStr )
    {
        if ( * lpszTmp = = cChar )
        {
            return lpszTmp;
        }
        lpszTmp = AnsiPrev(lpszStr, lpszTmp);
    }
    return NULL;
}
```

<IMPRTBMP.C>

```
# include "sysextrn.h"

int iReadBMP(HWND hWnd, HANDLE hDC, LPSTR lpszSrcFName);
void vStatusProcess(int iPercentage);
HPALETTE hCreatePalette(RGBQUAD FAR * lpRGB, WORD wColors);
void vDrawUpButton(HDC hDC, RECT Rect, HBRUSH hBrush, int iShade);

int iReadBMP(HWND hWnd, HANDLE hDC, LPSTR lpszSrcFName)
{
    BITMAPFILEHEADER    BFH;
    BITMAPINFOHEADER    BIH;
    HPALETTE            hPalette;
    HPALETTE            hOldPal;
    HPALETTE            hMemOldPal;
    HBITMAP             hBMP;
    HBITMAP             hMemOldBMP;
    HANDLE              hBI_Struct;
    HANDLE              hImage;
    HDC                 hMemDC;
    LPBITMAPINFOHEADER  lpBIH;
    LPWORD              lpwIndex;
    LPSTR               lpImage;
    RGBQUAD FAR         * lpRGB;
    WORD                wBI_Size;
    WORD                wPalSize;
    WORD                wColors;
    WORD                wBits;
    WORD                wWidthBytes;
    WORD                wWidth;
    WORD                wHeight;
```