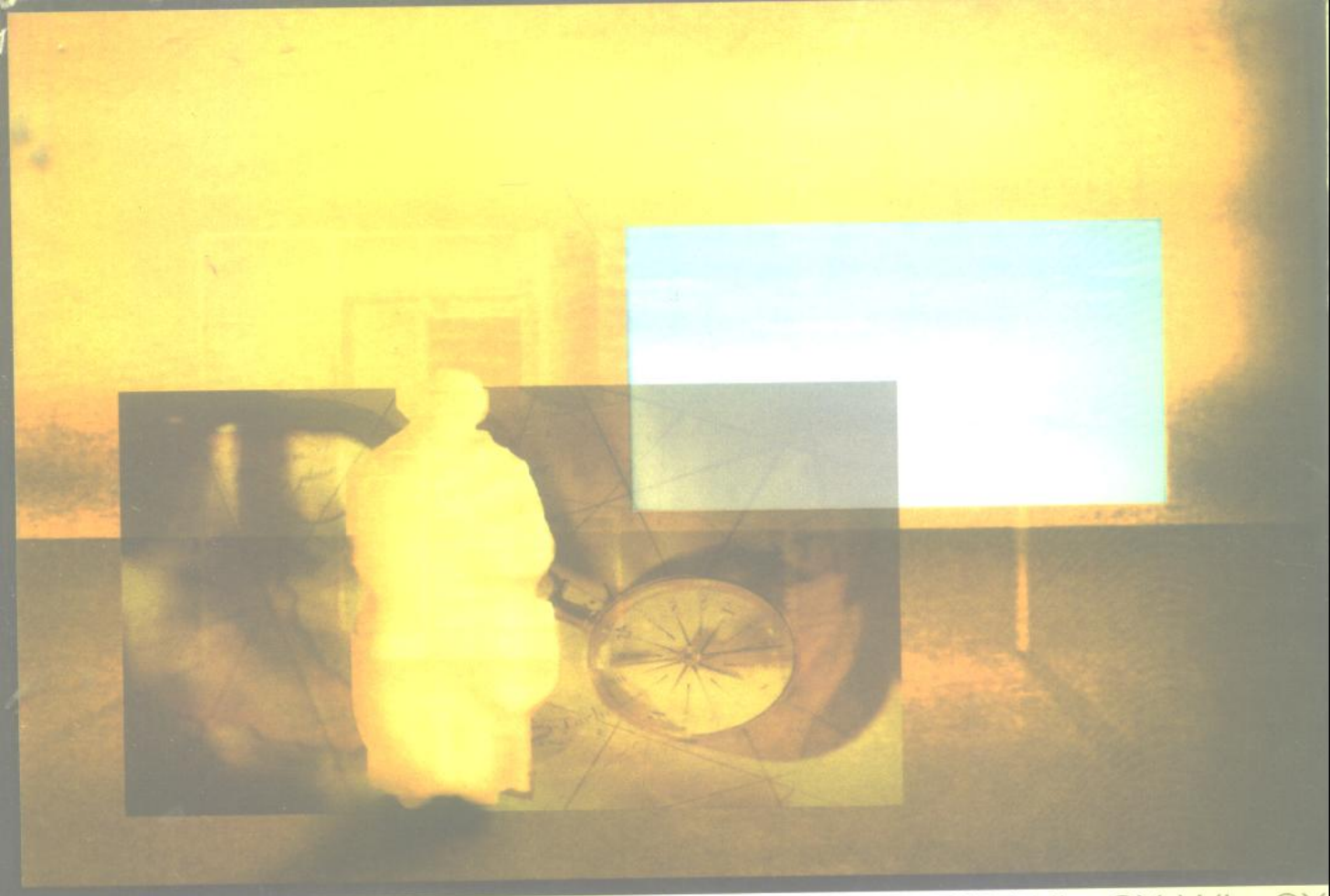

Win 9X Win 9X Win 9X Win 9X Win 9X Win 9X Win 9X Win 9X

Win 9X

虚拟设备驱动程序

编程指南

杨强 李堂秋 编著

清华大学出版社

Win 9X

虚拟设备驱动程序

编程指南

Win9x 虚拟设备驱动程序编程指南

杨 强 李堂秋 编著

清华大学出版社

(京)新登字 158 号

内 容 简 介

如何高效编写 Windows 98 及 Windows 95 的设备驱动程序（VxD）是系统编程人员和广大工程技术人员迫切需要解决的问题。本书作者采用 VToolsD 编写 Win 9x 虚拟设备驱动程序，积累了丰富的实践经验。书中详细剖析了 VToolsD 的类库，介绍了硬件中断、I/O 监控，软中断监控、异步事件等各类 VxD 的编写方法，还总结了 Win32 应用程序和 VxD 的通信方法。

本书是 C++编程人员和计算机应用技术人员的实用参考书。

图书在版编目(CIP)数据

Win 9X 虚拟设备驱动程序编程指南/杨强，李堂秋编著。—北京：清华大学出版社，1999.3
ISBN 7-302-03324-2

Ⅰ.W… Ⅱ. ①杨… ②李… Ⅲ. 窗口软件，Windows-程序设计 Ⅳ.TP312

中国版本图书馆 CIP 数据核字 (1999) 第 00803 号

出版者：清华大学出版社（北京清华大学校内，邮编 100084）
http://www.tup.tsinghua.edu.cn
印刷者：北京市清华园胶印厂
发行者：新华书店总店北京发行所
开 本：787×1092 1/16 印张：9 字数：209 千字
版 次：1999 年 3 月第 1 版 1999 年 3 月第 1 次印刷
书 号：ISBN 7-302-03324-2/TP · 1788
印 数：0001～5000
定 价：15.00 元

前　言

欢迎您进入 Windows 系统级编程的世界。

首先提一个问题：您为什么要选择这本书？我想，您肯定是要么想深入钻研 Windows 的底层结构，要么想通过这本书顺利地解决在实际工作中遇上的 windows 程序通常难以解决的困难。本书把重点放在对后者的解答上，在书中的必要之处，也介绍前者的有关知识。

虚拟设备驱动程序 (VxD) 是针对 Win9x 以下平台的。Microsoft 公司提供了 Windows 操作系统的多个版本，像 Win3.1、Win95、WinNT 等。这些版本所采用的核心技术不尽相同。为了保证兼容性，Microsoft 公司鼓励 Windows 程序员利用 WIN API 以使得应用程序在这些平台上都能运行。但是由于设备驱动程序运行在操作系统底层(ring0 层)，WIN API(ring3 层)不能使用，因此不同的 Windows 操作系统就必然有不同的设备驱动程序编制方法。

值得庆幸的是，从 Win3.0 到 Win95 (Win98?)，虽然它们的体系结构发生了众多变化，但是它们的底层结构却基本没有改变。这就告诉我们，为 Win3.x 编写的设备驱动程序，可以在基本不需要修改的情况下运行在 Win95 甚至 Win98 平台上。当然 Win95 和 Win98 又扩充了许多新的设备驱动程序特性，例如 Win95 增加了对 PnP 的支持，Win98 还加入了对 USB 的支持。

WinNT 采用比 Win95 和 Win98 先进得多的设计技术，因而系统有很好的灵活性和鲁棒性。可是 WinNT 在设计上却不再保持与 Win3.1 和 Win9x 的兼容性，为后者编写的设备驱动程序不能在 WinNT 下使用(NT 使用称作 NT Driver 的驱动程序)。

1. 什么是 VxD

VxD(Virtual Device Driver)是用来扩展 Windows 操作系统功能的一类程序。VxD 最初用来支持硬件设备的管理，它以 DLL 的形式链入 Windows 操作系统的核心层(ring 0)。VxD 主要解决不能被 ring3 层应用程序处理的一系列问题，它在 Win3.x 和 Win9x 中被普遍地使用。

Win3.x 和 Win9x 系统的核心(kernel)由虚拟机管理器(VMM)和 VxD 的集合组成。Kernel 提供了 900 多个服务函数来管理内存、控制物理设备、处理中断、创建网络协议栈、管理文件系统等，这些服务函数都可以被您自己写的 VxD 调用。

2. 什么时候需要编写 VxD

当需要一个驱动程序来支持应用程序时，一般来说应编写 VxD。VxD 种类很多，不一而足，这里只列出不是 VxD 的设备驱动程序：

打印驱动程序运行在 ring3 层，它不是 VxD；
在 Windows 启动前装载的 DOS 驱动程序不是 VxD；
有些驱动程序能够调用 VxD 服务函数，但它们只是 ring3 层的 DLL，而不是 VxD。
此外的所有驱动程序全都是 VxD。

3. Win3.x 的 VxD 特点

Win3.x VxD 能够直接运行在 Win95 平台上，在 Win98 平台(作者还未曾用过)上应该也能够运行。以前的 VxD 全部用 Intel 汇编语言编写，是一项枯燥、艰苦的工作。本书有幸为您提供用 C++类库编写 VxD 的方法(否则作者也就不会写这样一本书了)。

4. Win95 的 VxD 特点

Win95 针对 VxD 进行了一系列的改进，主要是提供了对即插即用(PnP)的支持。此外，系统注册表、可分页寻址、动态加载和卸载都是 Win95VxD 的新特性。在 Windows for Workgroup 版本中出现的一些重要 VxD 改进，如保护模式文件系统(IFS)和新的串口驱动程序(VCOMM)，也被 Win95 系统所保留。

5. 本书采用的 VxD 编写工具

过去的 VxD 全部用 Intel 汇编语言编写，编程者需要有很好的 x86 汇编语言功底，还得对 Windows 体系结构有相当的了解。本书针对广大使用流行的 C++语言编程的程序员，并不打算在汇编语言和 Windows 体系结构上做过多讨论，因为它们都足以写厚厚的一本书。

您首先需要准备 MS VC++2.0 以上的 C++版本，Borland 公司的 C++4.0 以上版本也可以用。但本书代码只用 VC++编写，您不必为此担心，由于 VxD 代码并不涉及到 MFC，因此稍作修改即适用于 BC++。

其次，您需要准备美国 Vireo Software 公司推出的 VToolsD for Win95 开发工具包，您可以在http://www.vireo.com网址上获得关于这个软件的一切信息。

VToolsD 开发包提供了对 VxD 编程的全线 C++类库支持，有了 VToolsD，就不再需要 Microsoft 的 MS DDK(您不久就会发现：对比 VToolsD，MS DDK 太艰深了)。

在 Win95 平台上运行 Setup.exe，VToolsD 安装程序将生成 VToolsD for Windows95 文件夹。文件夹中最重要的是 QuickVxD 程序。它提供了诸多选项用来快速生成 VxD 代码框架，这与 VC++的 Class Wizard 是极为相似的。

第一章将介绍 QuickVxD 的基本使用方法。从第一章起，通过您的仔细阅读，您将一层层撩开 VxD 的神秘面纱。还等什么，让我们开始吧。

目　录

前言…… I

第 1 章　QuickVxD 的使用介绍……1

1.1　QuickVxD 具体操作方法……1

1.1.1　VxD 设备参数 (Device Parameters) ……1

1.1.2　应用程序调用接口 (Application Interfaces) ……3

1.1.3　VxD 服务 (VxD Services) ……4

1.1.4　VxD 的控制消息 (Control Messages) ……4

1.1.5　VxD 的主要类 (Classes) ……5

1.1.6　输出文件 (Output Files) ……5

第 2 章　框架类 (Framework Classes) ……7

2.1　VDevice 类……7

2.1.1　VDevice 类的基本编程方法……7

2.1.2　VDevice 类的主要成员函数……7

2.2　VVirtual Machine 类……10

2.2.1　类 VVirtual Machine 针对控制消息的主要成员函数……10

2.2.2　类 VVirtual Machine 其它成员函数……11

2.3　VThread 类……12

2.4　创建一个 “hello”VxD……13

2.5　在 MSVC 集成环境中创建 VxD……14

第 3 章　I/O 设备驱动程序的编写……16

3.1　类 VIOPort 的成员函数介绍……16

3.2　使用类 VIOPort……17

3.3　注意事项……18

3.4　创建一个使用 VIOPort 的 VxD……19

第 4 章　中断 VxD 的编程……23

4.1　硬件中断编程……23

4.1.1　VHardwareInt 类……23

4.1.1.1　VPICD 的介绍……23

4.1.1.2　类 VHardwareInt 主要成员函数……24

4.1.1.3　使用类 VHardwareInt……26

4.1.1.4　COM2 中断实例……28

4.1.2　VSharedHardwareInt 类……30

4.1.2.1 VSharedHardwareInt 类主要成员函数……30
4.1.2.2 使用类 VSharedHardwareInt……30
4.2 软中断编程……30
4.2.1 类 VPreChainV86Int……30
4.2.1.1 VPreChainV86Int 类主要成员函数……30
4.2.1.2 使用类 VPreChainV86Int……31
4.2.2 类 VInChainInt……32
第 5 章 DMA 设备驱动程序的编程……34
5.1 DMA 的有关知识……34
5.1.1 DMA 简介……34
5.1.2 VDMAD 简介……34
5.1.3 其它 VxD 对 DMA 通道的虚拟化……34
5.1.4 DMA 内存缓冲(buffer)和应用程序内存区(region) ……35
5.2 与 DMA 有关的类……35
5.2.1 DMAChannel 类……35
5.2.1.1 类 VDMAChannel 主要成员函数……35
5.2.1.2 使用类 VDMAChannel……38
5.2.2 VDMABuffer 类……39
5.2.2.1 类 VDMABuffer 主要成员函数……39
5.2.2.2 使用类 VDMABuffer……40
第 6 章 热键 VxD 的编程……42
6.1 VHotKey 类主要成员函数……42
6.2 使用类 VHotKey……44
6.3 一个捕捉 Ctl+C 热键的 VxD……45
第 7 章 TimeOut 设备驱动程序的编写……47
7.1 类 VTimeOut 及其派生类……47
7.1.1 VTimeOut 类主要成员函数……47
7.1.2 VGlobalTimeOut 类新的成员函数……47
7.1.3 VVMTimeOut 类新的成员函数……48
7.1.4 VAsyncTimeOut 类新的成员函数……48
7.1.5 VThreadTimeOut 类新的成员函数……48
7.2 使用类 VTimeOut 及其派生类……49
7.3 创建一个 Beeper 设备驱动程序……50
第 8 章 针对错误处理的设备驱动程序的编写……52
8.1 错误处理类介绍……52
8.1.1 VFault 类……52

8.1.1.1 VFault 类主要成员函数……52
8.1.2 VNMIEvent 类……53
8.1.2.1 类 VNMIEvent 新的成员函数……53
8.1.2.2 使用类 VNMIEvent……53
8.1.3 类 VProtModeFault……54
8.1.3.1 类 VProtModeFault 新的成员函数……54
8.1.3.2 使用类 VProtModeFault……54
8.1.4 类 VVMMFault……54
8.1.4.1 类 VVMMFault 新的成员函数……54
8.1.4.2 使用类 VVMMFault……54
8.1.5 类 VV86ModeFault……54
8.1.5.1 类 VV86ModeFault 新的成员函数……54
8.1.5.2 使用类 VV86ModeFault……55
8.1.6 类 VInvalidPageFault……55
8.1.6.1 类 VInvalidPageFault 新的成员函数……55
8.2 使用错误处理类……56
第 9 章 事件处理类设备驱动程序的编写……58
9.1 事件处理类……58
9.2 类 VDeviceAPI……58
9.2.1 类 VDeviceAPI 主要成员函数……58
9.2.2 使用类 VDeviceAPI……59
第 10 章 异步事件设备驱动程序的编写……61
10.1 类 VEvent……61
10.1.1 类 VEvent 的主要成员函数……61
10.2 类 VGlobalEvent……62
10.2.1 类 VGlobalEvent 新的成员函数……62
10.2.2 使用类 VGlobalEvent……63
10.2.3 一个 IRQ8 中断发声 VxD 的编写……64
10.3 类 VVMEvent……67
10.3.1 类 VVMEvent 新的成员函数……67
10.3.2 使用类 VVMEvent……67
10.4 类 VAppyTimeEvent……68
10.4.1 类 VAppyTimeEvent 的主要成员函数……69
10.4.2 使用类 VAppyTimeEvent……71
第 11 章 回调函数类……73
11.1 类 VCallbackv……73

11.1.1 类 VCallback 主要成员函数……73
11.2 类 VV86Callback……74
11.2.1 类 VV86Callback 新的成员函数……74
11.2.2 使用类 VV86Callback……74
11.3 类 VProtModeCallback……74
11.3.1 类 VProtModeCallback 新的成员函数……74
11.3.2 使用类 VProtModeCallback……74
11.4 中断处理中的回调函数……75
11.4.1 类 VInChainInt……75
11.4.1.1 类 VInChainInt 的主要成员函数……75
11.4.2 类 VInChainV86Int……75
11.4.2.1 类 VInChainV86Int 新的成员函数……76
11.4.2.2 使用类 VInChainV86Int……76
11.4.3 类 VInChainPMInt……77
11.4.3.1 类 VInChainPMInt 新的成员函数……77
11.4.3.2 使用类 VInChainPMInt……77
第 12 章 内存管理类……79
12.1 类 VPageObject……79
12.1.1 类 VPageObject 的主要成员函数……79
12.1.2 使用类 VPageObject……79
12.2 类 VLockedPageObject……80
12.2.1 类 VLockedPageObject 的主要成员函数……80
12.2.2 使用类 VLockedPageObject……80
12.2.3 类 VLockedPageObject 和类 VPageObject 的差别……81
12.3 更多的内存管理类……81
12.3.1 类 VGlobalV86Area……81
12.3.2 类 VPageBlock 和 VV86Pages……81
第 13 章 同步对象类……82
13.1 类 VSemaphore……82
13.1.1 类 VSemaphore 主要成员函数……82
13.1.2 使用类 VSemaphore……83
13.2 类 VMutex……84
13.2.1 类 VMutex 主要成员函数……84
13.2.2 使用类 VMutex……86
第 14 章 在 VxD 中使用注册表……88
14.1 类 VRegistryKey 主要成员函数……88

14.2 使用类 VRegistryKey……91
第 15 章 管道类……92
15.1 类 VPipe……92
15.1.1 类 VPipe 主要成员函数……92
15.1.2 使用类 VPipe……93
15.2 类 VDosToWinPipe……94
15.2.1 类 VDosToWinPipe 新的成员函数……94
15.2.2 使用类 VDosToWinPipe……95
第 16 章 设备驱动程序的 DPMI Vendor 调用入口……97
16.1 类 VV86DPMIEntry 和类 VPMDPMIEntry 介绍……97
16.1.1 类 VV86DPMIEntry……97
16.1.1.1 类 VV86DPMIEntry 成员函数……97
16.1.2 类 VPMDPMIEntry……98
16.1.2.1 类 VPMDPMIEntry 成员函数……98
16.2 使用类 VV86DPMIEntry 和类 VPMDPMIEntry……98
16.2.1 使用类 VPMDPMIEntry 和 VV86DPMIEntry……98
16.3 例 Classtut VxD……101
第 17 章 VxD 调用细则……115
17.1 从其它 VxD 中调用 VxD 函数……115
17.2 从 V86 代码或 Win16(保护模式)代码中调用 VxD 函数……118
17.3 Win32 代码调用 VxD 函数……119
17.4 利用 DPMI 方式调用 VxD 函数……121
附录 1 控制消息……123
附录 2 VxD 中的数据结构……125
参考文献……132

第 1 章　QuickVxD 的使用介绍

QuickVxD 利用 VToolsD 封装的 C++类库来快速创建 VxD 程序代码框架。使用 QuickVxD 的好处在于让编程者把精力放在 VxD 所需要完成的功能上，而不是把精力放在 VxD 编程的低级代码细节上。QuickVxD 提供一个比较简单的可视化编程环境，通过它，编程者能够创建功能完善的 VxD 应用框架。

在 QuickVxD 可视化编程环境中，QuickVxD 根据编程者的各种选择快速创建一个 VxD 工程文件。此工程文件包括：

1. C/C++头文件（.h 文件）；
2. C/C++代码文本（.c/.cpp 文件）；
3. C/C++工程文件（.mak）文件。

.h 文件含有 VxD 所必需的类 (Class) 声明，还有 VxD 控制消息处理函数声明，Windows ring3 API 调用 VxD 的调用入口声明以及 VxD 其它服务函数声明等必要声明。此外.h 文件还定义了许多全局变量和类变量，同时它为.mak 文件附加必要的编译参数。在.c/.cpp 代码文本中，QuickVxD 预先生成了许多须继续编程来扩充其内在功能的类成员函数。编程者通过在这些类成员函数中添加代码，从而完成定制功能。.mak 文件指导编译程序对源程序的编译方式。通过用 NMAKE 编译源程序，即可生成最终的 VxD 程序。

1.1　QuickVxD 具体操作方法

在 QuickVxD 可视化编程环境中，程序员只需在 VxD 快速生成对话框中选择和定义所需 VxD 的基本特征，即可自动生成 VxD 源程序框架。下面几节将详细介绍 VxD 快速生成对话框中的各选择或填写项。

1.1.1　VxD 设备参数 (Device Parameters)

图 1.1 为 QuickVxD 设备参数定义栏，主要设备参数有：

1. Device Name

Device Name 是所建 VxD 的设备名。每一个 VxD 都有它的设备名字。设备名不多于 8 个字符，其命名规则同 C 语言的命名规则。

2. Device ID

Device ID 指明所建 VxD 的设备标识。如果 VxD 提供了对其它 VxD 的调用入口，提供了对其它虚拟 8086 模式程序 API 调用入口或保护模式程序 API 调用入口，则 Device ID 是必须要具备的。

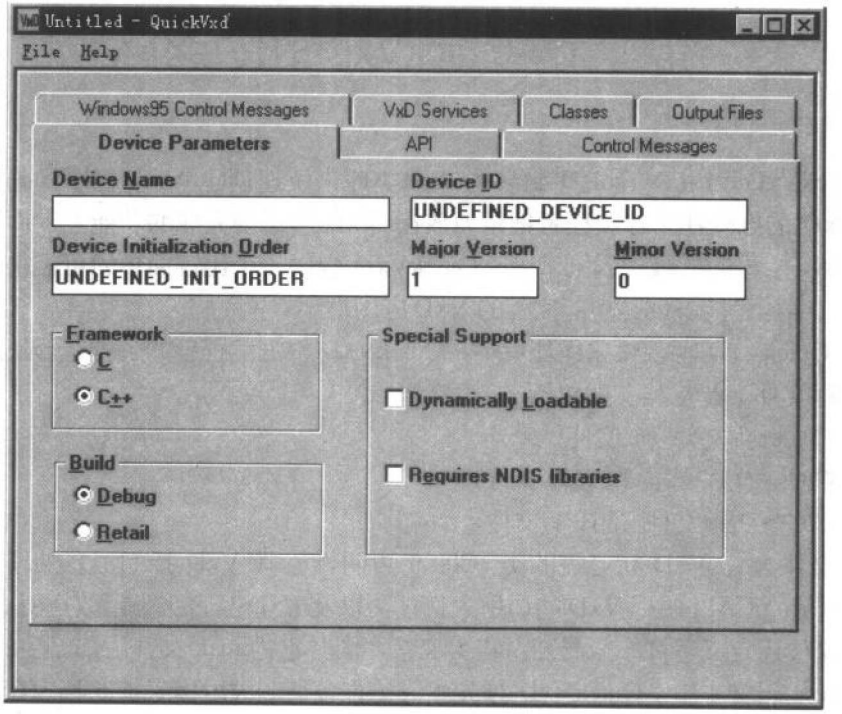

图 1.1　设备参数栏

Device ID 是一个 16 位的无符号整型数，不能任意取值，以免与其它 VxD 的 Device ID 相冲突。一般来说，应与 Microsoft 公司联系而获得一个唯一的 Device ID 值。

3. Device Initialization Order

Device Initialization Order 确定 Windows 对 VxD 的装载顺序。如果所建的 VxD 不依赖于其它 VxD 是否装载，则可以使用 VToolsD 提供的缺省值：UNDEFINED_INIT_ORDER。否则，Device Initialization Order 必须设置为某个已知 VxD 的装载顺序常数加(或减)1，加 1 代表所建 VxD 在已知 VxD 后装载，减 1 代表所建 VxD 在已知 VxD 前装载。Device Initialization Order 常数的形式为：xxxx_INIT_ORDER，其中 xxxx 表示已知 VxD 的设备名。

例如，若所建的 VxD 需要在虚拟显示设备驱动程序之后装载，则应设置此 VxD 的 Device Initialization Order 为 VDD_INIT_ORDER + 1。

4. C/C++ Framework

C/C++ Framework 选项决定了 VxD 源程序的代码生成类别。本书只论述 VxD 的 C++ 编程。

C++ Framework 包含了一整套 VxD 类库。通过从类 VDevice 和类 VVirtualMachine 派生而实现大量的 VxD 控制消息处理和提供 VxD 调用入口服务。C++ Framework 中的

其它类库则封装了 VMM/VxD 其它各方面的功能（如 I/O、DMA、Interrupt 等功能）。

编程者在 VxD 快速生成对话框中填写好 VxD 的 Device Name，并选择 C++ Framework，则 QuickVxD 将自动根据 Device Name 而生成设备类 (device class)，以及虚拟机类 (virtual machine class)。

5. Dynamically Loadable

Dynamically Loadable 选择项将决定所建的 VxD 是否可以在应用程序中动态加载，VxD 动态加载是 Win9x 才有的新特性。应注意的是，Win3.x 不支持 VxD 的动态加载。动态加载 VxD 的典型例子是对即插即用 (PnP) VxD 的加载。

6. Version

Version 填写项决定所建 VxD 的版本号。Version 包括 Major Version 和 Minor Version 两项。缺省情况下的版本号为 1.0，即 Major Version 为 1，Minor Version 为 0。Major Version 和 Minor Version 的取值不能大于 255。

1.1.2 应用程序调用接口 (Application Interfaces)

QuickVxD 采用下面几种方式为应用程序提供从 Ring3 至 Ring0 的调用接口，如图 1.2。

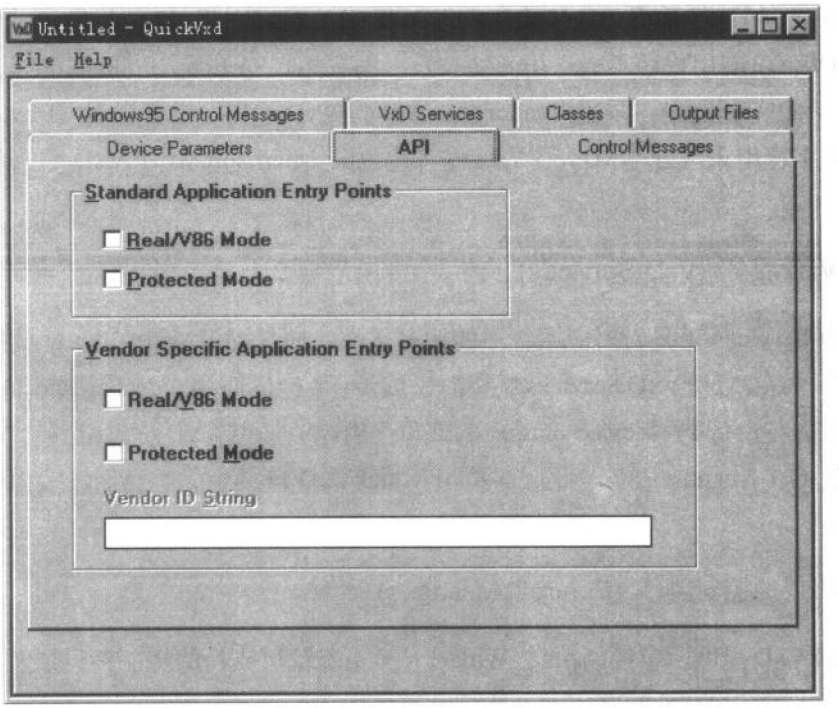

图 1.2 API 调用入口栏

1. 虚拟 8086 模式程序 API 调用入口 (V86 Entry Point)

V86 Entry Point 为虚拟 8086 模式程序调用 VxD 服务提供了调用入口点。在 Win9x 版本以下的 Windows 环境中，VxD 采用 V86 Entry Point 为虚拟 8086 模式应用程序提供服务是 Windows 推荐的调用机制之一。调用入口函数的形参将包含一个指向虚拟 8086 模式应用程序各个寄存器状态的指针。通过分析这些寄存器的值，VxD 就可以为 V86 模式应用程序提供相应的服务。

在 QuickVxD 中选取 Real/V86 Mode 检查框，则 QuickVxD 将自动生成 V86 Entry Point 函数原型。

2. 保护模式程序调用入口 (PM Entry Point)

PM Entry Point 为保护模式程序调用 VxD 服务提供调用入口点。VxD 采用 PM Entry Point 为保护模式应用程序提供服务是 Windows 推荐的调用机制之一。调用入口函数的形参将包含一个指向保护模式应用程序各个寄存器状态的指针。通过分析这些寄存器的值，VxD 就可以为保护模式应用程序提供相应的服务。

在 QuickVxD 中选取 Protected Mode 检查框，则 QuickVxD 将自动生成 PM Entry Point 函数原型。

3. DOS 保护模式调用入口 (Vendor v86 API)

Vendor v86 API 采用 DPMI 协议为 V86 程序以及 16 位和 32 位保护模式程序提供调用入口 (Win32 应用程序不能采用此协议)。与采用 V86 Entry Point 或 PM Mode entry point 的 VxD 不同的是，采用 Vendor v86 API 的 VxD 不需要设备标识 (Device ID)，它使用一个字符串来区分 VxD。这为编程者避免向 Microsoft 申请 VxD 设备标识提供了方便。

1.1.3 VxD 服务 (VxD Services)

VxD Services 不是针对应用程序的服务函数，而是提供给其它 VxD 的服务函数。在采用 C++框架下，VxD Services 将以静态成员函数 (同时用_cdecl 或_stdcall 方式声明) 的形式出现在设备类 (device class) 的成员函数中。通常 VxD 提供给其它 VxD 的第一个服务是 Get_Version 服务，此服务将由 QuickVxD 自动生成。当然，编程者可以不提供这个服务。

1.1.4 VxD 的控制消息 (Control Messages)

QuickVxD 提供两类控制消息：Win3.1 和 Win95 都可支持的控制消息以及只被 Win95 支持的控制消息 (如图 1.3)。

在 C++代码框架下，每一个控制消息有相对应的类成员函数。这些成员函数出现在 device class、virtual machine class 或 thread class 类中。

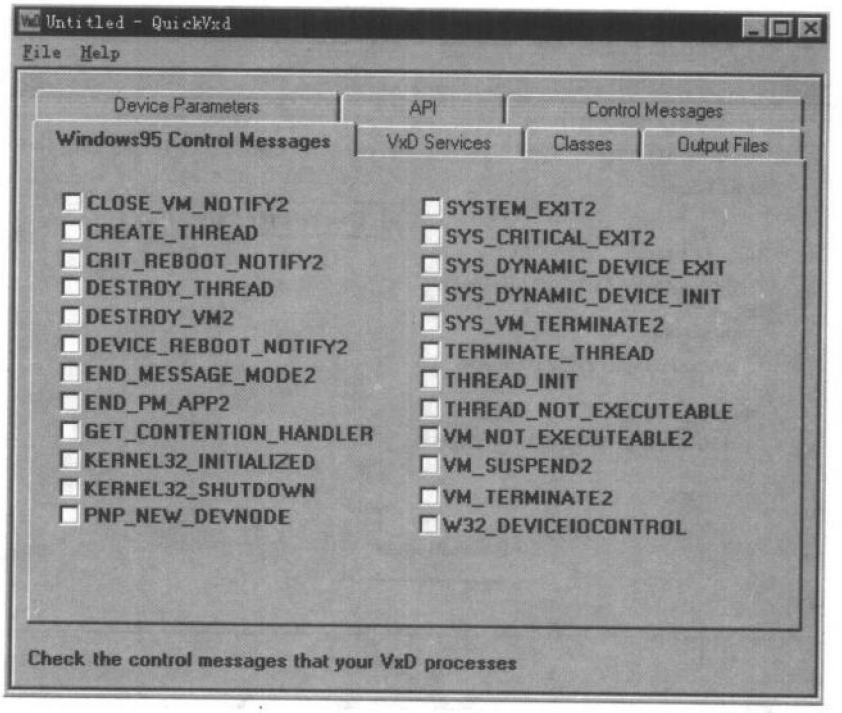

图 1.3 Win95 VxD 控制消息

1.1.5 VxD 的主要类 (Classes)

在 C++代码框架下，QuickVxD 最多创建三个类来处理可能的控制消息 (Control Messages)。

1. Device Class

Device Class 代表虚拟设备驱动程序类。大部分的控制消息由它的成员函数来处理，V86 模式程序调用入口和保护模式程序调用入口亦在其中。

2. VM Class

VM Class 代表虚拟机类。一些特定的虚拟机控制消息由它的成员函数来处理。

3. hread Class

Thread Class 处理系统中某些特定线程的控制消息。

1.1.6 输出文件 (Output Files)

程序员点按 Generate Now 按钮，QuickVxD 将在校验所有选择项和填写项无误的情况下自动生成输出文件，若检验有误，QuickVxD 会提供出错信息。

输出文件包含.h 文件、.cpp 文件以及.mak 文件 (如图 1.4)。它们的具体内容见本章开头。利用这几个文件，编程者可以在 MS VC++集成环境中添加自己的代码，从而得到完整的 VxD 应用。

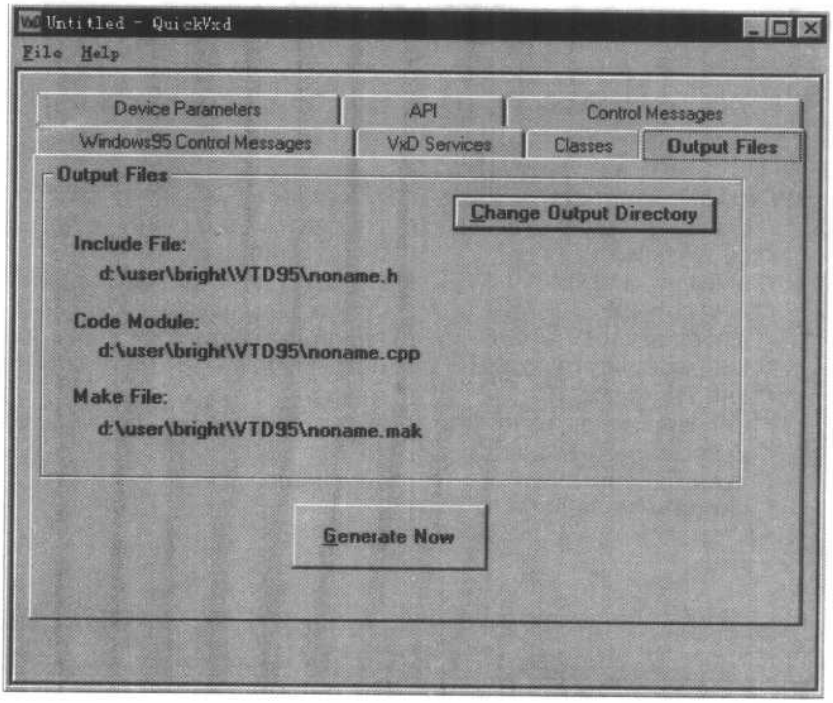

图 1.4　QuickVxD 的文件输出栏