简　介

Sound Blaster 系列开发工具的使用对象是二次开发者或者想要在 Creative 实验室的 Sound Blaster Audio Card（简称 SBC）系列下开发自己软件产品的用户。SBC 系列包括：

- Sound Blaster 1.5 版或其以下版
- Sound Blaster 2.0 版
- Sound Blaster MCV（微通道版）
- Sound Blaster Pro

本开发工具为开发人员提供了易于使用的函数和完整的接口，它支持数字化声音、FM 合成音乐、文本合成语音、混音器控制、MIDI 接口以及 CD-ROM 驱动器音频接口。同时还提供了 Sound Blaster 音频卡系列的硬件编程信息。

对于 DOS 程序设计人员面临的共同问题，本书格外予以关注，以满足各种类型开发者的需求，并力争减少同其它 TSR（内存驻留程序）的冲突。

SBC 提供的驱动程序是外部驱动程序，分为装载和驻留两种方式。这样做的原因是为了适应各种各样的硬件支持。同时驱动程序是以一个个独立文件的形式给出的，这样，应用程序就可以和驱动程序分离开来（即应用程序可不考虑驱动程序版本的变化）。

数字化声音输入输出（以后简写为 I/O），有三种途径：基本内存方法、扩展内存（Extended Memory）方法和磁盘双缓冲区方法。这些方法使得声音 I/O 不受软件限制，声音的记录仅受限于内存和磁盘空间。

本开发工具支持下列语言：

- Microsoft 汇编 4.0 及其以上版本
- Microsoft C 5.0 及其以上版本
- Turbo C 2.0 版
- Microsoft Quickbasic 4.5 版
- Microsoft Basic 专业级开发系统（PDS）7.0 版
- Turbo Pascal 6.0

一、本书约定

本书中“开发人员”、“程序员”或“读者”含义相同，均指使用本开发工具编写程序的人员；“应用程序”指开发人员用本开发工具所编写的程序；“用户”指所开发应用程序的使用者。

本书中使用了以下缩写：

1. “SB”指 Sound Blaster 1.5 或其以下版本
2. “SB20”指 Sound Blaster 2.0 版
3. “SBMCV”指 Sound Blaster 微通道（MCV）版
4. “SBPRO”指 Sound Blaster Pro
5. “SBK”指本开发工具

6．“DSP”指 Creative 的数字化声音处理器

“SBC”指包含上述 1～4 的 Sound Blaster 音频卡系列。

“定时器 0”指的是系统定时器，一般情况下，SBC 库函数为了满足定时要求，将为系统定时器编程。DSP 的其它定时器用来控制数字化声音的采样率和 MIDI 定时。

在本书中 SBK 函数以下述形式给出：

句法：

解释：

返回值：

参见

适于：

二、如何使用本书

本书假设读者是一个有经验的程序员并且熟悉 Sound Blaster 音频卡系列或 SBC 派生音频卡的使用．因此所讨论的问题集中在技术方面。

本书由三篇构成：

第一篇为编程参考。描述了如何通过调用驱动程序及库函数，实现 Sound Blaster 视频卡系列的以下功能：

- 声音的输入输出
- FM 音乐输出
- 文本合成语音
- MIDI 的输入和输出
- 混音器控制
- CD-ROM 驱动器音频接口

同时解释所支持的程序设计语言和驱动程序、库函数的接口。较详尽地描述了驱动程序和库函数。所提供的样例程序已经过编译、连接并运行通过。

第一篇包含下列内容：

第一章：使用 SBK 时的一般编程信息。

第二章：在 SBC 上运行程序时所需要的基本函数。

第三章：如何使用 Creative 的装载式和驻留式驱动程序。

第四章：介绍 VOC 文件格式和 Creative 的声音驱动程序的接口：CT-VOICE 和 CTVDSK。如果要用 SBC 对声音进行数字化并输出声音或只输出声音，阅读这章是必不可少的。

第五章：介绍 CMF 文件格式和 SBFMDRV 驱动程序接口。如果要播放 FM 音乐，必须阅读本章。同时，在这一章中，把 SBFMDRV 的函数被分为高级和低级存取函数。

第六章：在程序中如何实现文本到语音的合成。

第七章：MIDI 接口函数，当应用程序要用到 SBC 对 MIDI 接口时，本章是必须阅读的。

第八章：辅助驱动程序接口和函数。

第九章：CD ROM 音频接口。

第二篇为硬件技术参数。描述了硬件说明以及 Sound Blaster 音频卡系列的程序设计，包括以下内容：

第十章：SBC I/O 地址表。

第十一章：如何对 DSP 进行数据读写，认真读这一章可了解 DSP 的特性。

第十二章：如何在 DSP 上对数字化声音输入输出进行编程。

第十三章：MIDI 端口的 DSP 接口。

第十四章：DSP 命令。

第十五章：FM 合成芯片的硬件编程。

第十六章：混音器芯片的硬件编程。

第十七章：游戏杆端口规格。

第三篇为附录，讨论了 Basic 补充函数、DMA 控制器的一些编程技巧，并且概括了 SBC 库函数。

三、SBK 版本

为了帮助读者了解所使用的 SBK 库的版本，在所提供的软盘上给出了下述文件：

\BASIC\SBKVER.BAS

\MSC\SBKVER.C

\TC\SBVER.C

\TPASCAL\SBKVER.PAS

它们放在目录中，编译和执行后，就可知道 SBK 库版本。

SBK 适用于整个 Sound Blaster 音频卡系列，某些 SBC 卡还具有其它卡所不具备的附加特性。比如，混音器控制和 CD-ROM 驱动器接口就只适用于 SBPRO。同样，一些 SBK 函数只适于某个特殊的卡，参看函数参数中适用性一栏（“适于”），它指出了函数所适用的卡（其中“■”表示有效；“□”表示无效）。

在应用程序中，读者可以调用任何一个函数，而不用担心 SBC 驱动程序是否支持它。因为不被支持的函数的返回是一个空返回。读者可以检查 BLASTER 环境（参见 SOUND 和 BLASTER 环境）以决定应用程序是在哪种 Sound Blaster 卡下工作的。

四、最后的信息

请阅读所提供的软盘上的 README.TXT 文件（键入 README 即可），其中包含了没有来得及印刷的有关 SBK 的最后信息。

第一篇　编程参考

此为试读，需要完整PDF请访问：www.ertong

第一章 编程概述

本章介绍各种库和包含文件，讨论在使用 SBK 库时应了解的信息。

1.1 所支持的编程语言

SBK 支持下列编程语言：

- Microsoft Assembler 4.0 版或以上版
- Microsoft C 5.0 版或以上版
- Turbo C 2.0 版
- Microsoft QuickBasic 4.5 版
- Microsoft Basic PDS(专用开发系统)7.0 版
- Turbo Pascal 6.0

Microsoft C 和 Turbo C 使用了相同的 C 库和包含文件，它们在 SBK 软盘上。在以后讨论中所使用的 C 是 Microsoft C 。如果应用程序是用 Turbo C 编写的，那么函数包含文件和全局变量的使用同 Microsoft C 中的使用方式一样。

SBK 软盘中提供的 Basic 样例程序，可以在 QuickBasic 4.5 版和 Basic PDS 7.0 版环境下使用。

1.2 使用 SBC C 目标库

SBC C 目标库支持下列四种内存模式：

- SBCSR.LIB 小模式
- SBCMR.LIB 中模式
- SBCCR.LIB 压缩模式
- SBCLR.LIB 大模式

1.2.1 如何开始

在编译应用程序之前，把 C 的包含文件和库拷贝到所使用的 C 编译器的包含文件和目录之下。

在以下的讨论中，DEMOVDP 指的是用户程序，该程序的源代码在 SBK 软盘中。

1.2.2 Microsoft C

若使用 Microsoft C，那么分别将包含文件拷贝到包含文件目录(缺省为\INCLUDE)之下，库拷贝到库目录下(缺省为\LIB)，同时需要设置 INCLUDE 和 LIB 环境，以使 Microsoft C 编译器能读取这些文件。例如，“include”和“library”目录在 C 盘根目录下的 \INCLUDE 和\LIB 路径下，就要进行以下设置：

```
SET INCLUDE=C:\INCLUDE
SET LIB=C:\LIB
```

在小模式下编译和连接应用程序(以 PEMOVDP 为例)，只需在 DOS 命令行下键入：

```
CL/c DEMOVDP.C
LINK DEMOVDP,,,SBCSR.LIB
```

1.2.3 Turbo C

同样地，需要分别把包含文件拷入包括文件目录下(缺省为\TC\INCLUDE)，库拷入库目录(缺省为\TC\LIB)下。

有两种方式编译和连接应用程序：

1. 在集成环境下

程序员需要建立 Turbo C 工作路径，比如，包含文件和库目录在 C：\TC\INCLUDE 和 C：\TC\LIB 下，则在 Option 菜单下，选择 Directories，然后设置 Include directories 为 C：\TC\INCLUDE，Library directories 为 C：\TC\LIB。

为了编译和连接应用程序，应创建工程(Project)文件。它由下列两行组成：

```
DEMOVDP
SBCSR.LIB
```

然后指定该工程文件，并且按 ALT+R。

2. DOS 命令行方式下

键入命令：TCC-I\TC\INCLUDE-L\TC\LIB DEMOVDP SBCSR.LIB

如果包含文件和库所在的目录与上述不同，则需重新建立正确的包含文件和库的路径名。

注意：在使用时，库要和内存模式及驱动程序方式相对应。

1.3 使用 SBC Basic 库

SBK 支持 QuickBasic 4.5 版和 Basic PDS 7.0 版，其库名如下：

```
4.5版   QBSBC.QLB
        QBSBC.LIB
7.0版   QB7SBC.QLB
        QB7SBC.LIB
```

在编译 Basic 程序之前，需要把 Basic 包含文件和库拷贝到 Basic 编译器所存取的包含文件和库目标下，同时程序员还必须设置包含文件(INCLUDE)和库的(LIB)环境。例如，包含文件和库目录是 C 盘根目录下的\INCLUDE 和\LIB，那么需设置环境：

```
SET INCLUDE=C:\INCLUDE
SET LIB=C:\LIB
```

编译和连接有两种方法：

1. Basic 环境

启动 Basic，使用 SBC Basic Quick 库编译 DEMOVDP 程序的操作是：

对于 4.5 版，键入：

```
QB/Ah DEMOVDP/L QBSBC.QLB
```

对于 7.0 版，键入：

```
QBX/Ah DEMOVDP/L QB7SBC.QLB
```

然后按 ALT+R，并选择 Make EXE File 选项开始编译。

2. 命令行

在 DOS 命令行键入以下命令：

对于 4.5 版：

```
BC DEMOVDP/Ah/O
LINK DEMOVDP,,, QBSBC.LIB;
```

对于 7.0 版：

```
BC DEMOVDP/Ah/Lr/Fs/O
LINK DEMOVDP,,, QB7SBC.LIB;
```

注意：/Ah 命令选项仅用在下述情况，即任何一个数组占用了多于 64KB 的空间或其它元素超过 16384(因为每个整数占 4 个字节)。

1.4 使用 SBC Turbo Pascal Unit (TPU)

SBK 支持 Turbo Pascal 6.0。TPU，SBC_TP6.TPU 提供了对 SBC 驱动程序的编程接口。

程序员应在 Turbo Pascal 目录(缺省为\TP)下创建一个子目录，并且把 SBK 软盘提供的包含文件 TPU 拷入，下面的讨论假设这个子目录叫\SBK-TP。

程序员还需建立 Turbo Pascal 工作目录，以便编译器能存取这些文件和部件。例如，使用集成环境时，需在 Option 菜单下的 Directories 子菜单中，建立包含文件和部件目录。如果在 DOS 命令行编译，则需使用包含命令选项/I 和部件命令选项/U，来指定包含和部件的路径名如下：

```
TPC/I\TP\SBK-TP/U\TP\SBK-TP (库名)
```

当然，还可以连接路径\TP\SBK-TP 到包含和部件目标，以建立 TPC.CFG 文件。建议在包含和部件目录为：

```
/I\TP;\TP\BGI;
/U\TP;\TP\BGI;
```

连结后如下：

```
/I\TP;\TP\BGI;\TP\SBK-TP;
/U\TP;\TP\BGI;\TP\SBK-TP;
```

在所有的应用程序中，必须把：

```
USES SBC_TP6;
```

放在应用程序开头，以便告诉编译器，应用程序使用了这个部件，如：

```
Program SBKver;
USES sbc_tp6;
Begin
```

```
        .
        .
        .
    End
```

有两种编译和连接方法：

1．集成环境

为了启动 Turbo Pascal 和编译 DEMOVDP，键入：

```
TURBO DEMOVDP
```

然后按 ALT+C 开始编译。

2．命令行

在 DOS 命令行键入下列命令：

```
TPC/I\TP\SBK-TP/U\TP\SBK-TP DEMOVDP
```

1.5 使用包含文件

在包含文件中声明了：

- 常量
- 全局变量
- 函数或过程
- 结构类型定义

编译和连接应用程序时必须包含该类文件，如要包含 SBCVOICE，必须使用：

对于 C：#include＜sbcvoice.h＞

对于 Basic：'$ INCULUDE:'SBCVOICE.BI'

对于 Turbo Pascal：{$I SBCVOICE.INC}

1.6 用汇编语言编程

所有的 SBC 驱动程序都是用汇编语言编写的。数据通过寄存器在驱动程序和应用程序之间进行交换。在以后的章节中详细讨论寄存器级的驱动程序接口。

1.7 用 Miscrosift C 和 Turbo C 编程

本开发工具为 C 程序设计员提供了四种内存模式下的库。象 Miscosoft C 5.0 定义的缺省段和组，就用在这些 SBC 库中。库中的全局变量，用缺省的数据段"_DATA"来说明。

1.8 用 Microsoft QuickBasic 和 Basic PDS 7.0 编程

关于各种 SBC Basic 库的讨论中，假设程序员已具有下述 DOS 功能调用(通过 INT 21H)的知识。

1. DOS 文件系统和文件句柄的概念。

2. 创建，打开，关闭，读和写一个文件。

3. 移动文件读和写指针。

4. 分配和释放内存。

5. 获取 DOS 扩充错误代码。

如果程序员在 SBK 中遇到任何一个涉及到上述功能的问题，建议查阅 DOS 3.0 或以上版的技术参考手册。

缺省情况下，Basic 建立后占用所有剩余内存，如果应用程序使用 DOS INT 21H 分配内存，那么需要使用 Basic 的 SETMEM()函数，释放足够的内存，以使 DOS 成功地分配内存。

Creative 的磁盘双缓冲声音驱动程序(CTVDSK.DRV)使用 DOS 的功能为两个磁盘缓冲区分配内存。当使用 CTUDSK.DRV 驱动程序时，记住要调用 Basic SETMEM()函数释放足够的内存，这样 DOS 才能成功地分配内存。

为了帮助程序员理解如何从 Basic 程序中调用 DOS 功能，厂家在 SBK 软盘的\BASIC 目录下给出了样例程序。

1.9 用 Turbo Pascal 6.0 编程

在 Turbo Pascal 编程的讨论中，假设程序员已具有 DOS 功能调用(通过 INT 21H)的知识。

Turbo Pascal 缺省内存分配 16K 的栈和 640K 的堆，这就是说堆将占用剩余内存。因此，如果使用 DOS 分配内存，应用程序必须释放一些堆内存(使用{$M}指令减少最大堆)，这样 DOS 才能成功地分配内存。

Creative 的磁盘双缓冲区声音驱动程序(CTVDSK.DRV)使用 DOS 的功能为两个磁盘双缓冲区分配内存，当使用 CTVDSK.DRV 驱动程序时，记住要使用 Turbo Pascal 内存分区指令{$M}释放足够的堆内存，以便 DOS 成功地分配内存。当然也可使用 ctvd_buffer_addx()过程进行磁盘缓冲区内存分配。

为了帮助程序员理解 Turbo Pascal 应用程序如何调用 DOS 功能，厂家在 SBK 软盘的\TPASCAC 目录下提供了样例程序。

此为试读，需要完整PDF请访问：www.ertong

第二章 基本函数

本章讨论在 SBC 上运行的所有程序都适用的基本函数和全局变量。介绍 SBC 应用的两个重要环境变量：SOUND 和 BLASTER。

仔细阅读本章，以便在编写应用程序之前，熟悉 SOUND 和 BLASTER 两个环境变量以及基本函数。

2.1 SOUND 和 BLASTER 环境变量

正确地使用 SOUND 和 BLASTER 对驱动程序的定位和 SBC 硬件配置是非常重要的，因为应用程序需要读取这些变量。以后的应用程序假设用户已建立了环境变量。

2.1.1 关于 SOUND 环境变量

SOUND 指明了 SBC 驱动程序和其它软件的目录位置，应用程序为了寻找 SBC 驱动程序和其它的 SBC 文件，需要检查这些环境设置。

设置 SOUND 变量的格式如下：

SET SOUND = <path>

通常如下设置(假设 SBC 文件已安装在 C 盘上)：

对于 SBPRO：SET SOUND = C:\SBPRO

对于其它 SBC：SET SOUND = C:\SB

注意：在"="号前后没有空格。

所有可装载驱动程序，都放在 SOUND 路径下的\DRV 子目录中。例如：若 SOUND 环境变量是 C:\SB，那么可装载驱动程序就在 C:\SB\DRV 下。

2.1.2 关于 BLASTER 环境变量

BLASTER 指明了 I/O 地址、中断和 SBC 的 DMA 通道的硬件配置。它反映了用户当前的 SBC 硬件配置。

设置 BLASTER 命令格式如下：

SET BLASTER = A220 I7 D1 T1

这里，A　I/O 基址

I　中断请求号

D　DMA 通道号

T　SBC 卡的类型

其中，1　SB 或 SBMCV(缺省值)

2　SBPRO

3　SB20

注意：在"="号前后没有空格，但在两个设置之间必须有至少一个空格。

2.2 包含文件和全局变量

在应用程序中，必须使用下述包含文件：

对于 C：#include＜sbc.h＞

对于 Basic：'$ INCLUDE:'SBC.BI'

在包含文件中声明了基本 SBC 库函数和全局变量。对于 C，重要的全局变量如下：

unsigned int ct_io_addx：SBC 的 I/O 基址

unsigned int ct_int_num：SBC 所用的中断

unsigned int ct_dma_channel：SBC 所用的 DMA 通道

unsigned int wCard ID：SBC 卡类型

对于 Turbo Pascal，这些变量是 _ct_io_addx, _ct_int_num, _ct_dma_channel 和 _wCardID。它们在 SBC_TP6.TPU 作了声明。

对于 Basic，这些变量是 IOADD%, INTNUM%, DMACHL%和 CARDID。它们在 SBC.BI 作了声明。

在以后的讨论中，将用 ct_io_addx, ct_int_num, ct_dma_channel 和 wCardID 为代表来说明这些变量。对于 Turbo Pascal 和 Basic 用法相同。

应用程序需要调用函数 GetEnvSetting() 来建立上述变量。这个函数的作用是从 BLASTER 环境设置中检索硬件配置，并把结果赋给上述变量。

2.3 库函数

2.3.1 GetEnvSetting

句法：

C　　unsigned GetEnvSetting(void)

Basic　FUNCTION SBGETENV%()

Pascal　funtion GetEnvSetting:word

解释：

GetEnvSetting 函数检测 BLASTER 环境设置并建立以下变量：

ct_io_addx

ct_int_num

ct_dma_channel

wCardID

若 BLASTER 环境变量没有指明卡类型，则使用缺省卡类型 1，即 wCardID 为 1。

返回值：

若成功，返回 0 值；否则返回一错误码：

1　BLASTER 环境变量没有设置

2　I/O 基址没有设置

3　中断号没有设置

4　DMA 通道没有设置

参见：

无

适于：

■SB ■SB20 ■SBMCV ■SBPRO

2.3.2 sbc_scan_card

句法：

C unsigned sbc_scan_card(void)

Basic FUNCTION SBSCNCRD%()

Pascal funtion sbc_scan_card:word

解释：

sbc_scan_card 函数检测 SBC 是否存在并建立 I/O 基址。

警告：为了进行测试，该函数要在I/O地址2x6H和2x7H上写数据。这样也许会由于其它卡使用了这些地址而出现问题。

注意：该函数用以维护Sound Blaster SBK 的兼容性。在一个新的应用中，应使用GetEnvSetting() 和 sbc_check_card()函数来检查 SBC 是否存在(参看例子)。

返回值：

该函数返回一个字：

bit 0,3～7：空闲

bit 1：使用了 FM 音乐芯片

bit 2：使用了 DSP 和声音 I/O

该函数也在 ct_io_addx 中建立了 SBC 的 I/O 基址。

参见：

sbc_scan_int，sbc_check_card

适于：

■SB ■SB20 ■SBMCV ■SBPRO

2.3.3 sbc_scan_int

句法：

C unsigned sbc_scan_int(void)

Basic FUNCTION SBSCNINT%()

Pascal function sbc_scan_int:word

解释：

sbc_scan_int 函数检查SBC 上的中断设置，并修改全局变量ct_int_num 的值。在该调用函数之前，必须先建立变量 ct_io_addx。

如果仅用SBFMDRV 播放FM音乐，那么就不需要调用这个函数。因为FM音乐不使用中断或 DMA。

注意：该函数用以维护sound Vlaster SBK 的兼容性，在一个新的应用中，应使用GetEnvSetting() 函数来获取中断设置，使用 svb_test_int()函数来测试中断(参看例子)。

返回值：

返回 SBC 中断号，同时修改全局变量 ct_int_num 的值。

若返回值是 0，应用程序应终止，因为不能检测到中断。

参见：

sbc_scan_card，sbc_scan_int

适于：

■SB ■SB20 ■SBMCV ■SBPRO

2.3.4 sbc_check_card

句法：

C unsigned sbc_check_card(void)

Basic FUNCTION SBCHKCRD%()

Pascal function sbc_check_card:word

解释：

sbc_check_card函数检测ct_io_addx中的I/O基址设置，并检测硬卡上的声音和音乐线路，该函数除了不能自动检测 SBC 外，几乎和 sbc_scan_card() 函数一样。

返回值：

返回一个字：

bit 0,3～7：空闲

bit 1：FM 音乐芯片正在使用

bit 2：DSP 和声音 I/O 正在使用

参见：

sbc_scan_card

适于：

■SB ■SB20 ■SBMCV ■SBPRO

2.3.5 sbc_test_int

句法：

C unsigned sbc_test_int(void)

Basic FUNCTION SBTSTINT%()

Pascal function sbc_test_int:word

解释：

sbc_test_int函数检查ct_int_num中的中断设置。应用程序需要调用这个函数，以确保 SBC 上中断成功。在调用该函数之前，必须先建立变量 ct_int_num。

返回值：

若成功，返回中断号；否则返回 0 值。

参见：

sbc_scan_int

适于：

■SB ■SB20 ■SBMCV ■SBPRO

2.3.6 sbc_test_dma

句法：

C int sbc_test_dma(void)

Basic FUNCTION SBTSTDMA%()

Pascal function sbc_test_dma:integer

解释：

sbc_test_dma函数检查ct_dma_channel中的SBCDMA通道设置。应用程序需要调用这个函数，以确保 SBC 上 DMA 通道能够工作。在调用该函数之前，必须先建立 ct_dma_channel 变量。

返回值：

若成功，返回 DMA 通道设置；否则返回 1。

参见：

无

适于：

■SB ■SB20 ■SBMCV ■SBPRO

2.3.7 sbc_version

句法：

C unsigned sbc_version(void)

Basic FUNCTION SBVERSION%()

Pascal function sbc_version:word

解释：

sbc_version 函数返回 SBC 上 DSP 的版本号。

返回值：

返回值是一无符号整型值，高位字节和低位字节分别表示主版本号和次版本号。

参见：

无

适于：

■SB ■SB20 ■SBMCV ■SBPRO

2.3.8 sbc_dsp_resct

句法：

C unsigned sbc_dsp_reset(void)

Basic FUNCTION SBRSTDSP%()

Pascal function sbc_dsp_reset:word

解释：

sbc_dsp_reset 函数重置 SBC 上的 DSP。

在sbc_scan_card()函数中也执行了DSP的重置。如果程序没有失去对DSP的控制，并且不计划重新初始化，那么就不必调用该函数。

该函数关闭了扬声器。

返回值:

返回非0值表明重置错。这种情况的发生也许是由于ct_io_addx没有被正确地设置。

参见:

sbc_scan_card, sbc_scan_int

适于:

■SB ■SB20 ■SBMCV ■SBPRO

2.4 样例程序

2.4.1 C样例程序

```
/* ---------------------------------------------------------------------- */
/* @@ Source Documentation                          *** MSC/TC Version ***  */
/*                                                                        */
/* Copyright (c) Creative Technology Pte Ltd, 1991. All rights reserved.  */
/*                                                                        */
/*  TITLE       : SBCBLST.C                                               */
/*                                                                        */
/*  DESCRIPTION :                                                         */
/*    The program checks the BLASTER environment variable for the Card    */
/*    settings. It also performs a test based on the BLASTER environment  */
/*    settings to ensure they tally with the hardware settings on the     */
/*    Card.                                                               */
/*                                                                        */
/* ---------------------------------------------------------------------- */

#include <stdio.h>
#include <sbc.h>


char *CardType[] = {
    "Sound Blaster",
    "Sound Blaster Pro",
     "Sound Blaster 2.0"
 };


 main ()
 {
```


此为试读,需要完整PDF请访问: www.ertong