瑞博华公司软件编程指南

北京瑞博华控制技术有限公司

2004 年 4 月

目录

瑞博华公司软件编程指南

北京瑞博华控制技术有限公司

2004 年 4 月

一、瑞博华公司产品概貌

北京瑞博华控制技术有限公司提供各种数据采集、DA、信号发生器、数字量输入输出的板、卡、模块和相关的驱动程序、软件模块、应用程序。硬件采用 ISA、PCI、USB、串口等各种接口形式。支持 DOS、Windows 9x/Me/2000/XP 等运行环境。支持 Visual Baisc、Delphi、 C++Builder、Visual C++、LABVIEW/LabWin 等开发环境。

瑞博华公司开发的各类 AD 采集板都配 Windows 9x/Me/2000/XP 下使用的驱动程序，应用程序都最终调用本公司提供的 ADCARD.DLL，然后在 ADCard.DLL 内部再调用其它内核驱动程序。本驱动程序以用户易于理解和使用的方式提供给用户，使用户以简单、高效地方式操作 AD 板，而不用处理底层的硬件细节。而且，VB 编程用户还可以使用 ActiveX 控件方式的驱动程序，即调用 ADCardX.OCX，通过它再调用 ADCard.DLL。ADCard 提供的各类函数在 C 语言头文件 ADCard.H 中有完整定义。驱动程序分为 Windows 9x/Me 下专用的 VXD 类驱动程序和 Windows 98/Me/2000/XP 下通用的 WDM 驱动程序，这两类驱动使用的文件不同，但同样提供 ADCard.DLL 驱动，且 ADCard.DLL 提供的接口函数完全相同，因此，通过使用通用的 ADCard 接口，同一应用程序可不用修改就在 Windows 95/98/Me/2000/XP 下运行，并可以支持各种瑞博华公司的采集卡，正如瑞博华公司的网络录波器软件一样。

AD 板提供的 ADCard 驱动程序不但提供对 AD 板操作的各类函数，还提供通用的对硬件端口的操作函数 inport/outport 系列，他们可以避开 Windows 对硬件的屏蔽，直接操作各类硬件。同时，为未安装本公司的 AD 板驱动程序的用户，还提供通用的 IOTOOLS 驱动程序，专门提供 Windows 9x/Me/2000/XP 下的硬件端口的操作函数 inport/outport 系列。

瑞博华公司的 PCI 接口的开关量板也提供了专门的驱动 RBHIO.DLL，它通过对 PCI 接口卡的管理，并同时实现了对硬件端口的操作函数 inport/outport 系列。

针对 PCI 设备动态分配 IO 等系统资源的情况，而用户有时需要直接用 inport/outport 等函数操作 PCI 板的内部硬件资源，瑞博华公司对瑞博华提供的采用 PCI 接口的 AD、DA、数字量输入输出板提供查询板基地址的驱动程序 LocatePCI.DLL。此驱动程序依赖于 IOTOOLS 驱动程序。此驱动程序为同时使用多块 PCI 卡提供分配、管理功能。

针对瑞博华公司的 AD 卡，瑞博华公司不但提供完整的驱动程序，而且提供完整、高性能的采集、分析、保存、触发记录、网络传输等功能的录波器软件。同时，还把录波器功能封装在 FRecordCore.DLL 中，由用户程序用很少的代码实现完整的数据采集、分析功

能，并任意添加自己的分析、显示、控制功能，并集成在同一界面、程序中。

本文件针对以上各类驱动程序，介绍在各种编程环境下的注意事项和编程示例。

二、编程的通用说明

2.1 编程方法

瑞博华公司提供的驱动程序或软件模块均以 DLL 方式提供，由 DLL 驱动再调用相应的各种内核驱动程序(VXD、386、SYS 等)。用户程序直接调用 DLL 提供的函数即可完成所有的功能。因此，所有能调用 DLL 的编程语言或环境均可调用瑞博华公司提供的驱动程序，而几乎所有的编程语言均提供直接调用 DLL 的功能，而且瑞博华公司还提供在各种环境中调用 DLL 的更方便的手段。

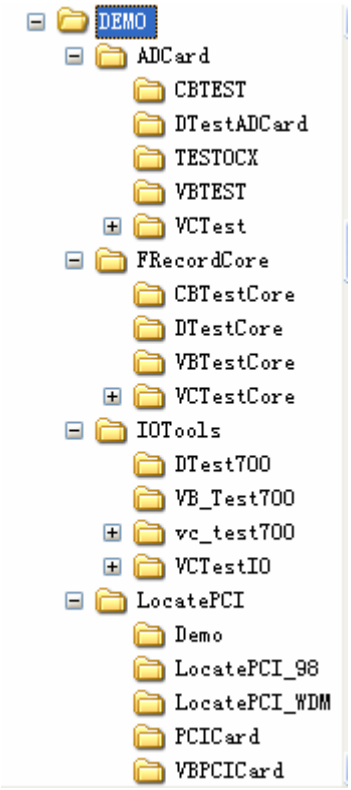
为在 VC、C++Builder 等环境中调用这些 DLL，最简单的方式是：在源程序中包含相关的头文件后，如 ADCard.H、LocatePCI.H 等，在程序中直接调用 DLL 中的函数，在连接时包含相关的库文件，如 ADCard.LIB、LocatePCI.LIB 等，就可以了。为包含这些库文件，可直接添加在 VC 或 C++Builder 的工程中即可(参见编程示例 VCTEST.ZIP)，或在命令行编译时添加在命令行中即可(参见编程示例 CON_TEST.ZIP)。另一种调用方式是，用 Windows 提供的 API 函数 LoadLibrary()打开 DLL 文件，用 GetProcAddress 得到相关函数的地址，再直接调用此地址即可（参见编程示例 CBTEST.ZIP）。编程例子可参见 Console 模式的例子 Con_test、VC 界面例子 VCTest、C++Builder 下例子 CBTest。

为在 VB 中调用这些 DLL，使用瑞博华公司提供的 VB 模块文件(如 DllDef.bas)文件后，就可直接在 VB 程序中直接调用这些 DLL 中的函数。具体例子可参见编程示例 VBTest.ZIP。VB 下使用 ADCard，还可直接使用 OCX 控件，编程示例参见 TestOCX.ZIP。

为在 Delphi 中调用这些 DLL，使用瑞博华公司提供的 Delphi 定义文件(如 AdCard.pas)，就可在 Delphi 程序中直接调用这些 DLL 中的函数。具体例子可参见编程示例 Delphi.ZIP。

2.2 编程示例目录列表

本文件所附带的编程示例文件见下面的目录树，覆盖了 ADCard、FRecordCore、IOTOOLS、LocatePCI 等在 VC++、C++Builder、VB、Delphi 等下的示例代码，和相关的头文件、库文件。另外还提供了在 Labview 下使用 ADCard 进行采集的示例程序。为正确的使用各示例程序，应先正确安装相应瑞博华公司采集板卡的驱动程序(如采集卡、开关量输入输出卡、IOTOOLS 工具驱动、LocatePCI 驱动等)。



三、 ADCard 的编程

3.1 功能说明

对瑞博华公司的所有 AD 卡、模块，瑞博华公司均提供统一接口的 ADCard 驱动程序，适用于 Windows 9x/Me/2000/XP，并可直接支持瑞博华公司的高性能网络录波器。ADCard 驱动程序提供了一套简捷、方便的操作 AD 板的函数，使用户以简单、高效地方式操作 AD 板，而不用处理底层的硬件细节。所有的 AD 板均以硬时钟中断、大容留 FIFO、DMA 等方式，实现高采集速度、严格定时、后台实时采集，但这些细节在 ADCard 驱动程序中用户程序均不用考虑就可高效实现。同时，针对不同 AD 板的特殊特性，如 DA 输出、信号发生器功能、各通道的程控放大等，提供了万能的设备数据发送接口 IOCtl()来统一实现这些功能。各驱动程序还实现了对硬件端口的访问函数 inport/outport 等，便于用户实现特殊功能、设备的底层控制。

ADCard 驱动程序实现了采集卡在后台的实时、高速采集模拟信号，应用程序只需在必要时取采集结果即可，可以实现数据的实时采集、连续显示与处理、采集与取数、显示的同步，而且可以很简单、方便地实现。通过 ADCard.DLL 提供的一序列函数(主要包括初始化函数 Initial()、多缓冲区方式开始函数 StartIntr()、单缓冲区方式开始函数 StartSnapshot()、采集停止函数 StopIntr()、取采集结果函数 ADResult()、ADResultRecent()、GetSnapshot()等)，

实现了两种方式的实时信号采集、缓冲数据管理方式：

一种是多缓冲区方式，或称为块 FIFO 方式。用 Initial()函数初始化驱动程序后，用 StartIntr()函数启动此工作模式，最后用 StopIntr()函数停止。此方式下，系统(驱动程序)内部开有 NumInnerBuf 个缓冲区(块)，形成先进先出的 FIFO 结构，存取采集结果的最小单位为 1 个缓冲区(块)。每个缓冲区可存放 NumChn 个通道、每通道 SampPerChn 个 AD 采样结果，这些缓冲区大小值、缓冲区的个数均可由应用程序自己指定，从而可能在性能、资源使用量方面得到最优的平衡。AD 采集的结果实时、顺序地存放在各缓冲区中，ADResult()函数就顺序地取出(最先)一个放满采样结果的缓冲区的内容(进行过此操作后此缓冲区又可接收新的采集结果)，和标准的 FIFO 工作模式完全一样。而 ADResultRecent()函数直接取最后(新)一个缓冲区的内容(进行过此操作后所有缓冲区又可重新接收新的采集结果)。在这种方式下，存放有 AD 采集结果的缓冲区数目可用 QueryBuf()函数进行查询。而且，通过使用 RegisterNotify()函数，可设置在每一个缓冲区满时会给用户程序发送消息，用户可通过此消息及时、高效地取得 AD 结果，且是不丢点、连续的 AD 采集结果，而不用等待、查询。当然，若用户程序没有及时从缓冲区取采样结果，而使所有缓冲区都满时，此后 AD 采样结果就将丢失，直到有缓冲区被 GetADResult()或 GetADResultRecent()释放、可存放新的采样结果为止。此工作方式适宜于连续记录 AD 采集结果。连续两次成功地 GetADResult()取得的 AD 结果在时间上不会重叠，之间是否丢失采样结果，要看返回结果的序列号(结果数组的第一个值，为无符号短整数)是否递增加一(注意，0xFFFF 加一后会变成 0)。只增一，说明两缓冲区在时间上是连续的。在采样率较低时，通过多缓冲区方式，有可能连续地记录、处理、保存长时间的 AD 结果。

另一工作方式为单缓冲区方式，驱动程序经 Initial()函数初始化后，用 StartSnapshot()方法启动，最后用 StopIntr()停止。在此模式下，系统(驱动程序)维护一个大的缓冲区，硬件始终不断地往缓冲区中存放 AD 结果，缓冲区大小为 NumChn 个通道，每通道 SampPerChn 个点，大小也可由用户自由指定。用 GetSnapshot()函数从此缓冲区中得到所有通道、最新指定个点(点数在方法的参数中指定)的采集结果(的快照)。此工作方式适宜于示波器方式，可随时得到最新的采集结果(的快照)。但两次 GetSanpshot()取得的结果在时间上可能有重叠或丢失，相对时间关系不能确定，或者通过函数 GetSnapshot()返回的 SampPtr 值和返回的 ADBuf 第一个元素所代表的系列号来求出。

各函数调用的顺序是：为了进行 AD 操作，程序先调用 Initial()函数，再调用 StartIntr()函数开始采集，反复调用 ADResult()函数或 ADResultRecent()函数取得采集结果后，或调用 QueryBuf()函数查询缓冲区块的状况，调用 StopIntr()函数停止采集。或通过先调用 Initial()初始化，调 StartSnapshot()开始采集，反复调用 GetSnapshot()取得采集结果后，调用 StopIntr()停止采集。可通过 ChannelFrq()函数计算实际每通道的采样频率。通过 ConfigInfo()函数查询驱动程序的设置。可通过 RegisterNotify()设置多缓冲区模式下每缓冲区满时下应用程序发

的消息。高级用户还可以通过 IOCtl()函数、根据采集板的使用说明书设置 AD 卡的特殊功能。

ADCard.DLL 中的命名规则为：如对于函数 Initial()，在 C 语言下原型定义为 int __stdcall Initial(WORD IOBase, WORD IRQNum, DWORD PhysAddr,WORD DMAChn)，有 4 个参数，则 DLL 中的真实名字为_Initial@16。而 VC 下的库 ADCard.LIB 和 C++Builder 下的库_ADCard.LIB 可自动连接正确的名称，在源程序中使用名称 Initial()即可。而在 VB 中，通过在定义中包括 Declare Function DllInitial Lib "adcard.dll" Alias _Initial@16 指令就可使用正确的 DLL 函数名字。在 Delphi 中，通过在函数引用语句中包含 stdcall; external'adcard.dll' name '_Initial@16'即可正确引用 DLL 中的正确名字。

3.2 编程示例列表

VC 下命令行(Console)模式的编程示例见 CON_TEST.ZIP，包含文件 Con_test.c，头文件 ADCard.H，库文件 ADCard.LIB。运行时需要安装瑞博华公司的任何一个 AD 采集卡或模块，即需要 ADCard.DLL 存在于系统目录。

Console 模式的编程示例 CON_TEST.ZIP 也可用于 C++Builder 任何版本，这时包含文件 Con_test.c，头文件 ADCard.H，库文件_ADCard.LIB(不同于 VC 用的库 ADCard.LIB)。运行时需要安装瑞博华公司的任何一个 AD 采集卡或模块，即需要 ADCard.DLL 存在于系统目录。

VC 带界面(MFC)模式下的编程示例见 VCTEST.ZIP。它使用了瑞博华公司提供的头文件 ADCard.H(通过在源程序中包括#include “ADCard.h”来引用)，库文件 ADCard.LIB(在工程中添加，连接时使用)。

C++Builder 下带界面的编程示例见 CBTest.ZIP。它演示了在 C/C++语言中通过 Windows 的 LoadLibrary()、GetProcAddress()等 API 直接调用 ADCard.DLL 的方法。配合 GetProcAddress()用地类型定义可参见头文件 ADCard1.H。同样的方法也可用于 VC 或任何其它编程语言。它不使用任何库文件，更更具灵活性。

VB 下的编程示例见 VBTest.ZIP。通过瑞博华公司提供的 VB 模块文件 ADTEST1.BAS 或 DLLDef.BAS，在 VB 程序下可直接调用 ADCard.DLL 通过的函数。

VB 下使用 ADCardX.OCX 框架的编程示例可见 TESTOCX.ZIP。把 ADCardX.OCX 拷贝到 Windows\system 目录后，并注册到 VB 环境后，可方便地通过 VB 控件的属性(property)、方法(Method)、事件(Event)等方式来控制 ADCard.DLL 的运行。ADCardX 控件的使用方法可参见专门的使用文档。

Delphi 下编程示例见 Delphi.ZIP，它实现了与 VC 编程示例完全一样的功能。它通过瑞博华通过的单元定义文件 ADCard.pas，可实现在 Delphi 中直接调用 ADCard.DLL 的函数。

3.3 ADCard 命令详解

3.3.1 定义的常数和结构

C/C++下的定义，已包含于 ADCard.H。VB 下的类似定义在 DllDef.BAS 中。

```
//AD Result Struct
typedef struct structADResult {
   WORD SeqNo;
   WORD ADValue[1];
} ADCard_Result;

//API return value
#define ADCard_Success    1
#define ADCard_Error    0
```

3.3.2 驱动程序初始化 Initial

VC/C++Builder 下定义原型：

```
int __stdcall Initial(WORD IOBase, WORD IRQNum, DWORD PhysAddr,WORD DMAChn);
```

VC/C++Builder 下使用示例：

```
  I = Initial(0x140,5,0xd800,0);
```

VB 下定义方式：

```
Declare Function DllInitial Lib "adcard.dll" Alias "_Initial@16" (ByVal IOBase As Integer, ByVal IRQNum As Integer, ByVal PhysAddr As Long, ByVal DMAChn As Integer) As Long
```

VB 下使用示例：

```
  I = DllInitial(&H140,5,&HD800,0)
```

Delphi 下定义方式

```
function Initial(IOBase:word; IRQNum:word; Physaddr:longword; DMAChn:word):longword; stdcall; external'adcard.dll' name '_Initial@16';
```

Delphi 下使用示例

```
  I := Initial($140,5,$d800,0);
```

功能及参数说明：

在调用 ADCard 的任何函数之前，应先调用此函数，进行硬件和采集卡的初始化。

参数 IOBase 指定采集卡的基地址、参数 IRQNum 指定卡所使用的中断号、参数 PhysAddr 指定卡所使用的外部物理内存的开始地址，DMAChn 指定卡使用的 DMA 通道号。对于采用 ISA 类型的卡，这些参数必须与硬件说明书给定的值一致；对自动分配资源的 PCI 接口卡，可使用任意值，驱动程序内部会自动使用合适的值。

如果初始化成功(驱动程序正确、AD 板存在、参数正确、采集板还未处于采集状态)，函数将返回 1(=ADCard_Sucess)，否则将返回 0(=ADCard_Error)。

3.3.3 开始多缓冲区模式的采集 StartIntr

VC/C++Builder 下定义原型：

```
DWORD __stdcall StartIntr(WORD NumBuf, DWORD NumSamp,\
                          WORD BegChn, WORD NumChn, \
                          DWORD FrqSamp, DWORD FrqFilter,\
                          WORD AmpGain);
```

VC/C++Builder 下使用示例：

```
I = StartIntr(NumBuf,NumSamp,BegChn,NumChn, FrqSamp, FrqFilter,AmpGain);
```

VB 下定义方式：

```
Declare Function DllStartIntr Lib "ADCard.Dll" Alias "_StartIntr@28" (ByVal NumBuf As Integer, ByVal NumSamp As Long, ByVal BegChn As Integer, ByVal NumChn As Integer, ByVal FrqSamp As Long, ByVal FrqFilter As Long, ByVal AmpGain As Integer) As Long
```

VB 下使用示例：

```
I = DllStartIntr(NumBuf, NumSamp, BegChn, NumChn, FrqSamp, FrqSamp, AmpGain)
```

Delphi 下定义方式

```
function StartIntr(NumBuf:word; NumSamp:Longword;
                          BegChn:word; NumChn:word;
                          FrqSamp:Longword; FrqFilter:Longword;
                          AmpGain:word):Longword; stdcall; external 'adcard.dll' name '_StartIntr@28';
```

Delphi 下使用示例

I := StartIntr(NumBuf, NumSamp, BegChn, NumChn, FrqSamp, FrqFilter,AmpGain);

功能及参数说明：

本函数启动多缓冲区工作方式。成功时返回每缓冲区的大小(字节数)，之后，用ADResult()或 ADResultRecent()函数取采集结果时用到的缓冲区大小应大于或等于此值(字节数)。失败时返回 0。

此函数应在 Initial 成功调用之后使用。此函数成功后可以使用 ADResult()、ADResultRecent()、QueryBuf()等函数，并用 StopIntr()函数结束多缓冲区采集方式。

参数 NumBuf 指定驱动程序内部使用的缓冲区的块数（相当于块 FIFO 的容留），一般 5~10 即可，若用户程序取数可能不及时，可加大此值，以防丢失采集数据。

参数 NumSamp 指定缓冲区块的大小，即每通道的采集点数。此值也规定以后 ADResult()或 ADResultRecent()使用的数据缓冲区的大小，即每块 NumChn 个通道、每通道 NumSamp 个采样点，每采集点为一无符号短整数。为保证驱动程序的性能(点采集、块处理方式)，不同的采集板的驱动程序会规定 NumSamp 的最小值。

参数 BegChn 表示采集的开始通道。对不同的采集板此值的含义可能不同。在有些采集板中，BegChn 无任何意义。在有些采集板中，BegChn 表示仅采集 1 通道时此通道所在的位置。在有些采集板中，BegChn 表示在 NumChn 个通道中，前 BegChn 个通道是真正的模拟量，以后的通道是开关量通道的组合。具体定义参见采集卡的硬件说明书。

参数 NumChn 表示采集的总通道数。此数的范围应工具相应采集卡的硬件说明书进行设置。

整型参数 FrqSamp 表示采集的名义频率。由于可能是多通道采集，有些采集卡还通过多路开关由多通道共享 1 个 AD 芯片，或通过一分频器由固定晶振分频得到，因此，每通道的数据采样频率可能不同于名义采样频率。每通道的实际采样频率可通过函数 ChannelFrq()以浮点数方式返回精确值。

参数 FrqFilter 指定电路的低通滤波器的截止频率。在有些采集卡上此值无意义。

参数 AmpGain 指定板上程控放大器的放大倍数。具体含义请参照使用的采集卡的硬件说明书。

3.3.4 停止采集函数 StopIntr

VC/C++Builder 下定义原型：

int __stdcall StopIntr(void);

VC/C++Builder 下使用示例：

StopIntr();

VB 下定义方式：

```
Declare Function DllStopIntr Lib "ADCard.Dll" Alias "_StopIntr@0" () As Long
```

VB 下使用示例：

```
I = DllStopIntr()
```

Delphi 下定义方式

```
function StopIntr :Longword; stdcall; external 'adcard.dll' name '_StopIntr@0';
```

Delphi 下使用示例

```
I := StopIntr;
```

功能及参数说明：

此函数停止单缓冲区或多缓冲区方式的采集。在成功调用 StartIntr()或 StartSnapshot()之后才需要调用此函数。在应用程序退出前，应保证调用此函数，否则下次采集可能不正确。函数成功调用时将返回 1(=ADCard_Sucess)，否则将返回 0(=ADCard_Error)。

3.3.5 开始单缓冲区采集方式函数 StartSnapshot

VC/C++Builder 下定义原型：

```
DWORD __stdcall StartSnapshot(WORD NumBuf, DWORD NumSamp,\
                              WORD BegChn, WORD NumChn,\
                              DWORD FrqSamp, DWORD FrqFilter,\
                              WORD AmpGain,DWORD NotifyLength);
```

VC/C++Builder 下使用示例：

```
I = StartSnapshot(NumBuf, NumSamp, BegChn, NumChn, FrqSamp, FrqFilter, AmpGain, NotifyLength);
```

VB 下定义方式：

```
Declare Function DllStartSnapshot Lib "ADCard.Dll" Alias "_StartSnapshot@32" (ByVal NumBuf As Integer, ByVal NumSamp As Long, ByVal BegChn As Integer, ByVal NumChn As Integer, ByVal FrqSamp As Long, ByVal FrqFilter As Long, ByVal AmpGain As Integer, ByVal NotifyLength As Long) As Long
```

VB 下使用示例：

I = DllStartSnapshot(NumBuf, NumSamp, BegChn, NumChn, FrqSamp, FrqFilter, AmpGain, NotifyLength)

Delphi 下定义方式

```
function StartSnapshot(NumBuf:word; NumSamp:Longword;
                       BegChn:word; NumChn:word;
                       FrqSamp:Longword; FrqFilter:Longword;
                       AmpGain:word;NotifyLength:Longword):Longword;stdcall;
external 'adcard.dll' name '_StartSnapshot@32';
```

Delphi 下使用示例

I := StartSnapshot(NumBuf, NumSamp, BegChn, NumChn, FrqSamp, FrqFilter, AmpGain, NotifyLength);

功能及参数说明：

本函数启动单缓冲区工作方式。成功时返回内部缓冲区的大小(字节数)。失败时返回 0。

参数 NumBuf, NumSamp, BegChn, NumChn, FrqSamp, FrqFilter, AmpGain 的含义与函数 StartIntr()的相同，但 NumBuf 在驱动程序内部会强制为 1。参数 NotifyLength 指定每通道每采集 NotifyLength 个采样点，就向应用程序发送通过 RegisterNotify 注册的消息。

3.3.6 查询多缓冲区方式下采集缓冲区数 QueryBuf

VC/C++Builder 下定义原型：

```
WORD __stdcall QueryBuf(void);
```

VC/C++Builder 下使用示例：

```
NumBuf = QueryBuf();
```

VB 下定义方式：

```
Declare Function DllQueryBuf Lib "ADCard.Dll" Alias "_QueryBuf@0" () As Integer
```

VB 下使用示例：

```
NumBuf = DllQueryBuf;
```

Delphi 下定义方式

```
function QueryBuf :word; stdcall; external 'adcard.dll' name '_QueryBuf@0';
```

Delphi 下使用示例

NumBuf := QueryBuf;

功能及参数说明：

返回多缓冲区方式下已有采集结果的缓冲区数目。失败(如 AD 板未通过 StartIntr()开始采集)返回 0。通过此函数，应用程序可在单独的线程中用查询方式实现多缓冲区方式下采集。

3.3.7 取多缓冲区方式下一个采集块 ADResult

VC/C++Builder 下定义原型：

```
int __stdcall ADResult(ADCard_Result * Buf);
```

VC/C++Builder 下使用示例：

```
WORD ADBuf[NumChn*NumSamp+1];
I = ADResult((ADResult *) ADBuf);
```

VB 下定义方式：

```
Declare Function DllADResult Lib "ADCard.Dll" Alias "_ADResult@4" (ByRef pBuf As Integer) As Long
```

VB 下使用示例：

```
Dim ADBuf[NumSamp*NumChn] as Integer
I = DllADResult(ADBuf[0])
```

Delphi 下定义方式

```
function ADResult(var Buf:word):longint; stdcall; external 'adcard.dll' name '_ADResult@4';
```

Delphi 下使用示例

```
ADBuf : array of word;
    SetLength(ADBuf,NumSamp*NumChn+3);
    I := ADResult(ADBuf);
```

功能及参数说明：

ADResult()函数用于在多缓冲区方式下把最先(旧)的一个有采集结果的缓冲区内容拷贝

到用户数组 ADBuf 中。并释放当前缓冲区。这是把驱动程序缓冲区当成以块为单位的 FIFO 时，以块为单位取采集结果的标准方式。若执行 ADResult()及时(如在 RegisterNotify()注册的消息响应函数中)，各 ADResult()函数返回的缓冲区将在时间上是连续的。

参数 ADBuf 为用户程序通过的存放采集结果的缓冲区地址。此时数组 ADBuf 的尺寸要求大于 SampPerChn*NumChn+1 个短整数，但结果正好为 NumChn 通道、NumSamp 个采样点，NumChn 和 NumSamp 在 StartIntr()函数中定义。

返回结果的格式是：ADBuf(0)为此次结果的序列号：在多缓冲区方式下，每个缓冲区的序列号按采集时间先后顺序增一，若缓冲区满而丢失数据时此值会增加以表示丢失了多少缓冲区；在单缓冲区方式下，表示此次快照是系统内部缓冲区被覆盖(填满)多少此时的结果。此值是按无符号短整数递增的，在 65535(=0xffff)后会反转为 0 的。

ADBuf(1 至 NumChn*SampPerChn)顺序存放各通道、各时刻的采样值。第 n 通道(n=0 至 NumChn-1)、第 m 时刻(m=1 至 SampPerChn)就存放在数组单元 ADBuf((m-1)*NumChn+n+1)中，即 ADBuf(1)存放第 0 通道第 1 时刻的值，ADBuf(2)存放第 1 通道第 1 时刻值，.....，ADBuf(NumChn)存放第 NumChn-1 通道第 1 时刻的值，ADBuf(NumChn+1)存放第 0 通道第 2 时刻的值，.....。对 12 位 AD 板，采样结果为偏移二进制编码，即：0 表示最低电压值(如-5V)，2048(=0x800)表示 0 电压，4095(=0xfff)表示最高电压(如+5V)。具体电压值可由属性 VZero 和 VMax 转换出。

若 AD 板既支持模拟量，又支持开关量，则在等价的 NumChn 个总通道中，通道 0 至 BegChn-1 为常规的模拟量，通道 BegChn 至 NumChn-1 为组合的开关量通道，每 16 个开关量通道占 1 个模拟量通道的缓冲区(即每模拟通道占 1 位，且通道 0 占短整数的 bit 0,通道 15 占短整数的 bit15，而每模拟通道占 16 位(无符号短整数))

对 AD 采集结果 ADBuf(?)的模拟量，它实际采用偏移二进制编码，它对应的实际电压为(单位：V)：

(ADBuf(?)-VZero)/(VMax-VZero)*5

其中，VMax 与 VZero 由 ConfigInfo()函数返回。由于 VB 不支持无符号整数(只能按有符号整数对待)，对 16 位 AD 芯片(VZero=32768., VMax=65535.)，要得到电压值，需用如下公式：

```
If (ADBuf(1) > 0) Then
V = (ADBuf(1) - VZero) / (VMax - VZero) * 5
Else
V = (65536 + ADBuf(1) - VZero) / (VMax - VZero) * 5)
End If
```

3.3.8 取多缓冲区方式下最新采集块 ADResultRecent

VC/C++Builder 下定义原型：

```
int __stdcall ADResultRecent (ADCard_Result * Buf);
```

VC/C++Builder 下使用示例：

```
WORD ADBuf[NumChn*NumSamp+1];
I = ADResultRecent ((ADResult *) ADBuf);
```

VB 下定义方式：

```
Declare Function DllADResultRecent Lib "ADCard.Dll" Alias "_ADResultRecent@4" (ByRef pBuf As Integer) As Long
```

VB 下使用示例：

```
Dim ADBuf[NumSamp*NumChn] as Integer
I = DllADResultRecent (ADBuf[0])
```

Delphi 下定义方式

```
function ADResultRecent (var Buf:word):longint; stdcall; external 'adcard.dll' name '_ADResult@4';
```

Delphi 下使用示例

```
ADBuf : array of word;
    SetLength(ADBuf,NumSamp*NumChn+3);
    I := ADResultRecent (ADBuf);
```

功能及参数说明：

函数 ADResultRecent()在多缓冲区方式下把最新的一个有采集结果的缓冲区内容拷贝到用户数组 ADBuf 中，并释放所有内部缓冲区。这仅用于用户想取得最新采集结果，而不关心各块采集结果之间的时间连续性的特殊情况。

参数 ADBuf 为用户程序通过的存放采集结果的缓冲区地址。缓冲区的大小与格式同于 ADResult()。此时数组 ADBuf 的尺寸要求大于 SampPerChn*NumChn+1 个短整数，但结果正好为 NumChn 通道、NumSamp 个采样点，NumChn 和 NumSamp 在 StartIntr()函数中定义。

3.3.9 取单缓冲区方式下最新采集结果 GetSnapshot

VC/C++Builder 下定义原型：

```
int __stdcall GetSnapshot(ADCard_Result * Buf, int SampPerChn, DWORD * SampPtr);
```

VC/C++Builder 下使用示例：

```
WORD ADBuf[NumChn*NumSamp+1];
I = ADResultRecent ((ADResult *) ADBuf);
```

VB 下定义方式：

```
Declare Function DllGetSnapshot Lib "ADCard.Dll" Alias "_GetSnapshot@12" (ByRef pBuf As Integer, ByVal SampPerChn As Long, ByRef SampPtr As Long) As Long
```

VB 下使用示例：

```
Dim ADBuf[NumSamp*NumChn] as Integer
I = DllADResultRecent (ADBuf[0])
```

Delphi 下定义方式

```
function GetSnapshot(var Buf:word; SampPerChn:longint;
                var SampPtr:longword):longint; stdcall; external 'adcard.dll' name '_GetSnapshot@12';
```

Delphi 下使用示例

```
ADBuf : array of word;
    SetLength(ADBuf,NumSamp*NumChn+3);
    I := ADResultRecent (ADBuf);
```

功能及参数说明：

此函数在单缓冲区方式下把最新的采集结果(仅 SampPerChn 个点、NumChn 个通道)拷贝到用户数组 ADBuf 中。最新数据在内部缓冲区中的位置返回在 SampPtr 中，它的范围为 0~NumSamp-1，而内部缓冲区可能已被覆盖过几次，覆盖的次数返回在数组 ADBuf 的第一个短整数中，即 SeqNo 中，用户可由这两个参数确定两次 GetSnapshot 取的结果的相对时间关系。

参数 ADBuf 为用户程序通过的存放采集结果的缓冲区地址。这时用户数组尺寸要求大于或等于 SampPerChn*NumChn+3 个短整数。缓冲区的大小与格式同于 ADResult()。但结果正好为 NumChn 通道、NumSamp0 个采样点，NumChn 和 NumSamp 在 StartSnapshot()函

数中定义。要求 SampPerChn<NumSamp。

3.3.10 多缓冲区方式注册缓冲区满的消息 RegisterNotify()

VC/C++Builder 下定义原型：

```
int __stdcall RegisterNotify(UINT NotifyNo, HANDLE hWnd,UINT uMsg,\
                             UINT wParam, LONG lParam);
```

VC/C++Builder 下使用示例：

```
RegisterNotify(0,hWnd, WM_KEYDOWN,48,0);
```

VB 下定义方式：

```
Declare Function DllRegisterNotify Lib "ADCard.Dll" Alias "_RegisterNotify@20" (ByVal NotifyNo As Long, ByVal hWnd As Long, ByVal uMsg As Long, ByVal wParam As Long, ByVal lParam As Long) As Long
```

VB 下使用示例：

```
i = DllRegisterNotify(0, C_Notify.hWnd, WM_KEYDOWN, 48, 0)
```

Delphi 下定义方式

```
function RegisterNotify(NotifyNo:Longword; hWnd:HWND;
                        uMsg:Longword;
                        wParam:longword; lParam:longint):Longword;
                        stdcall; external 'adcard.dll' name '_RegisterNotify@20';
```

Delphi 下使用示例

```
i := RegisterNotify(0,Button3.Handle,WM_KEYUP,48,0);
```

功能及参数说明：

此函数设置或取消提示(Notify)消息，即 AD 板驱动程序在多缓冲区方式下每缓冲区块装满采集结果时，将向用户程序(实际为句柄 hWnd 代表的子窗口(可以是按钮等控件))发送消息(uMsg,wParam,lParam)；可注册 0 至 3 号多个消息(NotifyNo=0~3)，即每个缓冲区块满时同时发送这些所有的消息，但通常使用 1 个消息就足够；若 hWnd=0 或 uMsg=0 将屏蔽此消息。在上面的示例中，说明要求 AD 驱动程序在每个缓冲区满时，向用户程序的按钮子窗口发送一个按下’A’键的消息。此后，用户在此子窗口的相应消息响应函数中，调用 ADResult()取一块采集结果即可。

此函数通常在 StartIntr 前调用，使通知消息有效。

特别注意：在这种多缓冲区、消息驱动取采集结果的工作模式下，如果采集速度比较高，且用户程序在此消息响应函数中处理采集结果时，或在其它消息响应函数中进行其它操作时，花了太多的时间，就可能来不及取采集结果，从而使驱动程序中所有缓冲区都满，并且向用户程序发送的消息淤积，丢失信息，从而使消息链断链，表现为应用程序从此以后再也得不到缓冲区满的消息。针对情况，用户程序应在代表缓冲区满的消息响应函数中，

取一块采集结果后，用 QueryBuf()判断，是否还有存放满采集结果的缓冲区块存在，若存在，可在此消息响应函数中直接再多取 1 或多块采集结果缓冲区，从而防止消息淤积或消息断链的情况。

3.3.11 返回实际采样频率 ChannelFrq

VC/C++Builder 下定义原型：

```
float __stdcall ChannelFrq(int NumChn,long FrqSamp);
```

VC/C++Builder 下使用示例：

```
Frq = ChannelFrq(NumChn, FrqSamp);
```

VB 下定义方式：

```
Declare Function DllChannelFrq Lib "ADCard.Dll" Alias "_ChannelFrq@8" (ByVal NumChn As Long, ByVal FrqSamp As Long) As Single
```

VB 下使用示例：

```
Frq = DllChannelFrq(NumChn, FrqSamp)
```

Delphi 下定义方式

```
function ChannelFrq(NumChn:longint; FrqSamp: longint):single;stdcall; external 'adcard.dll' name '_ChannelFrq@8';
```

Delphi 下使用示例

```
Frq := ChannelFrq(NumChn, FrqSamp);
```

功能及参数说明：

函数 ChannelFrq()返回值实际每通道的准确采样频率，单位 Hz。NumChn 和 FrqSamp 为 StartIntr()或 StartSnapshot()中使用的值。

3.3.12 设备专用数据发送函数 IOCtl

VC/C++Builder 下定义原型：

```
int __stdcall IOCtl(int InSize,char * InBuff,int OutSize,char * OutBuff);
```

VC/C++Builder 下使用示例：

```
int InSize, OutSize;
char InBuff[InSize], OutBuff[OutSize];
I = IOCtl(InSize, InBuff, OutSize, OutBuff);
```

VB 下定义方式：

```
Declare Function DllIOCtl Lib "ADCard.Dll" Alias "_IOCtl@16" (ByVal InSize As Long, ByVal InBuff As String, ByVal OutSize As Long, ByVal OutBuff As String) As Long
Declare Function DllIOCtlByte Lib "ADCard.Dll" Alias "_IOCtl@16" (ByVal InSize As Long, ByRef InBuff As Byte, ByVal OutSize As Long, ByRef OutBuff As Byte) As Long
```

VB 下使用示例：

```
DIM InBuff[InSize] as byte, OutBuff[OutSize] as byte
Dim InBuffS as string, OutBuffS as string
I = DllIOCtl(InSize, InBuffS, OutSize, OutBuffS)
I = DllIOCtlByte(InSize, InBuff[0],OutSize, OutBuff[0])
```

Delphi 下定义方式

```
function IOCtl(InSize:longint; InBuff:pchar;
                    OutSize:longint; OutBuff:pchar):longint;stdcall; external 'adcard.dll' name '_IOCtl@16';
```

Delphi 下使用示例

```
I := IOCtl(InSize, InBuff[0], OutSize, OutBuff[0]);
```

功能及参数说明：

函数 IOCtl()通过对 AD 板专用功能设置的通用接口。比如实现信号发生器的功能、设置各通道的单独程控放大倍数等。调用成功时返回 1(=ADCard_Success)，调用失败时返回 0(=ADCard_Error)。具体 InSize，OutSize,InBuff[],OutBuff[]的设置格式、要求、意义参见各采集卡的硬件说明书。

3.3.13 获取驱动程序配置信息 ConfigInfo

VC/C++Builder 下定义原型：

```
void __stdcall ConfigInfo(char * ADCard_Name,\
                          int * MaxChn,\
                          int * LowFreq,\
                          int * HighFreq,\
```

int * MinSampNum,\
float * VZero,\
float * VMax,\
int * MaxBinChn);

VC/C++Builder 下使用示例：

char CardName[32];

ConfigInfo(CardName,&MaxChn, &LowFreq, & HighFreq, & MinSampNum, &Vzero, &Vmax, & MaxBinChn);

VB 下定义方式：

Declare Sub DllConfigInfo Lib "ADCard.Dll" Alias "_ConfigInfo@32" (ByVal ADCardName As String, ByRef MaxChn As Long, ByRef LowFrq As Long, ByRef HighFrq As Long, ByRef MinSampNum As Long, ByRef VZero As Single, ByRef VMax As Single, ByRef MaxBinChn As Long)

VB 下使用示例：

Dim ADCardName as string

ADCardName = space(32)

DllConfigInfo(ADCardName,MaxChn,LowFrq,HighFrq,MinSampNum, Vzero, Vmax, MaxBinChn)

Delphi 下定义方式

procedure ConfigInfo(ADCard_Name:pchar;
var MaxChn : longint;
var LowFreq : longint;
var HighFreq : longint;
var MinSampNum : longint;
var VZero : Single;
var VMax : Single;
var MaxBinChn:longint); stdcall; external 'adcard.dll' name '_ConfigInfo@32';

Delphi 下使用示例

var CardName: array[0..31] of char;

var pName : pChar;

pName := @Cardname;

ConfigInfo(pName, MaxChn,LowFrq,HighFrq,MinSampNum, Vzero, Vmax, MaxBinChn);

功能及参数说明：

函数 ConfigInfo()返回安装的 AD 采集卡及驱动程序的各种配置信息。参数 CardName 返回采集卡的名称；MaxChn 返回采集卡支持的最大通道号；LowFrq 返回最低(名义)采集频率；HighFrq 返回最高(名义)采集频率；MinSampNum 返回 StartIntr()或 StartSnapshot()函数中 NumSamp 的最小值； Vzero 返回电平 0V 代表的 AD 值(12 位 AD 值时应为 2048，16 位 AD 时应为 32768), Vmax 代表+5V 代表的 AD 值(12 位 AD 值时应为 4095，16 位 AD 时应为 65535)； MaxBinChn 返回支持的最大开关量通道数。

3.3.14 按字节读硬件端口 inportb

VC/C++Builder 下定义原型：

```
WORD __stdcall inportb(WORD port);
```

VC/C++Builder 下使用示例：

```
B = inportb(port);
```

VB 下定义方式：

```
Declare Function DllInportB Lib "ADCard.Dll" Alias "_inportb@4" (ByVal Port As Integer) As Integer
```

VB 下使用示例：

```
B = DllInportB(port);
```

Delphi 下定义方式

```
function inportb(port:word):word; stdcall; external 'adcard.dll' name '_inportb@4';
```

Delphi 下使用示例

```
B := inportb(port);
```

功能及参数说明：

函数 inportb()按字节从硬件端口读。此函数能避开 Windows 对硬件操作的限制。同样的函数在 IOTOOLS、RBHIO 也提供。

3.3.15 按字读硬件端口 inport

VC/C++Builder 下定义原型：

```
WORD __stdcall inport(WORD port);
```

VC/C++Builder 下使用示例：

```
W = inport(port);
```

VB 下定义方式：

```
Declare Function DllInport Lib "ADCard.Dll" Alias "_inport@4" (ByVal Port As Integer) As Integer
```

VB 下使用示例：

```
W = DllInport(port);
```

Delphi 下定义方式

```
function inport(port:word):word; stdcall; external 'adcard.dll' name '_inport@4';
```

Delphi 下使用示例

```
W := inport(port);
```

功能及参数说明：

函数 inport()按字从硬件端口读。此函数能避开 Windows 对硬件操作的限制。同样的函数在 IOTOOLS、RBHIO 也提供。

3.3.16 按双字读硬件端口 inportdw

VC/C++Builder 下定义原型：

```
DWORD __stdcall inportdw(WORD port);
```

VC/C++Builder 下使用示例：

```
DW = inportdw(port);
```

VB 下定义方式：

```
Declare Function DllInportDW Lib "ADCard.Dll" Alias "_inportdw@4" (ByVal Port As Integer) As Long
```

VB 下使用示例：

DW = DllInportDW(port);

Delphi 下定义方式

function inportdw(port:word):longword; stdcall; external 'adcard.dll' name '_inportb@4';

Delphi 下使用示例

DW := inportdw(port);

功能及参数说明：

函数 inportdw()按双字从硬件端口读。此函数能避开 Windows 对硬件操作的限制。同样的函数在 IOTOOLS、RBHIO 也提供。

3.3.17 按字节写硬件端口 outportb

VC/C++Builder 下定义原型：

viod __stdcall outportb(WORD port,WORD value);

VC/C++Builder 下使用示例：

outportb(port, value);

VB 下定义方式：

Declare Sub DllOutportB Lib "ADCard.Dll" Alias "_outportb@8" (ByVal Port As Integer, ByVal Value As Integer)

VB 下使用示例：

DllOutportB(port, value);

Delphi 下定义方式

procedure outportb(port:word; value:word);stdcall; external 'adcard.dll' name '_outportb@8';

Delphi 下使用示例

outportb(port, value);

功能及参数说明：

函数 outportb()按字节往硬件端口读。此函数能避开 Windows 对硬件操作的限制。同样的函数在 IOTOOLS、RBHIO 也提供。

3.3.18 按字写硬件端口 outport

VC/C++Builder 下定义原型：

viod __stdcall outport(WORD port,WORD value);

VC/C++Builder 下使用示例：

outport(port, value);

VB 下定义方式：

Declare Sub DllOutport Lib "ADCard.Dll" Alias "_outport@8" (ByVal Port As Integer, ByVal Value As Integer)

VB 下使用示例：

DllOutport(port, value);

Delphi 下定义方式

procedure outport(port:word; value:word);stdcall; external 'adcard.dll' name '_outport@8';

Delphi 下使用示例

outport(port, value);

功能及参数说明：

函数 outportb()按字往硬件端口读。此函数能避开 Windows 对硬件操作的限制。同样的函数在 IOTOOLS、RBHIO 也提供。

3.3.19 按双字写硬件端口 outportdw

VC/C++Builder 下定义原型：

viod __stdcall outportdw(WORD port,DWORD value);

VC/C++Builder 下使用示例：

outportdw(port, value);

VB 下定义方式：

Declare Sub DllOutportDW Lib "ADCard.Dll" Alias "_outportdw@8" (ByVal Port As Integer, ByVal Value As Long)

VB 下使用示例：

DllOutportDW(port, dvalue);

Delphi 下定义方式

procedure outportdw(port:word; value:longword);stdcall; external 'adcard.dll' name '_outportdw@8';

Delphi 下使用示例

outportdw(port, value);

功能及参数说明：

函数 outportdw()按双字往硬件端口读。此函数能避开 Windows 对硬件操作的限制。同样的函数在 IOTOOLS、RBHIO 也提供。

四、IOTOOLS 工具的使用

4.1 IOTOOLS 工具概况

为了避开 Windows 对底层硬件操作的屏蔽，特别是为了从底层操作瑞博华公司提供各种板卡，瑞博华公司提供了 WDM 方式下(从而可同时支持 Windows 98/Me/2000/XP 环境，也有相应的 VXD 版本，供 Windows 95/98/Me 使用)操作硬件端口的驱动程序 IOTOOLS。应用程序通过直接调用 IOTOOLS.DLL(正确安装后位于 Windows\system 或 Winnt\system32 目录)来实现对硬件端口的访问。

IOTOOLS 驱动程序主要提供了按字节、字、双字读和写硬件端口的 inport/outport 函数。同样的函数在 ADCard 驱动程序和 RBHIO.DLL 中也提供，用户可以自由选择。

4.2 IOTOOLS 编程所需文件

用户程序为使用 IOTOOLS 驱动程序，直接调用 IOTOOLS.DLL 中的函数即可。为在 C、C++语言中调用 IOTOOLS，头文件为 IOTOOLS.H，也可直接借用 ADCard.H。VC 的库文

件为 IOTOOLS.LIB，C++Builder 的库文件为_IOTOOLS.LIB。VB 下使用模块文件 IOTOOLS.BAS 即可。Delphi 下使用单元文件 IOTOOLS.PAS 即可。

4.2.1 C/C++下的头文件 iotools.h 定义为

```
#ifdef __cplusplus
extern "C" {
#endif

//IO tools function
void __stdcall outportb(WORD port, WORD value);
WORD __stdcall inportb(WORD port);
void __stdcall outport(WORD port, WORD value);
WORD __stdcall    inport(WORD port);
void __stdcall outportdw(WORD port, DWORD value);
DWORD __stdcall inportdw(WORD port);

#ifdef __cplusplus
}
#endif
```

4.2.2 VB 下模块文件 iotools.bas 的内容为

Declare Sub DllOutportB Lib "IOTOOLS.DLL" Alias "_outportb@8" (ByVal Port As Integer, ByVal Value As Integer)

Declare Sub DllOutport Lib " IOTOOLS.DLL" Alias "_outport@8" (ByVal Port As Integer, ByVal Value As Integer)

Declare Sub DllOutportDW Lib " IOTOOLS.DLL" Alias "_outportdw@8" (ByVal Port As Integer, ByVal Value As Long)

Declare Function DllInportB Lib " IOTOOLS.DLL" Alias "_inportb@4" (ByVal Port As Integer) As Integer

Declare Function DllInport Lib " IOTOOLS.DLL" Alias "_inport@4" (ByVal Port As Integer) As Integer

Declare Function DllInportDW Lib " IOTOOLS.DLL" Alias "_inportdw@4" (ByVal Port As Integer) As Long

4.2.3 Delphi 下的单元定义文件 IOTOOLS.PAS 的内容为

```
unit IOTools;

interface

uses
   Windows;

     {Function declaration or IOTools.DLL}
     //IO tools function
     procedure outportb(port:word; value:word);stdcall; external 'iotools.dll' name '_outportb@8';
     function inportb(port:word):word; stdcall; external 'iotools.dll' name '_inportb@4';
     procedure outport(port:word; value:word);stdcall; external 'iotools.dll' name '_outport@8';
     function inport(port:word):word; stdcall; external 'iotools.dll' name '_inport@4';
     procedure outportdw(port:word; value:longword);stdcall; external 'iotools.dll' name '_outportdw@8';
     function inportdw(port:word):longword; stdcall; external 'iotools.dll' name '_inportdw@4';

implementation
end.
```

4.2.4 IOTOOLS 编程示例

瑞博华公司提供使用 IOTOOLS 的 C 语言 Console 模式(IO_Test)、VC(VCTestIO)、VB 等的使用示例程序。

演示 IO700 板功能的 VC 示例程序 VC_Test700 综合使用了 IOTools 和 LocatePCI 工具。同样功能的 VB 版演示程序见 VB_Test700，Delphi 版的演示程序见 Dtest700。

4.3 IOTOOLS 函数说明

各函数的具体说明可参见 3.3.13 至 3.3.18。

五、LOCATEPCI 工具说明

5.1 LOCATEPCI 功能说明

由于 PCI 接口的板卡具有自动分配 IO、MEM、IRQ 等资源的特性，在不同的配置下、插在不同的 PCI 槽上，系统都会动态分配一套可用的资源，这提供相当的灵活性和适应性。但若用户需要操作具体的板的底层硬件资源时，就必须动态地确定这些硬件资源。同时，若用户同时使用多块采集卡、开关量卡，如何区分、确定这些卡，这些就是 LocatePCI 工具要解决的问题。

具体地说，LocatePCI 可以对瑞博华公司提供的各种采集、开关量控制卡进行电子编号、按编号查询各板使用的硬件资源，从而，即使多个卡混合使用，插在任意 PCI 槽上，用户程序也可正确区分、定位、使用每块卡。

LocatePCI 以 DLL 方式提供，在内部它依赖于 IOTOOLS 工具。因此，使用 LocatePCI.DLL 前应正确安装 IOTOOLS 工具。

LocatePCI 和 IOTools 一样，分为 Windows 9x/Me 专用的版本和 Windows 98/2000/XP 通用的 WDM 版本，使用时请注意。

为配合 LocatePCI 使用，瑞博华公司还通过一个实用软件工具 PCICard，由它进行 PCI 卡的编号、资源查询和设定工作。

使用 LocatePCI 时，有两个概念需要确定，一是 PCI 板的顺序号(Order)，它依赖于板所在的 PCI 槽的位置，和它之前的槽有几块 PCI 卡；另一概念是 PCI 卡的编号(CardNo)，它是通过本工具对每块卡设置的一个编号，保存在卡上的非散失性内存中，应用程序应依赖它来区分和定位各 PCI 卡，未设置前可能为 0xffff。LocatePCI 可以设定各卡的编号，并查询各编号的卡的硬件资源(I/O 基地址)。详情可参见 5.3 节。

5.2 编程示例

瑞博华公司提供 C++Builder 编制的 PCICard 的源程序。并提供 VB 下使用 LocatePCI 的示例程序。在 IO700 的 VC 和 VB 使用示例中，也使用了 LocatePCI。

C/C++下使用 LocatePCI 需要的头文件为 LocatePCI.H。VC 下需要的库文件为 LocatePIC.LIB。C++Builder 下需要的库文件为_LocatePCI.LIB。VB 下编程需要的模块文件为 LocatePCI.BAS 或 LocatePCIDef.BAS。

演示 IO700 板功能的 VC 示例程序 VC_Test700 综合使用了 IOTools 和 LocatePCI 工具。同样功能的 VB 版演示程序见 VB_Test700，Delphi 版的演示程序见 Dtest700。

5.3 LocatePCI 函数详解

5.3.1 机器上安装的 RBH 公司 PCI 卡的数量 FindPCICard

C/C++下的原型定义：

int __stdcall FindPCICard(WORD * CardNoArray); //CardNoArray[10]

C/C++下的使用示例：

WORD CardNoArray[10];

I = FindPCICard(CardNoArray);

VB 下的定义

Declare Function FindPCICard Lib "LocatePCI.DLL" Alias "_FindPCICard@4" (ByRef CardNoArray As Integer) As Long ' //Dim CardNoArray[10] as Integer

VB 下的使用示例

Dim CardNoArray(10) as Integer

I = FindPCICard(CardNoArray(0))

功能及参数说明：

本函数返回当前计算机上安装的瑞博华公司的 PCI 卡的数量，并把各卡的编号返回在 CardNoArray 中。若当前计算机上未安装瑞博华公司的 PCI 卡，函数返回 0。

CardNoArray 的容留为至少 10 个短整数。CardNoArray[]反映了安装顺序与卡编号的关系，即 CardNoArray[Order]代表安装顺序为 Order 的卡的编号。

5.3.2 返回指定卡号的 PCI 卡的资源 GetIOBaseByCardNo

C/C++下的原型定义：

int __stdcall GetIOBaseByCardNo(WORD CardNo,WORD * IOBase, WORD * IRQNum);

C/C++下的使用示例：

WORD CardNo, IOBase, IRQNum;

I = GetIOBaseByCardNo(CardNo, &IOBase, &IRQNum);

VB 下的定义

Declare Function GetIOBaseByCardNo Lib "LocatePCI.DLL" Alias "_GetIOBaseByCardNo@12" (ByVal CardNo As Integer, ByRef IOBase As Integer, ByRef IRQNum As Integer) As Long

VB 下的使用示例

```
I = GetIOBaseByCardNo(CardNo, IOBase, IRQNum)
```

功能及参数说明：

若当前计算机上存在编号为 CardNo 的 PCI 卡，本函数返回 1，且返回它的 IO 基地址于 IOBase 中，使用的中断号于 IRQNum 中；若无卡号为 CardNo 的 PCI 卡存在，本函数返回 0。而卡号 CardNo 为通过函数 ChangeCardNo()或函数 SetCardNoByOrder()、或工具软件 PCICard 设置的编号。

5.3.3 返回指定安装顺序的 PCI 卡的资源 GetIOBaseByOrder

C/C++下的原型定义：

```
int __stdcall GetIOBaseByOrder(WORD Order,WORD * IOBase, WORD * IRQNum);
```

C/C++下的使用示例：

```
WORD Order, IOBase, IRQNum;
I = GetIOBaseByOrder(Order, &IOBase, &IRQNum);
```

VB 下的定义：

Declare Function GetIOBaseByOrder Lib "LocatePCI.DLL" Alias "_GetIOBaseByOrder@12" (ByVal Order As Integer, ByRef IOBase As Integer, ByRef IRQNum As Integer) As Long

VB 下的使用示例：

```
I = GetIOBaseByOrder(Order, IOBase, IRQNum)
```

功能及参数说明：

本函数返回安装顺序为 Order 的 PCI 卡的 IO 基地址于 IOBase 中，使用的中断号于 IRQNum 中，若成功，函数返回 1；若 Order 不正确，本函数返回 0。Order 的范围从 0 开始递增，安装顺序依赖于 PCI 卡安装的 PCI 槽的位置。

5.3.4 改变指定卡号的 PCI 卡的卡号 ChangeCardNo

C/C++下的原型定义：

int __stdcall ChangeCardNo(WORD OldCardNo,WORD NewCardNo);

C/C++下的使用示例：

I = ChangeCardNo(OldCardNo, NewCardNo);

VB 下的定义

Declare Function ChangeCardNo Lib "LocatePCI.DLL" Alias "_ChangeCardNo@8" (ByVal OldCardNo As Integer, ByVal NewCardNo As Integer) As Long

VB 下的使用示例

I = ChangeCardNo(OldCardNo,NewCardNo)

功能及参数说明：

本函数把编号为 OldCardNo 的 PCI 卡的编号改为 NewCardNo，若成功(存在编号为 OldCardNo 的 PCI 卡)则返回 1，否则，返回 0。

5.3.5 设定指定安装顺序的 PCI 卡的卡号 SetCardNoByOrder

C/C++下的原型定义：

int __stdcall SetCardNoByOrder(WORD Order,WORD NewCardNo);

C/C++下的使用示例：

I = SetCardNoByOrder(Order,NewCardNo);

VB 下的定义

Declare Function SetCardNoByOrder Lib "LocatePCI.DLL" Alias "_SetCardNoByOrder@8" (ByVal Order As Integer, ByVal NewCardNo As Integer) As Long

VB 下的使用示例

I = SetCardNoByOrder(Order, NewCardNo)

功能及参数说明：

本函数把安装顺序为 Order 的 PCI 卡的编号设置为 NewCardNo，若成功(存在安装顺序

为 Order 的 PCI 卡)则返回 1，否则，返回 0。安装顺序 Order 的范围从 0 开始递增，其值依赖于 PCI 卡安装的 PCI 槽的位置。

六 FrecordCore 模块编程说明

6.1 录波器模块简介

录波器模块封装了瑞博华公司网络录波器的所有功能，并以 FRecordCore.DLL(最好拷贝到系统目录 Windows\system 或 Winnt\system32 中)的方式提供，用户可在各种能直接调用 DLL 的编程语言中直接调用，并集成在用户程序的界面中。C++Builder 下的连接库为 FRecordCore.LIB，接口定义文件为 DllInterface.h。VB 下的接口定义文件为 CoreDef.BAS。VC 下使用接口头文件 FrecordCoreVC.H，并在工程中包含源文件 VCFRecordCore.CPP 即可，正如 VC 示例 VCTestCore 所演示的那样。Delphi 程序包含单元定义文件 FRecordCore.pas 即可，正如 Delphi 示例工程 DTestCore 所演示的那样。

本模块的函数的命名规则为：对于 C/C++的函数如: int __stdcall InitialCore(void);其在 DLL 中的名称即为 InitialCore；在 C++Builder 和 VC 中直接以名称 InitialCore()进行调用，在 C++Builder 中，通过连接到库文件 FrecordCore.Lib 即可直接调用 DLL 中的对应函数；在 VC 中，使用瑞博华公司提供的 VCFRecordCore.CPP 中函数 int __stdcall InitialFRecordCoreInterface(void)后，(此函数在内部用 LoadLibrary()和 GetProcAddress()等 Windows 的 API 定位 DLL 中的各函数)，就可直接按 InitialCore 等名字使用 DLL 中各函数，但应保证函数 InitialFRecordCoreInterface 返回值为 TRUE，以确保正确版本的 FrecordCore.DLL 存在于系统中。在 VB 中，通过模块定义文件 CoreDef.BAS 的 Declare Function InitialCore Lib "FRecordCore" () As Long 等指令就可直接使用 DLL 中各函数。在 Delphi 中，通过单元定义文件 FrecordCore.pas 中的 function InitialCore: integer ;stdcall; external'FRecordCore.dll'等质量也把相关函数直接连接到 DLL 中。

6.2 调用方法说明

本模块提供了丰富的调用接口，可被 VC、VB、C++Builder、Delphi 等直接调用。用户只需设计一个简单的程序(VB、C++Builder、VC 等)，通过调用本模块(DLL 方式提供)，应用程序就拥有强大的数据采集、显示、分析、记录、重放等功能和界面。本程序包含有 C++Builder、VB 下的示例程序源代码。下面是程序接口的调用规则：

规则 1：在调用本模块的任何功能前，应先调用 InitialCore()函数进行初始化，在应用程序退出前，应调用 CloseCore()函数以释放模块分配的资源。

规则 2：本程序模块可以在后台(不显示界面)进行数据采集、分析、保存，用户程序可以选择是否显示本模块的界面，或界面的不同部分，或在显示后重新隐藏界面，这可以通过 ShowMainForm()、HideMainForm()、ShowMenu()、HideMenu()、ShowToolBar()、HideToolBar()等来实现。

规则 3：通过调用 StartFRecord()或 StartOscil()函数来开始或停止录波器或双踪示波器的工作，这两个函数均是头一次启动(未采样前调用时)，第二次停止(开始采样后调用时)，再一次又启动。如果录波器界面是显示的，用户有可能通过录波器界面中进行启动或停止采样操作，因此，当前是处于采样还是停止状态，可通过 QuerySampStatus()函数进行查询。

规则 4：在调用 InitialCore()后，可通过 GetSimpleConf()函数查询录波器的基本配置信息，也可通过 DumpConfigFromFRecord 得到录波器的信息配置信息。同时，可通过 SetConfigToFRecord()详细配置录波器的各种参数，或在录波器内部进行参数配置后，通过 LoadConfigFromFile()或 SaveConfigToFile()函数把配置信息从配置文件中读取或保存。如果特别需要，高级用户还可以通过配对使用 DumpConfigFromFRecord 和 SetConfigToFRecord 来修改录波器内部的各种配置参数，但使用这组命令时要特别小心，因为涉及到 FRecordCore 的内部数据结构，设置错误或导致不可预计的后果，不鼓励用户使用。

规则 5: 录波器开始采样后，可通过 GetADBuf()取得最新一块采集缓冲区的内容，或通过 GetRAMBuf()取得连续储存的工作缓冲区中任意位置的采集内容，或通过 GetDispBufAny()拷贝当前显示缓冲区中的采集结果。同时，可以通过 GetEngineeringValueAllChn()函数返回采集结果的工程量纲值。

规则 6：若以示波器方式进行采集，录波器模块在工作时，用户重新还可以在任意时候直接调用 ADCard.DLL 中的 GetSanpshot()函数，直接从驱动程序中返回最新采集结果。

规则 7：对于 C/C++用户，还可以调用 SetupADCallBack()函数，在录波器模块每得到一块采集数据时，回调用户设置的处理程序，并把当前采集数据的指针通过给用户处理程序，实现采集数据的连续、实时处理。使用示例可参见 C++Builder、VC、Delphi 下示例程序源代码。

规则 8：对 VC++程序，头文件定义在 FrecordCoreVC.h 中，程序在初始时必须调用函数 InitialFRecordCoreInterface(实现在文件 VCFrecordCore.cpp)，且必须返回 1，才能调用其它 FRecordCore.DLL 中的其它函数，且用户程序的工程中必须包含 VCFrecordCore.cpp 文件。

VC 编程示例见 VCTestCore.ZIP。

C++Builder 编程示例见 CBTestCore.ZIP。

VB 下编程示例见 VBTestCore.ZIP。

Delphi 下编程示例见 DtestCore.ZIP。

6.3 详细编程接口说明

6.3.1 系统初始化及结束

//system initial & cleanup

1. 初始化录波器核心功能模块。

C 语言调用接口：

```
int __stdcall InitialCore(void);
```

VB 调用接口：

```
Declare Function InitialCore Lib "FRecordCore" () As Long
```

功能及参数说明：

在使用如下任何功能前应先调用此函数进行初始化。初始化成功则函数返回 1，失败则返回 0。失败的原因可能是初始化成功后再调用此函数，而且未调用 CloseCore()函数。

2. 关闭录波器核心功能模块。

C 语言调用接口：

```
int __stdcall CloseCore(void);
```

VB 调用接口：

```
Declare Function CloseCore Lib "FRecordCore" () As Long
```

功能及参数说明：

程序退出之前，应调用此函数，以便释放所分配的资源，并进行必要的善后处理。调用成功函数返回 1，失败返回 0。

6.3.2 窗口管理

// Form management

1. 显示录波器主界面。

C 语言调用接口：

```
int   __stdcall ShowMainForm(void);
```

VB 调用接口：

Declare Function ShowMainForm Lib "FRecordCore" () As Long

功能及参数说明：

录波器的主界面是录波器所有功能的控制中心，并且是波形显示的窗口。需要调用此函数才能显示。调用成功函数返回 1，失败返回 0。

2. 隐藏录波器主界面。

C 语言调用接口：

int __stdcall HideMainForm(void);

VB 调用接口：

Declare Function HideMainForm Lib "FRecordCore" () As Long

功能及参数说明：

录波器的主界面是录波器所有功能的控制中心，并且是波形显示的窗口。调用此函数将从程序隐藏它。调用成功函数返回 1，失败返回 0。

3. 显示录波器主菜单

C 语言调用接口：

int __stdcall ShowMenu(void);

VB 调用接口：

Declare Function ShowMenu Lib "FRecordCore" () As Long

功能及参数说明：

调用此函数才能显示录波器主界面上的菜单项。调用成功函数返回 1，失败返回 0。

4. 隐藏录波器主菜单

C 语言调用接口：

int __stdcall HideMenu(void);

VB 调用接口：

Declare Function HideMenu Lib "FRecordCore" () As Long

功能及参数说明：

调用此函数隐藏录波器主界面上的菜单项。调用成功函数返回 1，失败返回 0。

5. 显示录波器主工具栏

C 语言调用接口：

```
int   __stdcall ShowToolBar(void);
```

VB 调用接口：

```
Declare Function ShowToolBar Lib "FRecordCore" () As Long
```

功能及参数说明：

调用此函数才能显示录波器主界面上的工具栏。调用成功函数返回 1，失败返回 0。

6. 隐藏录波器主工具栏

C 语言调用接口：

```
int   __stdcall HideToolBar(void);
```

VB 调用接口：

```
Declare Function HideToolBar Lib "FRecordCore" () As Long
```

功能及参数说明：

调用此函数隐藏录波器主界面上的工具栏。调用成功函数返回 1，失败返回 0。

7. 显示录波器版本说明窗口

C 语言调用接口：

```
int   __stdcall ShowAbout(void);
```

VB 调用接口：

```
Declare Function ShowAbout Lib "FRecordCore" () As Long
```

功能及参数说明：

调用此函数将显示录波器的版本说明窗口，此窗口几秒钟后将自动隐藏。调用成功函数返回 1，失败返回 0。

8. 在主窗口中显示启动/停止录波器的命令按钮

C 语言调用接口：

```
int   __stdcall ShowStartButton(void);
```

VB 调用接口：

Declare Function ShowStartButton Lib "FRecordCore" () As Long

功能及参数说明：

调用此函数将在主窗口中显示启动/停止录波器/示波器的命令按钮，从而可以在主窗口中直接进行启动、停止操作。调用成功函数返回 1，失败返回 0。

9. 在主窗口中隐藏启动/停止录波器的命令按钮

C 语言调用接口：

int __stdcall HideStartButton(void);

VB 调用接口：

Declare Function HideStartButton Lib "FRecordCore" () As Long

功能及参数说明：

调用此函数将在主窗口中隐藏启动/停止录波器/示波器的命令按钮，从而不允许在主窗口中直接进行启动、停止操作，而只能在用户程序中通过 StartFRecord/StartOscil 进行启动，以在启动时完成必要的辅助工作。调用成功函数返回 1，失败返回 0。

10. 此调用使录波器显示波形曲线

C 语言调用接口：

int __stdcall FRecordViewWaveform(void);

VB 调用接口：

Declare Function FRecordViewWaveform Lib "FRecordCore" () As Long

功能及参数说明：

此函数使录波器显示采集的波形。FRecordViewWaveform()与 FRecordViewText()、FRecordViewGridDirectValue()、FRecordViewGridEngineeringValue()、FRecordViewGridHexValue()等是互斥的。

11. 此调用使录波器显示文本方式的运行日志

C 语言调用接口：

int __stdcall FRecordViewText(void);

VB 调用接口：

Declare Function FRecordViewText Lib "FRecordCore" () As Long

功能及参数说明：

此函数使录波器显示文字方式的运行日志。FRecordViewWaveform() 与 FRecordViewText()、FRecordViewGridDirectValue()、FRecordViewGridEngineeringValue()、FRecordViewGridHexValue()等是互斥的。

12. 此调用使 Grid 中的采集数据以直接采集量方式显示

C 语言调用接口：

int __stdcall FRecordViewGridDirectValue(void);

VB 调用接口：

Declare Function FRecordViewGridDirectValue Lib "FRecordCore" () As Long

功能及参数说明：

此函数使录波器在网格中显示文字方式采集结果，且是直接采集量方式(-5V~5V 的原始 AD 值)。FRecordViewWaveform()与 FRecordViewText()、FRecordViewGridDirectValue()、FRecordViewGridEngineeringValue()、FRecordViewGridHexValue()等是互斥的。

13. 此调用使 Grid 中的采集数据以工程量纲方式显示

C 语言调用接口：

int __stdcall FRecordViewGridEngineeringValue(void);

VB 调用接口：

Declare Function FRecordViewGridEngineeringValue Lib "FRecordCore" () As Long

功能及参数说明：

此函数使录波器在网格中显示文字方式采集结果，且是工程量纲方式。FRecordViewWaveform() 与 FRecordViewText() 、 FRecordViewGridDirectValue() 、 FRecordViewGridEngineeringValue()、FRecordViewGridHexValue()等是互斥的。

14.此调用使 Grid 中的采集数据以十六进制方式显示原始采集值

C 语言调用接口：

int __stdcall FRecordViewGridHexValue(void);

VB 调用接口：

Declare Function FRecordViewGridHexValue Lib "FRecordCore" () As Long

功能及参数说明：

此函数使录波器在网格中显示文字方式采集结果，且是十六进制、原始 AD 值方式(0~0xffff)。FRecordViewWaveform()与 FRecordViewText()、FRecordViewGridDirectValue()、FRecordViewGridEngineeringValue()、FRecordViewGridHexValue()等是互斥的。

6.3.3 录波器、示波器操作

//FRecord / Oscil operation

1. 启动/停止录波器的连续记录方式(录波器方式)

C 语言调用接口：

int __stdcall StartFRecord(void);

VB 调用接口：

Declare Function StartFRecord Lib "FRecordCore" () As Long

功能及参数说明：

调用此函数将启动或停止录波器的连续采集工作方式，头一次调用启动，再一次调用停止，再一次将再启动。调用成功函数返回 1，失败返回 0。

2. 启动/停止录波器的示波器方式

C 语言调用接口：

int __stdcall StartOscil(void);

VB 调用接口：

Declare Function StartOscil Lib "FRecordCore" () As Long

功能及参数说明：

调用此函数将启动或停止录波器的录波器工作方式，头一次调用启动，再一次调用停止，再一次将再启动。调用成功函数返回 1，失败返回 0。

3. 查询录波器当前工作方式

C 语言调用接口：

int __stdcall QuerySampStatus(void);

VB 调用接口：

Declare Function QuerySampStatus Lib "FRecordCore" () As Long

功能及参数说明：

调用此函数将查询录波器的工作方式。函数返回 0 表示调用失败，返回 1 表示录波器处于未采集(未启动) 状态，返回 2 表示处于连续采集状态，返回 3 表示处于示波器状态。

6.3.4 配置信息

//config information

1. 得到录波器简单配置信息

C 语言调用接口：

```
int   __stdcall GetSimpleConf(int * Analog_Num,
                              int * SW_Num,
                              int * NumChn,
                              int * SampNum,
                              int * MaxBlock,
                              int * ADBit,
                              float * ChnFrq,
                              short * AmpGain,
                              int * SimulateAD);
```

VB 调用接口：

```
Declare Function GetSimpleConf Lib "FRecordCore" (ByRef AnalogNum As Long, ByRef SWNum As Long, _
                                                  ByRef NumChn As Long, ByRef SampNum As Long, _
                                                  ByRef MaxBlock As Long, ByRef ADBit As Long, _
                                                  ByRef ChnFrq As Single, ByRef AmpGain As Integer, ByRef SimulateAD As Long) As Long
```

功能及参数说明：

调用此函数将查询录波器的简单配置信息。函数返回 0 表示调用失败，返回 1 表示需要的参数正确返回：Analog_Num 表示模拟量通道数；SW_Num 表示开关量通道数；NumChn 表示模拟量＋开关量合起来占的模拟量通道位置数；SampNum 表示每块采样数；MaxBlock

表示连续采集方式下内存缓冲区的块数；ADBit 反映当前使用的 AD 芯片的分辨率位数; ChnFrq 表示每通道的实际采样频率, AmpGain 返回程控放大倍数，SimulateAD 返回录波器模块设置的是使用实际 AD 采集板或是仿真数据(0 为实际采集板)。

2. 返回通道属性配置缓冲区 structChnProp[MAX_LINES]的尺寸

C 语言调用接口：

```
int __stdcall QueryChnPropSize(void);
```

VB 调用接口：

```
Declare Function QueryChnPropSize Lib "FRecordCore" () As Long
```

功能及参数说明：

此调用作为 GetChnProp 的正确性的保证。

3. 返回通道属性的详细配置，

C 语言调用接口：

```
int __stdcall GetChnProp(char * CardName,
                         void * pChnProp,
                         void * pSWConfig);
```

VB 调用接口：

```
Declare Function GetChnProp Lib "FRecordCore" _
        (ByRef CardName As Any, _
         ByRef pChnProp As Any, _
         ByRef pSWConfig As Any) As Long
```

功能及参数说明：

CardName 应为 16 字节的字符数组

pChnProp 应指向 structChnProp ChnProp[MAX_LINES]

pSWConfig 应指向 structSWConfig SWConfig[MAX_SW]

调用成功时，返回 true,否则返回 False

4. 根据配置文件重新配置录波器

C 语言调用接口：

```
int   __stdcall SetCompleteConfig(char * ConfigFileName);
```

VB 调用接口：

Declare Function SetCompleteConfig Lib "FRecordCore" (ByVal ConfigFileName As String) As Long

功能及参数说明：

调用此函数将迫使录波器按 ConfigFileName 指定的配置文件重新配置。函数返回 0 表示调用失败，返回 1 成功地重新配置。

5. 返回系统配置缓冲区 ConfigBuf 的尺寸

C 语言调用接口：

int __stdcall QueryConfigBufSize(void);

VB 调用接口：

Declare Function QueryConfigBufSize Lib "FRecordCore" () As Long

功能及参数说明：

此函数返回录波器配置缓冲区 ConfigBuf 的大小，以便应用程序正确分配缓冲区尺寸。

6. 从指定文件中读取配置信息于缓冲区 ConfigBuf

C 语言调用接口：

int __stdcall LoadConfigFromFile(void * ConfigBuf,char * ConfigFileName);

VB 调用接口：

Declare Function LoadConfigFromFile Lib "FRecordCore" (ByRef ConfigBuf As structFRecordCFG, ByVal ConfigFileName As String) As Long

功能及参数说明：

ConfigBuf 应指向结构 struct structFRecordConfig，将存放 ConfigFileName 中的配置信息。

7. 把缓冲区 ConfigBuf 中的配置信息存入指定文件

C 语言调用接口：

int __stdcall SaveConfigToFile(void * ConfigBuf,char * ConfigFileName);

VB 调用接口：

Declare Function SaveConfigToFile Lib "FRecordCore" (ByRef ConfigBuf As structFRecordCFG, ByVal ConfigFileName As String) As Long

功能及参数说明：

此函数把 ConfigBuf 中的配置信息写入文件 ConfigFileName。

8. 把系统配置信息复制入缓冲区 ConfigBuf

C/C++语言调用接口：

int __stdcall DumpConfigFromFRecord(void * ConfigBuf);

VB 调用接口：

Declare Function DumpConfigFromFRecord Lib "FRecordCore" (ByRef ConfigBuf As structFRecordCFG) As Long

功能及参数说明：

此函数把 FRecordCore 的完整配置信息存放在用户程序提供的缓冲区 ConfigBuf 中，供进一步修改等操作用。而 ConfigBuf 应指向一个结构 struct structFRecordConfig，此结构在 FRecordConfig.h 或 CoreDef.bas 中定义。这是一个很有用的功能，但使用时要特别小心，因为涉及到 FRecordCore 的内部数据结构。

注意，结构 struct structFRecordConfig 的大小应等于 QueryConfigBufSize 返回的值，然后才能修改。因为不同版本的 FRecordCore 的 struct structFRecordConfig 可能定义不同，不同 C/C++语言中结构中各数据成员的对齐模式可能不同(此结构要求编译时按双字(4 个字节)对齐结构，在 C++Builder 和 VC 中均要修改编译开关)，因此，必须满足 QueryCOnfigBufSize=sizeof(struct structFRecordConfig)后才能操作。

9. 把缓冲区 ConfigBuf 中的配置信息在 Frecord 中生效

C/C++语言调用接口：

int __stdcall SetConfigToFRecord(void * ConfigBuf);

VB 调用接口：

Declare Function SetConfigToFRecord Lib "FRecordCore" (ByRef ConfigBuf As structFRecordCFG) As Long

功能及参数说明：

此函数使录波器使用缓冲区 ConfigBuf 中的配置信息，即使 FRecordCore 重新使用存放在用户程序提供的缓冲区 ConfigBuf 的完整配置信息。而 ConfigBuf 应指向一个结构 struct structFRecordConfig，此结构在 FRecordConfig.h 或 CoreDef.bas 中定义。而且，应该是先用

DumpConfigFromFRecord 把系统配置存放在缓冲区 ConfigBuf 中，再修改相应内部变量，然后再使用 SetConfigToFRecord，这样才能保证配置信息的完整性和有效性。**这是一个很有用的功能，通过这种方法用户程序可以设置录波器的几乎所有采集参数。**但使用时要特别小心，因为涉及到 FRecordCore 的内部数据结构。

注意，结构 struct structFRecordConfig 的大小应等于 QueryConfigBufSize 返回的值，然后才能修改。因为不同版本的 FRecordCore 的 struct structFRecordConfig 可能定义不同，不同 C/C++语言中结构中各数据成员的对齐模式可能不同(此结构要求编译时按双字(4 个字节)对齐结构，在 C++Builder 和 VC 中均要修改编译开关)，因此，必须满足 QueryCOnfigBufSize=sizeof(struct structFRecordConfig)后才能操作。

对 FRecordCore 内部配置信息的修改，在 C++Builder/VC/VB 的例程中均有演示。

10. 此调用设置万能页显示的通道

C 语言调用接口：

int __stdcall SetVPage(int nChnAD,int nChnSW, int * DispChn);

VB 调用接口：

Declare Function SetVPage Lib "FRecordCore" (ByVal nChnAD As Long, ByVal nChnSW As Long, ByRef DispChn As Long) As Long

功能及参数说明：

其中：DispChn 中依次排列 nChnAD 个模拟量通道号(0~Analog_Num-1)，

和 nChnSW 个开关量通道号(0~SW_Num-1)

11. 此调用设置示波器方式下显示的两通道的属性，

C 语言调用接口：

int __stdcall SetOscilChn(int ADChn1,int ADChn2,int ADChn1Enabled,int ADChn2Enabled);

VB 调用接口：

Declare Function SetOscilChn Lib "FRecordCore" (ByVal ADChn1 As Long, ByVal ADChn2 As Long, ByVal ADChn1Enabled As Long, ByVal ADChn2Enabled As Long) As Long

功能及参数说明：

其中： 综 1 和综 2 是否显示(ADChn1Enabled,ADChn2Enabled)、

和显示的采集通道号(ADChn1,ADChn2，其中 iChn>=Analog_Num 表示是开关量通道(iChn-Analog_Num))

12. 读取系统缺省配置

C 语言调用接口：

int __stdcall LoadDefaultFRecondConfig(void);

VB 调用接口：

Declare Function LoadDefaultFRecondConfig Lib "FRecordCore" () As Long

功能及参数说明：

从当前目录中以采集板名称为名称或 Default.FCG 为名称的配置文件中读取系统配置

13. 执行"系统参数|从文件读取配置信息"菜单命令

C 语言调用接口：

int __stdcall FRecordLoadSysConfigWithPrompt(void);

VB 调用接口：

Declare Function FRecordLoadSysConfigWithPrompt Lib "FRecordCore" () As Long

功能及参数说明：

此函数使录波器模块执行"系统参数|从文件读取配置信息"菜单命令。

14. 执行"系统参数|当前配置信息存文件"菜单命令

C 语言调用接口：

int __stdcall FRecordSaveSysConfigWithPrompt(void);

VB 调用接口：

Declare Function FRecordSaveSysConfigWithPrompt Lib "FRecordCore" () As Long

功能及参数说明：

此函数使录波器软件模块执行"系统参数|当前配置信息存文件"菜单命令。

6.3.5 AD 数据管理

//AD Data manifest

1. 拷贝录波器中当前单 AD 缓冲区的采集结果

C 语言调用接口：

int __stdcall GetADBuf(WORD * Buf);

VB 调用接口：

Declare Function GetADBuf Lib "FRecordCore" (ByRef Buf As Integer) As Long

功能及参数说明：

调用此函数将把当前最新的 AD 缓冲区的采集结果拷贝到 Buf 中。Buf 要求容量为 NumSamp*NumChn+1 个 WORD，第一个 WORD 为序号，以后顺序存放各通道(共 NumChn 通道)的 NumSamp 个采样点的值(原始值)。调用成功函数返回 1，失败返回 0。

2. 拷贝录波器中当前连续 AD 缓冲区的采集结果

C 语言调用接口：

int __stdcall GetRAMBuf(WORD * Buf,int StartPoint, int EndPoint);

VB 调用接口：

Declare Function GetRAMBuf Lib "FRecordCore" (ByRef Buf As Integer, ByVal StartPoint As Long, ByVal EndPoint As Long) As Long

功能及参数说明：

录波器工作在连续采集方式时，调用此函数将把多缓冲区中的 AD 采集结果拷贝到 Buf 中，多缓冲区的起始点位置为 StartPoint，终止点位置为 EndPoint，其范围为 0～NumSamp*MaxBlock-1。Buf 要求容量为(EndPoint-StartPoint+1)*NumChn 个 WORD，各通道(共 NumChn 通道)、各采样点的原始值顺序存放。调用成功函数返回 1，失败返回 0。

3. 拷贝录波器中当前示波器方式的采集结果

C 语言调用接口：

int __stdcall FromSnapshot(WORD * Buf);

VB 调用接口：

Declare Function FromSnapshot Lib "FRecordCore" (ByRef Buf As Integer) As Long

功能及参数说明：

录波器工作在示波器方式时，调用此函数将强制录波器再从 AD 板中获取最新采样数据，并把它拷贝到 Buf 中。Buf 要求容量为 NumSamp*NumChn＋1 个 WORD，第一个 WORD 存放 AD 块的序号，以后顺序存放各通道(共 NumChn 通道)、各采样点的 AD 值(原始值)。

调用成功函数返回 1，失败返回 0。

4. 查询录波器当前显示波形的格式

C 语言调用接口：

int __stdcall QueryDispBuf(int * NumChn0, int * SW_Num0, int * PointNum);

VB 调用接口：

Declare Function QueryDispBuf Lib "FRecordCore" (ByRef NumChn0 As Long, ByRef SW_Num0 As Long, ByRef PointNum As Long) As Long

功能及参数说明：

调用此函数将返回当前录波器显示波形的格式(有可能显示的是连续采集、示波器、或历史数据等不同的可能)。NumChn0 为总通道数，其中有 SW_Num0 个开关量通道(占用(SW_Num0+15)/16 个模拟量通道的位置)，PointNum 为总采样点数。调用成功函数返回 1，失败返回 0。

5. 拷贝录波器中当前显示的采集结果

C 语言调用接口：

int __stdcall GetDispBufAny(WORD * Buf, int StartPoint, int EndPoint);

VB 调用接口：

Declare Function GetDispBufAny Lib "FRecordCore" (ByRef Buf As Integer, ByVal StartPoint As Long, ByVal EndPoint As Long) As Long

功能及参数说明：

调用此函数将把当前录波器显示的缓冲区中的指定 AD 数据拷贝到 Buf 中。Buf 要求容量为(EndPoint-StartPoint+1)*NumChn 个 WORD，各通道(共 NumChn 通道)、各采样点的 AD 值(原始值) 顺序存放。调用成功函数返回 1，失败返回 0。

6. 查询录波器当前显示波形的范围

C 语言调用接口：

int __stdcall QueryDisplayed(int * StartPoint, int * EndPoint);

VB 调用接口：

Declare Function QueryDisplayed Lib "FRecordCore" (ByRef StartPoint As Long, ByRef EndPoint As Long) As Long

功能及参数说明：

调用此函数将返回当前录波器在屏幕上显示波形在缓冲区中的范围。StartPoint 为起始位置，EndPoint 为结束位置。调用成功函数返回 1，失败返回 0。

7. 拷贝录波器中当前显示范围的采集结果

C 语言调用接口：

```
int   __stdcall GetDisplayed(WORD * Buf);
```

VB 调用接口：

```
Declare Function GetDisplayed Lib "FRecordCore" (ByRef Buf As Integer) As Long
```

功能及参数说明：

调用此函数将把当前录波器屏幕上显示范围内的 AD 数据拷贝到 Buf 中。Buf 要求容量为(EndPoint-StartPoint+1)*NumChn 个 WORD，各通道(共 NumChn 通道)、各采样点的 AD 值(原始值) 顺序存放，EndPoint 和 StartPoint 为 QueryDisplayed()的返回值。调用成功函数返回 1，失败返回 0。

8. 查询录波器当前瞬时值标线的位置

C 语言调用接口：

```
int   __stdcall QueryMarked(int * StartPoint, int * EndPoint);
```

VB 调用接口：

```
Declare Function QueryMarked Lib "FRecordCore" (ByRef StartPoint As Long, ByRef EndPoint As Long) As Long
```

功能及参数说明：

调用此函数将返回当前录波器在屏幕上显示波形时双瞬时标线在整个缓冲区中的位置。StartPoint 为一个瞬时标线的位置，EndPoint 为另一个瞬时标线的位置。调用成功函数返回 1，失败返回 0。

9. 拷贝录波器中当前瞬时值波形范围内的采集结果

C 语言调用接口：

```
int   __stdcall GetMarked(WORD * Buf);
```

VB 调用接口：

Declare Function GetMarked Lib "FRecordCore" (ByRef Buf As Integer) As Long

功能及参数说明：

调用此函数将把当前录波器屏幕上双瞬时标线范围内的 AD 数据拷贝到 Buf 中。Buf 要求容量为(EndPoint-StartPoint+1)*NumChn 个 WORD，各通道(共 NumChn 通道)、各采样点的 AD 值(原始值) 顺序存放，EndPoint 和 StartPoint 为 QueryMarked()的返回值。调用成功函数返回 1，失败返回 0。

10. 返回 AD 采集结果多通道的工程量

C 语言调用接口：

int __stdcall GetEngineeringValueAllChn(double * EngBuff, int PointPos);

VB 调用接口：

Declare Function GetEngineeringValueAllChn Lib "FRecordCore" (ByRef EngBuff As Double, ByVal PointPos As Long) As Long

功能及参数说明：

此调用把显示缓冲区 DispBuf 中第 PointPos(逻辑)采样点(从 0 编号,显示全部波形时波形区上最左边的为第 0 点)的 Analog_Num 个通道的工程值返回在缓冲区 EngBuff 中。缓冲区 EngBuff 要求 Analog_Num 个双精度浮点数。调用成功函数返回 1，失败返回 0。

11.返回 AD 采集结果单通道的工程量

C 语言调用接口：

int __stdcall GetEngineeringValue1Chn(double * EngBuff, int PointPos, int iChn);

VB 调用接口：

Declare Function GetEngineeringValue1Chn Lib "FRecordCore" (ByRef EngBuff As Double, ByVal PointPos As Long, ByVal iChn As Long) As Long

功能及参数说明：

此调用把显示缓冲区 DispBuf 中第 PointPos(逻辑)采样点(从 0 编号,显示全部波形时波形区上最左边的为第 0 点)的第 iChn 通道的工程值返回在缓冲区 EngBuff 中。缓冲区 EngBuff 要求 1 个双精度浮点数。调用成功函数返回 1，失败返回 0。

12. 返回读取环境温度值

C 语言调用接口：

double __stdcall FRecordUpdateEnvTemp(WORD * ADBuf, int NumChn, int iPos);

VB 调用接口：

```
Declare Function FRecordUpdateEnvTemp Lib "FRecordCore" _
        (ByRef iADBuf As Integer, _
         ByVal iNumChn As Long, _
         ByVal iPos As Long) As Double
```

功能及参数说明：

注：若环境温度是由某传感器测量的话，此调用根据 NmChn 个通道的采集缓冲区 ADBuf,确定第 iPos 点处的环境温度值 (环境温度测量值在 ADBuf[iPos*NumChn+iChn]处，iChn 由 FRecord 中环境温度通道指定)。在调用 FRecordEngineeringValueFromBuff,FRecordEngineeringValueFromADValue 和 FRecordADValueFromEngineeringValue 之前,应调用此函数,以便正确地计算需要补偿的 J、K 型热电偶的环境温度。

13. 返回缓冲区中采集量的工程量纲值

C 语言调用接口：

double __stdcall FRecordEngineeringValueFromBuff(WORD * ADBuf, int NumChn, int iPos, int iChn);

VB 调用接口：

```
Declare Function FRecordEngineeringValueFromBuff Lib "FRecordCore" _
        (ByRef ADBuf As Integer, _
         ByVal NumChn As Long, _
         ByVal iPos As Long, _
         ByVal iChn As Long) As Double
```

功能及参数说明：

此调用返回 NumChn 通道的采集缓冲区 ADBuf 中，第 iChn 通道、第 iPos 位置处的工程量纲下的值(采集值原始量为 ADBuf[iPos*NumChn+iChn]，通道属性在 FRecord 中定义)

14. 返回特定采集值的工程量纲值

C 语言调用接口：

double __stdcall FRecordEngineeringValueFromADValue(float ADValue, int iChn);

VB 调用接口：

```
Declare Function FRecordEngineeringValueFromADValue Lib "FRecordCore" _
        (ByVal ADValue As Single, _
         ByVal iChn As Long) As Double
```

功能及参数说明：

此调用返回原始采集量 ADValue 的工程量纲下的值， 所处的第 iChn 通道的属性在 FRecord 中定义

15. 返回工程量纲对应的直接 AD 值

C 语言调用接口：

```
WORD __stdcall FRecordADValueFromEngineeringValue(double EngValue, int iChn);
```

VB 调用接口：

```
Declare Function FRecordADValueFromEngineeringValue Lib "FRecordCore" _
        (ByVal EngValue As Double, _
         ByVal iChn As Long) As Integer
```

功能及参数说明：

此调用返回工程量纲下值 EngValue 所对应的原始采集量值， 所处的第 iChn 通道的属性在 FRecord 中定义

16. 返回 RAMBUF 的详细信息

C 语言调用接口：

```
int __stdcall GetRAMBufInfo(WORD ** pRAMBuf,
                            int * pCntNumSamp,
                            int * pCntNumChn,
                            int * pMaxBlock,
                            int * pidBlock);
```

VB 调用接口：

```
Declare Function GetRAMBufInfo Lib "FRecordCore" (ByRef pRAMBuf As Long, _
                            ByRef pCntNumSamp As Long, _
                            ByRef pCntNumChn As Long, _
                            ByRef pMaxBlock As Long, _
                            ByRef pidBlock As Long) As Long
```

功能及参数说明：

注：此函数提供特殊的内部编程信息，使用要注意。

// pRAMBUF: 返回 RAMBUF 的地址，通过它可以直接访问 RAMBUF

// pCntNumSamp: 返回 CntNumSamp,块长度(每块的每通道采集点数)

// pCntNumChn: 返回 CntNumChn, 采集通道数

// pMaxBlock: 返回 pMaxBlock, RAMBuf 中的块数

// pidBlock:返回 idBlock,当前块在 RAMBuf 中的位置，0 为第 1 块

//

6.3.6 配置界面辅助工具

//Tool function

1. 调用 AD 板采集参数配置界面

C 语言调用接口：

int __stdcall SetupADCard(void);

VB 调用接口：

Declare Function SetupADCard Lib "FRecordCore" () As Long

功能及参数说明：

成功时返回 true，负责返回 false(0)。

2. 调用 AD 板采集通道配置界面

C 语言调用接口：

int __stdcall SetupChannel(void);

VB 调用接口：

Declare Function SetupChannel Lib "FRecordCore" () As Long

功能及参数说明：

成功时返回 true，负责返回 false(0)。

3. 调用录波器缓冲区配置界面

C 语言调用接口：

int __stdcall SetupRecordBuf(void);

VB 调用接口：

Declare Function SetupRecordBuf Lib "FRecordCore" () As Long

功能及参数说明：

成功时返回 true，负责返回 false(0)。

4. 调用录波器配色板配置界面

C 语言调用接口：

int __stdcall SetupColorPal(void);

VB 调用接口：

Declare Function SetupColorPal Lib "FRecordCore" () As Long

功能及参数说明：

成功时返回 true，负责返回 false(0)。

5. 调用录波器启动算法配置界面

C 语言调用接口：

int __stdcall SetupAlgorithm(void);

VB 调用接口：

Declare Function SetupAlgorithm Lib "FRecordCore" () As Long

功能及参数说明：

成功时返回 true，负责返回 false(0)。

6. 调用录波器设备专用参数配置界面

C 语言调用接口：

int __stdcall SetupIOCtl(void);

VB 调用接口：

Declare Function SetupIOCtl Lib "FRecordCore" () As Long

功能及参数说明：

成功时返回 true，负责返回 false(0)。

7. 调用录波器通道校准界面

C 语言调用接口：

int __stdcall SetupCalibrate(void);

VB 调用接口：

Declare Function SetupCalibrate Lib "FRecordCore" () As Long

功能及参数说明：

成功时返回 true，负责返回 false(0)。

8. 调用录波器万能页配置界面

C 语言调用接口：

int __stdcall SetupVPage(void);

VB 调用接口：

Declare Function SetupVPage Lib "FRecordCore" () As Long

功能及参数说明：

成功时返回 true，负责返回 false(0)。

9. 设置录波器 AD 采集回调处理函数

C 语言调用接口：

int __stdcall SetupADCallBack(void __stdcall fCallBack(WORD * ADBuf, int NumChn, int NumSamp));

功能及参数说明：

本函数设置录波器每采集一块结果时,用户程序对采集数据进行处理的回调函数.C/C++下回调函数的例子为:

```
void __stdcall CallBackFunction(WORD * ADBuf, int NumChn, int NumSamp)
{
    //用户对 ADBuf[1+NumChn*NumSamp]中的数据进行处理
}
```

调用方式为： SetupADCallBack(&CallBackFunction); 成功时返回 true，负责返回 false(0)。

注：此函数只能在 C/C++语言(要求支持函数地址)下使用,VB 下不支持。

10. 设置启动录波器前的回调处理函数

C 语言调用接口：

int __stdcall SetupPrepareStartFRecordCallBack(int __stdcall fCallBack(int SampStatus));

功能及参数说明：

本函数设置在启动录波器或示波器前，用户程序为检查或调整录波器的系统参数而进行处理的回调函数。注意，由于参数配置不合理，启动录波器可能失败，而且，启动过程中，如果要求使用从历史记录文件中回放数据的模拟调试功能，或网络远程启动功能，录波器的参数还可能变化(使用临时配置)，因此，此回调函数与 StartFRecordCallBack 用法稍有不同。而且，回调过 PrepareStartFRecordCallBack 后，可能不会回调 StartFRecordCallBack(因为启动不成功)，而且，两次回调时系统配置可能不同。特别注意的是，在此回调函数中可以修改系统配置，在 StartFRecordCallBack 中一定不能再修改配置参数。C/C++下回调函数的例子为:

```
int __stdcall PrepareStartFRecordCallBackFunction(int SampStatus)
{
    //用户对开始采样前的参数检查、停止等准备工作
    if(SampStatus==1) {
        //FRecord mode

    } else
    if(SampStatus==2) {
        // Oscil Mode

    }   else {
        // error, not possible

    }
    return true;     //return true,若系统参数正确，允许系统继续启动录波器
                        //否则，  return false，指示系统停止启动录波器的任务;
}
```

调用方式为： SetupPrepareStartFRecordCallBack(&PrepareStartFRecordCallBackFunction);

成功时返回 true，负责返回 false(0)。

注：此函数只能在 C/C++语言(要求支持函数地址)下使用,VB 下不支持。

11. 设置启动录波器成功后的回调处理函数

C 语言调用接口：

int __stdcall SetupStartFRecordCallBack(void __stdcall fCallBack(int SampStatus));

功能及参数说明：

本函数设置在启动录波器或示波器成功后，但在有实际 AD 采集数据到达前，用户程序进行必要准备工作而进行处理的回调函数。注意，由于参数配置不合理，启动录波器可能失败，而且，启动过程中，如果要求使用从历史记录文件中回放数据的模拟调试功能，或网络远程启动功能，录波器的参数还可能变化(使用临时配置)，因此，此回调函数与前一个 PrepareStartFRecordCallBack 用法稍有不同。特别注意的是，在此回调函数中一定不能再修改系统配置参数。C/C++下回调函数的例子为:

```
void __stdcall StartFRecordCallBackFunction(int SampStatus)
{
    //用户对启动成功后、但实际处理 AD 采样数据前的准备工作
    if(SampStatus==1) {
        //FRecord mode

    } else
    if(SampStatus==2) {
        // Oscil Mode

    }   else {
        // error, not possible

    }
    return;
}
```

调用方式为： SetupStartFRecordCallBack(&StartFRecordCallBackFunction); 成功时返回 true，负责返回 false(0)。

注：此函数只能在 C/C++语言(要求支持函数地址)下使用,VB 下不支持。

12. 设置停止录波器后的回调处理函数

C 语言调用接口：

int __stdcall SetupStopFRecordCallBack(void __stdcall fCallBack(int SampStatus));

功能及参数说明：

本函数设置停止录波器或示波器后，用户程序进行必要善后处理工作而进行处理的回

调函数。此回调函数与 StartFRecordCallBack 应成对出现并被成对调用。C/C++下回调函数的例子为:

```
void __stdcall StopFRecordCallBackFunction(int SampStatus)
{
    //程序自动或手动停止采样后的后处理工作
    if(SampStatus==1) {
        //FRecord mode

    } else
    if(SampStatus==2) {
        // Oscil Mode

    }   else {
        // error, not possible

    }
    return;
}
```

调用方式为： SetupStopFRecordCallBack(&StopFRecordCallBackFunction); 成功时返回 true，负责返回 false(0)。

注：此函数只能在 C/C++语言(要求支持函数地址)下使用,VB 下不支持。

6.3.7 杂类功能

//miscellenouse function

1. 使录波器工作于单机方式

C 语言调用接口：

```
int   __stdcall DoAlone(void);
```

VB 调用接口：

```
Declare Function DoAlone Lib "FRecordCore" () As Long
```

功能及参数说明：

调用此函数将使录波器工作在单机方式，即屏蔽网络采集功能。注意：程序会自动记忆网络工作模式，即使用户程序重新启动后。调用成功函数返回 1，失败返回 0。

2. 使录波器工作于现场采集站方式(服务器方式)

C 语言调用接口：

```
int    __stdcall DoServer(void);
```

VB 调用接口：

```
Declare Function DoServer Lib "FRecordCore" () As Long
```

功能及参数说明：

调用此函数将使录波器工作在现场采集站方式，即它采集的 AD 数据通过网络可供远程控制台读取。注意：程序会自动记忆网络工作模式，即使用户程序重新启动后。调用成功函数返回 1，失败返回 0。

3. 使录波器工作于远程控制台方式(客户站方式)

C 语言调用接口：

```
int    __stdcall DoClient(void);
```

VB 调用接口：

```
Declare Function DoClient Lib "FRecordCore" () As Long
```

功能及参数说明：

调用此函数将使录波器工作在远程控制台方式，即它通过网络从现场采集站获取 AD 数据。这时要求从键盘输入现场采集站的 IP 地址。注意：程序会自动记忆网络工作模式，即使用户程序重新启动后。调用成功函数返回 1，失败返回 0。

4. 使录波器显示远程遥控窗口

C 语言调用接口：

```
int    __stdcall DoRemoteControl(void);
```

VB 调用接口：

```
Declare Function DoRemoteControl Lib "FRecordCore" () As Long
```

功能及参数说明：

调用此函数将使录波器弹出远程控制窗口，从而通过 TCP/IP 协议遥控另外一台录波器。调用成功函数返回 1，失败返回 0。

5. 查询录波器的网络状态

C 语言调用接口：

int __stdcall QueryNetwork(int * IsAlone, int * Controlled);

VB 调用接口：

Declare Function QueryNetwork Lib "FRecordCore" (ByRef IsAlone As Long, ByRef Controlled As Long) As Long

功能及参数说明：

调用此函数返回当前录波器的网络状态，IsAlone=0 说明处于单机方式，=1 说明处于现场采集站方式，=2 说明处于远程控制台模式。Controlled=0 说明此录波器没有被别的录波器遥控，否则说明网络上有另一个录波器遥控着本录波器的运行。调用成功函数返回 1，失败返回 0。

6. 发送设备专用数据

C 语言调用接口：

int __stdcall CoreIOCtl(int InSize, char * InBuff, int OutSize, char * OutBuff);

VB 调用接口：

Declare Function CoreIOCtl Lib "FRecordCore" (ByVal InSize As Long, ByRef InBuff As Byte, ByVal OutSize As Long, ByRef OutBuff As Byte) As Long

功能及参数说明：

调用此函数向低层设备驱动程序发送专用数据。发送和接收的数据的格式依赖于具体使用的 AD 板驱动程序。调用成功函数返回 1，失败返回 0。

7. 端口操作

C 语言调用接口：

int __stdcall ToolInportB(int port);
int __stdcall ToolInport(int port);
int __stdcall ToolInportDW(int port);

void __stdcall ToolOutportB(int port,int value);
void __stdcall ToolOutport(int port,int value);
void __stdcall ToolOutportDW(int port,int value);

VB 调用接口：

Declare Sub ToolOutportB Lib "FRecordCore" (ByVal Port As Integer, ByVal Value As Integer)

Declare Sub ToolOutport Lib "FRecordCore" (ByVal Port As Integer, ByVal Value As Integer)

Declare Sub ToolOutportDW Lib "FRecordCore" (ByVal Port As Integer, ByVal Value As Long)

Declare Function ToolInportB Lib "FRecordCore" (ByVal Port As Integer) As Integer

Declare Function ToolInport Lib "FRecordCore" (ByVal Port As Integer) As Integer

Declare Function ToolInportDW Lib "FRecordCore" (ByVal Port As Integer) As Long

功能及参数说明：

调用此组函数将可直接操作硬件端口，。

8 字符串操作

int __stdcall ToolCopyString(char * dString, char * sString);

VB 调用接口：

Declare Sub ToolCopyString Lib "FRecordCore" (ByRef dBuf As Byte, ByVal sBuf As String)

Declare Sub ToolCopyToString Lib "FRecordCore" Alias "ToolCopyString" (ByVal dBuf As String, ByRef sBuf As Byte)

功能及参数说明：

此函数实现 C 语言的 strcpy 功能，可用于 VB 中进行由字符数组向字符串的赋值。

9 波形存盘操作

int __stdcall SaveWaveFormAsBitmap(char * FileName);

VB 调用接口：

Declare Function SaveWaveFormAsBitmap Lib "FRecordCore" (ByRef FileName As Byte) As Long

Declare Function SaveWaveFormAsBitmap1 Lib "FRecordCore" Alias "SaveWaveFormAsBitmap" (ByRef FileName As Byte) As Long

功能及参数说明：

此函数把当前屏幕上的采集波形按 Bitmap 文件的格式存盘，文件名由 FileName 指定

int __stdcall SaveWaveFormAsText(char * FileName);

VB 调用接口：

Declare Function SaveWaveFormAsText Lib "FRecordCore" (ByRef FileName As Byte) As Long

Declare Function SaveWaveFormAsText1 Lib "FRecordCore" Alias "SaveWaveFormAsText" (ByRef FileName As Byte) As Long

功能及参数说明：

此函数把当前屏幕上的采集数据按 Text 文件的格式存盘，文件名由 FileName 指定

10 FFT 运算

int __stdcall fft(float *x, float * y, int n); //FFT

int __stdcall ifft(float *x, float * y, int n); //逆 FFT

VB 调用接口：

Declare Function fft Lib "FRecordCore" (ByRef x As Single, ByRef y As Single, ByVal n As Long) As Long;

Declare Function ifft Lib "FRecordCore" (ByRef x As Single, ByRef y As Single, ByVal n As Long) As Long;

功能及参数说明：

本函数实现 FFT 和逆 FFT 变换。

输入： x 为输入序列的实部

y 为输入序列的虚部

n 为 fft 运算的点数，从 4 至 32768，要求为 2 的阶乘。

返回：

x 为变换结果的实部.

y 为变换结果的虚部。

FFT 变换(fft) 结果为：

X(k)＋j y(k) = sum [x(n)*exp(-j*2*pi*(k-1)*(n-1)/N) , n=1,N], 1 <= k <= N.

逆 FFT 变换(ifft)结果为：

x(n) + j y(n) = (1/N) sum[X(k)*exp(j*2*pi*(k-1)*(n-1)/N), k=1,N], 1 <= n <= N.

七、ADCardX.OCX 控件使用说明

7.1 VB 下控件使用方法

本控件可在 Windows 95/98 下支持 ActiveX 的任何开发环境中使用，如 Visual Baisc、Delphi、 C++Builder、Visual C++等。它把对北京瑞博华控制技术有限公司开发的各类 AD 采集板的控制命令以用户易于理解和使用的方式提供给用户，使用户以简单、高效地方式操作 AD 板，而不用处理底层的硬件细节。而且，本控件可支持本公司的系列 AD 板，即同一个控件和同一套用户程序可支持不同的 AD 板。

本控件使用方式极为简单，现以 VB 开发环境为例说明其使用方法：

首先，在 VB 环境中先引用此控件(在 Windows 95 系统目录下，文件名 ADCardX.OCX，控件类型为 ADCardX)；

其次，在用户的 Form 中，放置此控件，缺省控件名为 ADCardX1，它的界面比较简单，直接显示本机器中所安装的 AD 采集板的名称；

然后，可在属性窗口中设置 AD 板的各种属性，如 I/O 基地址、采样频率等，或在程序中直接用代码设置各属性。

然后，在程序中操作 AD 板的过程如下：

第一步先要调用 Initial()方法初始化 AD 板，返回值告诉用户初始化是否成功(=0 即失败)。

第二步要调用 StartIntr()方法或 StartSnapshot()方法让 AD 板开始采样；

第三步，为了得到采样结果，调用方法 GetADResult()或 GetADResultRecent()或 GetSnapshot()，就直接把采样结果存放在用户指定的数组中，然后用户可以直接在此数组中使用 AD 采样结果了，如存盘、图形显示或进行别的处理。

第四步，AD 采样结束时，调用 StopIntr()或 StopSnapshot()方法使 AD 板停止工作。

最后，需要说明的是本控件有两种工作方式：

一种是多缓冲区方式，用 StartIntr()方法启动。此时系统(驱动程序)内部开有 NumInnerBuf 个缓冲区，每个缓冲区可存放 NumChn 个通道、每通道 SampPerChn 个 AD 采样结果。AD 采集的结果顺序存放在各缓冲区中，GetADResult()方法就顺序地取出(最先)一个放满采样结果的缓冲区的内容(进行过此操作后此缓冲区又可接收新的采集结果)，而 GetADResultRecent()方法直接取最后(新)一个缓冲区的内容(进行过此操作后所有缓冲区又可重新接收新的采集结果)。在这种方式下，存放有 AD 采集结果的缓冲区数目可用方法 ResultBuf()进行查询，而且，在设置属性 EnableNotify 为 True 时，每一个缓冲区满时会给用户程序发送事件 Notify()，用户可在这个事件响应函数中进行取 AD 缓冲区的工作。此方

式用 StopIntr()停止。若没有及时从缓冲区取采样结果，而使所有缓冲区都满时，此后 AD 采样结果就将丢失，直到有缓冲区被 GetADResult()或 GetADResultRecent()释放、可存放新的采样结果为止。此工作方式适宜于连续记录 AD 采集结果。连续两次成功地 GetADResult()取得的 AD 结果在时间上不会重叠，之间是否丢失采样结果，要看返回结果的序列号(结果数组的第一个值，为无符号短整数)是否递增加一(注意，0xFFFF 加一后会变成 0)。只增一，说明两缓冲区在时间上是连续的。在采样率较低时，通过多缓冲区方式，有可能连续地记录、处理、保存长时间的 AD 结果。

另一工作方式为单缓冲区方式，用 StartSnapshot()方法启动，它始终不断地往缓冲区中存放 AD 结果，缓冲区大小为 NumChn 个通道，每通道 SampPerChn 个点。方法 GetSnapshot()从此缓冲区中得到所有通道、最新指定个点(点数在方法的参数中指定)的采集结果(的快照)，这时 EnableNotify 属性值一般应置为无效。此工作方式由命令 StopSnapshot()终止。此工作方式适宜于示波器方式，可随时得到最新的采集结果(的快照)。但两次 GetSanpshot()取得的结果在时间上可能有重叠或丢失，相对时间关系不能确定，或者通过函数 GetSnapshot(ByRef ADBuf as Integer, ByVal SampPerChn0 as Long, ByRef SampPtr as Long)返回的 SampPtr 值和返回的 ADBuf 第一个元素所代表的系列号来求出。

7.2 属性和方法说明

重要的属性含义如下，按名称、类型、含义进行介绍：

IOBase: Integer。指定 AD 板 I/O 基地址。

IRQNum: Integer。指定 AD 板占用的 IRQ 值(0-15)。

PhysAddr: Long。指定 AD 板占用的 RAM 地址。

DMAChn:Integer。指定 AD 板使用的 DMA 通道号。

NumInnerBuf: Integer。指定多缓冲区方式下内部缓冲区个数。

SampPerChn: Long。指定每个缓冲区内可存放的每通道采样点数。

BegChn: Integer。在单通道采集方式下，指定采集的通道号；多通道方式下，指定起始通道号；在既有模拟通道又有开关量通道的板上，BegChn 表示模拟通道道第 BegChn 及之后就存放开关量通道的值。注：有些 AD 板 BegChn 有定义，有些 AD 板可为任意值而无任何作用。

NumChn: Integer。指定要采集的总通道数。

FrqSamp: Long。指定的名义采样频率。而每通道实际采样频率可通过 CHnFrq 方法获得。

FrqFilter: Long。指定内部滤波器截止频率。对有些 AD 板此参数无作用。

AmpGain：Integer。指定板上程控放大器的增益。一般可为 1，2，4，8，10，100，1000。

对有些 AD 板此参数无作用。

EnableNotify: Boolean(True or False)。指定多缓冲区方式或单缓冲区方式下是否触发事件 Notify。

NofityLength：Long。指定单缓冲区方式下各通道采集 NotifyLength 个数时触发一个 Notify 事件。注：多缓冲区方式下，每缓冲区满就触发一个 Notify 事件，只要 EnableNotify=True。

ADCardName: String(只读，并显示在控件本身上)。返回 AD 板类型(名称)。

ResultBuf: Integer(只读)。返回已有采集结果的缓冲区数目。仅多缓冲区方式下有意义。

Visible: Integer(better set to False)。指定控件是否可件。为美观起见一般设为不可见。

MaxChn: Long(只读)。返回当前采集板支持的最多通道数(含开关量通道，每 16 个开关量占一个模拟量的内存地址)。

MaxBinChn：Long(只读)。返回当前采集板支持的最多开关通道数。

VZero: Single(只读)。返回当前采集板采集结果的格式：在 16 位无符号短整数表示的 AD 值中，电平 0 对应的值。

VMax: Single(只读) 。返回当前采集板采集结果的格式：在 16 位无符号短整数表示的 AD 值中，最高电平(+5V)对应的值。

注：由上可知：对 AD 采集结果 ADBuf(?)，它实际采用偏移二进制编码，它对应的实际电压为(单位：V)：

(ADBuf(?)-VZero)/(VMax-VZero)*5

但由于 VB 不支持无符号整数(只能按有符号整数对待)，对 16 位 AD 芯片(VZero=32768., VMax=65535.)，要得到电压值，需用如下公式：

```
If (ADBuf(1) > 0) Then
V = (ADBuf(1) - VZero) / (VMax - VZero) * 5
Else
V = (65536 + ADBuf(1) - VZero) / (VMax - VZero) * 5)
End If
```

重要的方法:

ChannelFrq(ByVal NumChn As Long, ByVal FrqSamp As Long) as Single:返回值的含义为：若属性 NumChn(仅指模拟通道数目)和 FrqSamp 置为本方法指定的参数时每通道的实际采样率(单位：Hz)。

Initial() as integer：初始化 AD 板。AD 板存在，且属性 IOBase、IRQNum、PhysAddr 正确时返回常数 ADCard_Success，否则返回 ADCard_Error(=0)。

StartIntr() as Long：启动多缓冲区工作方式。成功时返回每缓冲区的大小(字节数)，这时，用 GetADResult()或 GetADResultRecent()时用到的缓冲区大小应大于或等于此值(字节数)。失败时返回 0。

StopIntr() as Integer：停止多缓冲区工作方式。

StartSnapshot() as Integer：启动单缓冲区工作方式。成功时返回内部缓冲区的大小(字节数)。失败时返回 0。

StopSnapshot() as Integer：停止单缓冲区工作方式。

ResultBuf() as Integer：返回多缓冲区方式下已有采集结果的缓冲区数目。

GetADResult(ByRef ADBuf as Integer) as Integer：在多缓冲区方式下把最先的一个有采集结果的缓冲区内容拷贝到用户数组 ADBuf 中。并释放当前缓冲区。

这时用户数组尺寸要求大于 SampPerChn*NumChn+1 个短整数。

GetADResultRecent(ByRef ADBuf as Integer) as integer：在多缓冲区方式下把最新的一个有采集结果的缓冲区内容拷贝到用户数组 ADBuf 中，并释放所有内部缓冲区。

这时用户数组尺寸要求大于或等于 SampPerChn*NumChn+1 个短整数。

GetSnapshot(ByRef ADBuf as Integer, ByVal SampPerChn0 as Long, ByRef SampPtr as Long) as Integer：

在单缓冲区方式下把最新的采集结果(仅 SampPerChn0 个点、NumChn 个通道)拷贝到用户数组 ADBuf 中。最新数据在内部缓冲区中的位置返回在 SampPtr 中，它的范围为 0~SampPerChn-1，而内部缓冲区已被覆盖过几次，这值返回在数组 ADBuf 的第一个短整数中，即 SeqNo 中，用户可由这两个参数确定两次 GetSnapshot 取的结果的相对时间关系。要求 SampPerChn0<SampPerChn。

这时用户数组尺寸要求大于或等于 SampPerChn0*NumChn+3 个短整数。

IOCtl(ByVal InSize as Long, ByVal InBuff as String, ByVal OutSize as Long, ByVal OutBuff as String) as Long: 和 AD 板专用数据交换接口。

IOCtlByte(ByVal InSize as Long, ByRef InBuff as Byte, ByVal OutSize as Long, ByRef OutBuff as Byte) as Long: 和 AD 板专用数据交换接口的另一种参数类型。

InportB(ByVal Port as Integer) as Integer：单字节端口读。辅助工具接口函数。

Inport(ByVal Port as Integer) as Integer：单字端口读。辅助工具接口函数。

InportDW(ByVal Port as Integer) as Long：双字端口读。辅助工具接口函数。

OutportB(ByVal Port as Integer, ByVal Value as Integer)：单字节端口写。辅助工具接口函数。

Outport(ByVal Port as Integer, ByVal Value as Integer)：单字端口写。辅助工具接口函数。

OutportDW(ByVal Port as Integer, ByVal Value as Long)：双字端口写。辅助工具接口函数。

7.3 AD 缓冲区格式说明

返回 AD 结果格式说明：

方法 GetADResult()、GetADResultRecent()、GetSnapshot()把 AD 采集结果拷贝在用户数组 ADBuf()中，ADBuf 的格式定义为：

Dim ADBuf(NumChn*SampPerChn+1) as Integer

则： ADBuf(0)为此次结果的序列号：在多缓冲区方式下，每个缓冲区的序列号按采集时间先后顺序增一，若缓冲区满而丢失数据时此值会增加以表示丢失了多少缓冲区；在单缓冲区方式下，表示此次快照是系统内部缓冲区被覆盖(填满)多少此时的结果。此值是按无符号短整数递增的，在 65535(=0xffff)后会反转为 0 的。

ADBuf(1 至 NumChn*SampPerChn)顺序存放各通道、各时刻的采样值。第 n 通道(n=0 至 NumChn-1)、第 m 时刻(m=1 至 SampPerChn)就存放在数组单元 ADBuf((m-1)*NumChn+n+1)中，即 ADBuf(1)存放第 0 通道第 1 时刻的值，ADBuf(2)存放第 1 通道第 1 时刻值，.....，ADBuf(NumChn)存放第 NumChn-1 通道第 1 时刻的值，ADBuf(NumChn+1)存放第 0 通道第 2 时刻的值，.....。对 12 位 AD 板，采样结果为偏移二进制编码，即：0 表示最低电压值(如 -5V)，2048(=0x800)表示 0 电压，4095(=0xfff)表示最高电压(如+5V)。具体电压值可由属性 VZero 和 VMax 转换出。

若 AD 板既支持模拟量，又支持开关量，则在等价的 NumChn 个总通道中，通道 0 至 BegChn-1 为常规的模拟量，通道 BegChn 至 NumChn-1 为组合的开关量通道，每 16 个开关量通道占 1 个模拟量通道的缓冲区(即每模拟通道占 1 位，且通道 0 占短整数的 bit 0,通道 15 占短整数的 bit15，而每模拟通道占 16 位(无符号短整数))

7.4 杂类说明

1. 重要的事件:

Notify(): 在多缓冲区方式下，每一个内部缓冲区填满采集结果时调用一次此事件。

2. 重要的常数定义:

ADCard_Error = 0

ADCard_Success = 1

3. 注意事项:

此控件需要在\Windows\System 目录下的 ADCard.DLL 和相应的 VxD 驱动程序支持。

7.5 简单的 VB 示例

一个简单的使用示例如下(VB5):

建立一个窗体(Form)，在其上定义几个控件：Label1, Label2, ADCardX1,Command1,Timer1

拷贝如下代码，就建立起一个简单但完整的 AD 采集程序：(点击命令按钮就开始采样，再点击就停止。采样结果显示在 Label1(表示缓冲区序列号)和 Label2(表示第 iChn 通道、第 iSamp 点的采样值)上)。注意，在 Form_Load 中设置 WorkMode = 0 表示多缓冲区工作方式，否则为单缓冲区方式。代码如下：

```
Dim ADBuf() As Integer    'size-variable array
Dim Working As Integer
Dim WorkMode As Integer
Const NumChn = 1
Const SampPerChn = 300
'range for iChn: 0 -- NumChn-1
'range for iSamp: 1--SampPerChn
Const iChn = 0
Const iSamp = 1

Private Sub ADCardX1_Notify()
  'event handler for multiple buffer mode
  Dim i as Long
  i = ADCardX1.GetADResult(ADBuf(0))
  If i = ADCard_Success Then
    Label1.Caption = "SeqNo=" + CStr(ADBuf(0))
    Label2.Caption = "Chn" + CStr(iChn) + ":" + CStr(ADBuf((iSamp - 1) * NumChn + iChn + 1))
  End If
End Sub

Private Sub Command1_Click()
  Dim i as Long

  If Working = 0 Then
    'start command
    ADCardX1.IOBase = &H110       ‘不同的 AD 板此参数不一样
    ADCardX1.IRQNum = 5           ‘不同的 AD 板此参数不一样
    ADCardX1.PhysAddr = &Hd8000   ‘不同的 AD 板此参数不一样
```

```
 ADCardX1.DMAChn= 5          ‘不同的 AD 板此参数不一样
 ADCardX1.NumInnerBuf = 1
 If WorkMode = 0 Then
  'multiple buffers mode
  ADCardX1.NumInnerBuf = 2      'range: >=1
  ADCardX1.SampPerChn = SampPerChn
 Else
  'single buffer mode
  ADCardX1.NumInnerBuf = 1      'will be set to 1 internally
  ADCardX1.SampPerChn = SampPerChn * 2
 End If
 ADCardX1.BegChn = 0
 ADCardX1.NumChn = NumChn
 ADCardX1.FrqSamp = 20000
 ADCardX1.EnableNotify = True
 If ADCardX1.Initial() = ADCard_Error Then
  Label1.Caption = "Initial Error!"
  Exit Sub
 End If
 If WorkMode = 0 Then
  If ADCardX1.StartIntr() = ADCard_Error Then
   Label1.Caption = "Start Interrupt Error!"
   Exit Sub
  End If
 Else
  If ADCardX1.StartSnapshot() = ADCard_Error Then
   Label1.Caption = "Start Snapshot Error!"
   Exit Sub
  End If
  Timer1.Interval = 20      '20 ms interval
  Timer1.Enabled = True
 End If
 ReDim ADBuf(SampPerChn * NumChn + 1)
 Working = 1
 Command1.Caption = "Stop"
Else
 'stop command
 If WorkMode = 0 Then
```

```
    i = ADCardX1.StopIntr()
   Else
    i = ADCardX1.StopSnapshot()
    Timer1.Enabled = False
   End If

   Working = 0
   Command1.Caption = "Start"
  End If
End Sub

Private Sub Form_Load()
   Working = 0
   'set multiple/single buffers mode
   WorkMode = 0     '1
End Sub

Private Sub Timer1_Timer()
  'event handler for single buffer mode
  Dim SampPtr as Long, i as Long
  If Working = 0 Then Exit Sub
  i = ADCardX1.GetSnapshot(ADBuf(0), SampPerChn, SampPtr)
  If i = ADCard_Success Then
   Label1.Caption = "SeqNo=" + CStr(ADBuf(0))
   Label2.Caption = "Chn" + CStr(iChn) + ":" + CStr(ADBuf((iSamp - 1) * NumChn + iChn + 1))
  End If

End Sub
```

八、Labview 下编程示例说明

8.1、RBHCardOperation.llb 控件

- **功能说明：**

RBHCardOperation.llb 是瑞博华公司为用户提供的在 LabVIEW 下编程使用的一组板卡驱动程序 vi。当前版本 1.0，支持 6.0 及以上版本的 LabVIEW 开发环境使用。

该组 vi 包括了用户级的模拟量输入/输出，数字量输入/输出 vi 模块，以及为方便用户灵活编程需要的底层硬件操作 vi 模块。每个 vi 均带有详细的在线使用帮助，使用这些 vi，用户可以在 LabVIEW 环境下对本公司提供的硬件板卡进行快捷、方便的编程操作。

- **安装说明：**运行安装程序 RBH_LabVIEWsetup.exe，按程序提示正确安装本控件。或

者将 RBHCardOperation.llb 复制到 National Instruments\LabVIEW 7.0\user.lib 目录中，如用户将 Labview 安装到 C 盘中，则复制到 C:\Program Files\National Instruments\LabVIEW 7.0\user.lib 目录中，安装后如下图所示：

- **注意事项：**编程前需首先认真阅读所购板卡的硬件使用说明书，并确认当前系统中已正确安装了相应板卡的驱动程序。
- **检验是否安装成功：**安装完成后，重新启动 Labview 软件，进入框图面板，注意不是前面板。如下图所示。

- **控件简介：**

控件窗口见如图 1 所示。（每个 vi 的详细说明见在线帮助）驱动控件分为四类：

8.2、用户级 vi 模块

共 8 个 vi，其中包括 4 个模拟量输入 vi，一个模拟量输出 vi，两个数字量输入 vi 及一个数字量输出 vi。

用户可以使用这些 vi 方便的实现数字量的输入、输出，数字量的输入输出。

8.3、中级数据采集 vi 模块

为方便用户灵活编程和实现特定的数据采集要求而提供的模拟量输入（A/D）操作的 11 个 vi。

用户可以根据这些 vi 方便的实现数据连续采集及控制，包括单缓冲区采集方式操作和多缓冲区操作方式。具体编程方法见《编程指南》及本公司提供的 LabVIEW 下多缓采集例程。

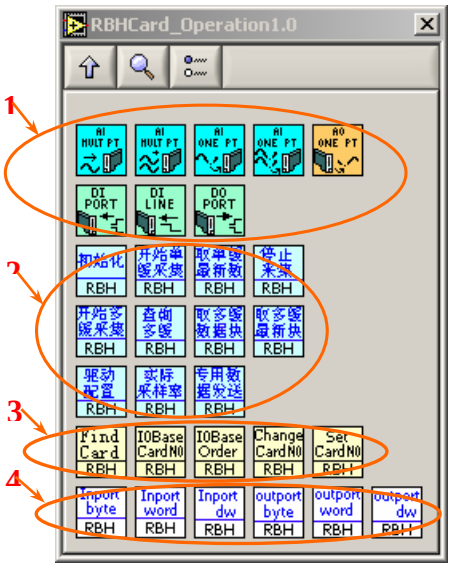


图 1 LabVIEW 下瑞博华板卡

8.4 多板卡操作 vi 模块

即 LocatePCI 中 5 个多板卡设置操作函数的 vi 模块。

用户可以根据这些 vi 进行多板卡的识别及标识等操作。

关于多板卡操作的详细介绍请参看《编程指南》及所构硬件的《使用说明书》。

8.5、底层 I/O vi 模块

4 个 vi，完成底层硬件端口的读/写操作。

8.6、多缓数据采集例程说明：

例程 RBH Cont Acq&Graph(multibuffered).vi 是采用上面描述的中级数据采集 vi 模块编写而成，采用多缓冲区方式实现了数据的连续采集和显示。用户可以参考或修改该例程以实现不同的数据采集目的。

查看例程的在线说明：在 LabVIEW 环境下点击“Help >> Show Context Help”或者使用组合键“Ctrl+H”，然后将鼠标移至程序右上角的图标处即可。