ART2003

WIN98/2000 驱 动 程 序 使 用 说 明 书

请您务必阅读《使用纲要》，他会使您事半功倍!

目 录

第一章 版权信息

第二章 绪 论

第一节、使用纲要

一、使用上层用户函数，高效、简单

如果您只关心通道及频率等基本参数，而不必了解复杂的硬件知识和控制细节，便可如能所需，那么我们强烈建议您使用上层用户函数，它们就是几个简单的形如 Win32 API 的函数，具有相当的灵活性、可靠性和高效性。诸如 WriteDeviceProDA 等。而底层用户函数如 WritePortByte、ReadPortByte……则是满足了解硬件知识和控制细节、且又需要特殊复杂控制的用户。但不管怎样，我们强烈建议您使用上层函数（在这些函数中，您见不到任何设备地址、寄存器端口、中断号等物理信息，其复杂的控制细节完全封装在上层用户函数中。）对于上层用户函数的使用，您基本上可以必参考硬件说明书，除非您需要知道板上 D 型插座等管脚分配情况。

二、如何管理设备

由于我们的驱动程序采用面向对象编程，所以要使用设备的一切功能，则必须首先用 CreateDevice 函数创建一个设备对象句柄 hDevice，有了这个句柄，您就拥有了对该设备的绝对控制权。然后将此句柄作为参数传递给其他函数，如 WriteDeviceProDA 函数可以用 hDevice 句柄实现对 AD 数据的采样读取。最后可以通过 ReleaseDevice 将 hDevice 释放掉。

三、如何实现 DA 的简便输出

当您有了 hDevice 设备对象句柄后，然后反复调用 WriteDeviceProDA 函数输出每一个 DA 数据。

四、哪些函数对您不是必须的？

当公共函数如 CreateFileObject, WriteFile,ReadFile 等一般来说都是辅助性函数，除非您要使用存盘功能。如果您使用上层用户函数访问设备，那么 WritePortByte, WritePortWord, WritePortULong, ReadPortByte, ReadPortWord, ReadPortULong 等函数您可完全不必理会，除非您是作为底层用户管理设备。它们可以帮助您在 NT、Win2000 等操作系统中实现对您原有传统设备如 ISA 卡、串口卡、并口卡的访问，而没有这些函数，您可能在新操作系统中无法继续使用您原有的老设备（除非您自己愿意去编写复杂的硬件驱动程序）。

第二节　驱动程序功能概述(不针对具体某一种设备)

一、数据传输采集方式

我公司提供的驱动程序完全支持***程序查询方式、硬件中断方式、直接内存存取 DMA）方式***。您从我公司所购买的硬件产品能支持的数据采集方式，我们的驱动程序均予以满足。

二、数据传输与数据处理的独立性

为了提高数据吞吐率以及实现实时数据处理(如随时取数、随时暂停设备、随时开始传输、随时存盘、随时显示波形、随时设备控制输出等功能), 我们采用一种最新、最灵活的设计思想，即数据采集传输和数据处理相独立的思想。即用我们所创建的设备对象在 Windows 系统空间管理一个一级强制性缓冲队列，该缓冲队列可支持 128K 字（即 256K 字节）的系统内存空间 Buffer，该队列采用先进先出策略和动态链表等技术来更高效地管理这个 Buffer。这个队列缓冲与用户数据缓冲区相独立，设备对象在后台负责数据采集和传输，将其数据映射到相应的队列缓冲单元，且维持一个动态链表，并向用户发送相应的通知消息。而用户则不必知道内部的任何复杂操作，而只须在这个消息到来时，使用 ReadDeviceIntAD 函数读一批 AD 数据或几批即可。重要的是，在这个消息没有到来时，用户代码不必花任何 CPU 时间去轮询等待，而用户正好利用这段空闲时间去处理更多的任务。即轻松实现了数据采集与数据处理的同步并发进行。这将是最高效的。这个队列缓冲跟先进先出存储器 FIFO 芯片功能基本一致，只不过这个缓冲是一个被软件仿真的 FIFO 存储器。使用这项技术的最大优点就是完全解决了在多任务环境中实现高速连续采集数据难的问题。特别是整个系统突然繁忙的时候，比如用户在高速采集数据或实时存盘时，偶而移动窗口或改变窗口大小或弹出对话框时，这项技术足以保证所采集的数据完整无缺。如果用户希望应用程序有更好的处理能力和克服操作系统的陡然忙碌对连续数据采集的影响，可以考虑在用户模式中再使用二级缓冲队列和相应的缓冲区链表技术。具体细节请参考 NT 下的中断演示程序。(目前在 Window NT 中完全支持此项技术，在以后的 Win2000 和 WinXP 版本中应该会进一步提供)。

三、连续不间断大容量采集存盘

在虚拟仪器、实验室数据分析、医疗设备、记录仪等诸多研究和应用领域中，对数据的要求很高，一方面数据容量较大，如几百兆甚至几仟兆，另一方面采样速度都较高，如 200KHz，300KHz 等，更重要是要求在高速长时间的采集数据过程中，不能丢掉一个点，必须全部存入硬盘，同时还要进行一些点的抽样分析，这在 DOS 环境中实现起来就有较大的难度，就更别说在 Windows 这样的多任务环境中（对于 Windows 多任务机制请参阅有关 Windows 手册）。大家知道 Windows 的各应用程序总是不断地被任务调度器调度，循环处在睡眠、排队、就绪、触发运行等状态中。Win95 任务之间的切换密度至少大于 1 毫秒，那么如果要以 300KHz 频率采样（即每 3.3 微秒就得传输一个数据），很显然有大量的数据在传输中由于任务之间的切换而被丢失掉。这就是基于 Windows 客户程序在传统模式下，高速连续采集传输数据时所具有的局限性。为了突破这种局限性，就得采用别的办法，如非客户程序、内核程序、驱动程序（如 VxD、微代码）等，再加上我们所掌握的新技术，如内存映射、直接写盘技术以及独有的设计思想便可以很好的解决这些问题。从 1998 年 9 月开始，已有部分用户实际使用，反映良好。我们自己也经过全面测试，比如在 Windows95 下使用无 FIFO 芯片的 BH5104 模板，实际结果是：以 200KHz 频率，双通道采集正弦波且存盘，写满整个硬盘近 4000 兆数据，其时间长达 6 个小时左右，随后再读盘回放磁盘数据，整个波形没有发现任何串道、断点和畸形状。当然 ART2003 等设备同样具这样的性能。它不仅具有一级硬件缓冲 FIFO(其缓冲深度可调 1KB、2KB、4KB、8KB、16KB 等)，同样具有第二节中叙述的二级强制队列缓冲，这个软件防真的缓冲比一级缓冲要大几十倍。如果用户需要的话，可以在应用程序中再建立循环式用户缓冲，即可实现高速不间断大容量采集存盘功能。

四、后台工作方式

我们的驱动程序为用户提供了后台工作方式进行数据传输，这样可以保证您的前台应用程序能实时高效的进行数据处理。后台方式的特点是在进行数据采集和传输过程中不占用客户程序的任何时间，当采集的数据长度达到客户指定的值时便触发客户事件，客户程序接受该事件便开始进行数据处理。在数据处理的同时，驱动程序依然在进行下一批数据的传输，即实现了并行操作，极大的提高了数据的吞吐量和计算机系统的整体处理能力。

五、与设备无关性

通过总结各数据采集卡的的共同特点，设计了基本一致的接口方式，可以让您的应用程序不仅能适应您所购买的我公司第一种产品，同时也能不经修改地适应我公司的其他同类产品（只有极少数设备需要极少的修改，其修改的比例基本不超过 5%）。所以可以保证您的应用程序在我们的硬件产品基础上极为容易地进行功能和应用扩展，节省您的大部分软件投资，极大的缩短工程开发周期。

六、驱动程序的坚固性

我们的驱动程序都是经过严密彻底的测试和验证，并经部分用户试用之后，确认没有任何问题后才予以正式发行的，所以当您使用起来应该有十足的安全感。

七、驱动程序特点

由于我们的驱动程序均采用动态虚拟技术（Windows 95），微内核代码（Windows NT）因此可动态装载和卸载，而且可以重入，即可实现多道任务同时访问硬件设备的功能。这样可以保证您的软硬件资源可以被充分有效的利用。特别是在 Windows NT 下，采用队列突发机制，可以实现几十道线程序同时访问一设备的功能。

八、高效与灵活兼备

如果您只是应用系统的上层用户，您多半不愿意了解硬件设备的各种复杂控制和操作协议，而只需要设置好您最关心的硬件参数（比如采集的 AD 通道、采样频率等），然后用一个读数的函数，跟上 AD 数据缓冲区和请求采集的数据长度，一执行程序便可以得到外部的数据，试想，这是不是一种最高效、最简单易用的方式。这种方式我们优先提供。比如下面将介绍的接口函数：InitDeviceProAD、ReadDeviceProAD 等两三个函数便可以帮助上层用户实现数据采集；但如果您是较为底层的用户，对硬件设备很熟悉，且有更特殊的编程和控制模式，那么您可能需要对硬件进行直接编程，就象传统的 ISA 总线设备，您想用 C 语言的 outp、inp，汇编语言的 out、in 的命令访问设备,那么我们为您提供了这类似的方法，只是函数名不一样，如写端口函数名为：WritePortWord(16 位方式),读端口函数名为 ReadPortWord（16 位方式）,但是功能更强，不仅有 32 位的，更有 16 位的，8 位的读写函数。它能访问硬件说明书里提供的任何一个寄存器。当然，随着设备与驱动程序的升级，我们还会提供 64 位的端口读写接口。

九、安装程序特点

关于驱动程序的安装方式我们采用大多数 Windows 应用程序所使用的标准模式，因而简捷、方便、直观。您只需执行安装盘上的 Setup.exe 启动文件即可进行驱动程序的安装工作。在安装过程中您设置好安装目标路径以及文件夹名称等信息后，安装程序便自动而又快捷地为您安装好驱动程序，随后您便可以用驱动程序接口编写应用程序或用我们提供的简易测试程序测试设备了。

十、多语言编程环境

本系统提供 Visual C++、C++ Builder、 Visual Basic、Delphi、 LabView、LabWindows/CVI 的函数接口，使您完全可以根据自己的需要和喜爱选择合适的编程语言。请记住，您得使用 32 位编程模式。但对于 LabWindows/CVI 接口，属于定制服务，如果用户需要，请另与我公司或指定代理商联系以便协商解决。

十一、为 Visual Basic 环境提供直接的多线程支持

在 VB 环境中进行各种实时控制和用户级后台操作，不用子线程,那简直是不可想象的事情。但是在通常情况下，要在 VB 环境中实现多线程操作并不象 VC 那么容易了。往往要相当复杂的对象操作，而且很不具有灵活性。但是有了我们的驱动程序支持，使这件事变得极为容易，甚至比 VC 还要容易。比如执行 CreateVBThread 函数,跟上 hThread 和 NewRoutine 两个参数，即可创建线程对象，并获得对象句柄，随后便可用 ResumeThread 函数启动子线程。在 VB 应用程序中,可以创建任意多个子线程。

十二、我公司动态库与其他公司动态库的比较

值得注意的是，我们的 DLL 库不同于其它许多公司所编写的那样，只是对动态库的简单直接地调用,其硬件控制、数据传输代码都放在 DLL 中，那么其代码的优先执行级别跟一般的用户程序是一样的，它总要定期地、不断地被系统级任务调度器调度，所以当这些代码在负责传输数据时往往被瞬时中断，有时这个时间还很长，故此，极有可能造成丢点的严重现象。且这种方案不可能提供硬件中断以及内存直接存取（**DMA**）方式来传输数据，这样难以满足用户的各种需求。为了解决这些问题，在 **Win95、Win98** 环境下，我们没有把硬件控制、数据传输代码简单地放在 DLL 中，而是通过动态虚拟技术以 **VxD** 的形式放在了 Windows 系统空间中，以 CPU 的 0 级环级别同系统代码协同工作，也就是说它可以获得与任务调用器一样的级别，且不受任务调度器的调度管理。在 **NT** 环境下，我们通过**微内核技术**把硬件控制、数据传输代码以微内核代码（简称**微代码**）形式放在 NT 的内核模式中，成为 NT 操作系统的一部分,并可根据代码的重要程度进一步迅速临时提升 CPU 的 IRQL 级别,使这些代码以高优先级,高速度工作，极大的提高了数据采集和传输的质量。而我们的 DLL 的主要任务不是采集数据，而是对驱动程序的全面封装，对用户负责简化所有复杂的繁琐的操作细节，特别是 Windows 底层管理，提供简洁一致的函数接口供用户使用。它具体表现在从用户空间到系统空间（Windows95,98）、从用户模式到内核模式(Windows NT)、从 CPU 的 3 级环到 0 级环(Windows)等相互间的转换以及设备 I/O 请求的来回传递。所以，我们的驱动程序不是 DLL，而是形如***.VxD**（Win95）或***.SYS**(NT)的代码文件。通过这样的技术便能实现设备所有功能，极大范围地满足用户需要。

十三、跨平台设计

至今，Windows95 与 Windows NT 是两大主流操作系统，它们各有其优点，但随着计算机的进一步网络化以及追求高可靠性和高稳定性，Windows NT 在不远的将来便会最终取代其它相关的 Windows 平台，成为更先进的视窗操作系统。为保护用户的软硬件投资，满足用户更长远的需要，我们不仅同时提供了基于 Windows95(98)和 Windows NT 操作平台的驱动程序，而且尽力做到了跨平台设计，使您的用户程序不用作怎么修改，一般只须在不同的平台重新编译、链接，便可运行在两种平台上。

十四、LabView/CVI 支持

LabView/CVI 是美国国家仪器公司(National Instrument)的虚拟仪器开发平台，特别是基于图形化编程的 LabView 语言，在测量、工控、虚拟仪器方面受到广大工程师和用户的青睐。其全球销售量仅次于 C++语言。我们自主开发的硬件（PCI、USB、ISA 总线系列）产品提供了基于 LabView 的驱动软件接口模块，与 LabView 软件平台完全兼容，让您轻松实现图形化编程。

十五、对传统总线设备的支持

由于某些用户可能除了使用现在流行的高级设备，如 PCI、USB 等总线的设备，但同时还要沿用以前购置的老设备（如基于 ISA 总线、并口、串口等设备）这些设备只能使用 I/O 地址，而不象 PCI 设备那样能使用内存映射地址。特别是对这些设备的访问，只能使用汇编语言的 in、out 指令和 C 语言的 outp、inp 等函数来实现读写。但是这些手段只能在 DOS、Win3.1、Win95 等操作系统中使用，如果您要在 Windows NT、Windows 2000、Windows XP 中使用，那是绝对不行的。因为这些指令已被操作系统作为权限控制，不允许用户在上层应用程序中使用这为了尽可能的保护您的前期投资，我们为您提供了能直接访问这些老设备的相应接口。如 WritePortByte、ReadPortByte 等。注意，为了提高速度，您应使用带后缀“Ex”函数访问设备，如 WritePortByteEx、ReadPortByteEx 等，因这些函数通过驱动程序底层的支持，突破了操作系统的某些限制，使它们能在用户直接访问 I/O 端口。

十六、自动卸载功能

在您已安装了本软件系统后，如果不再准备使用本系统，您可以通过我们为您提供的组件 unInstallShield 从 Windows 系统中自动卸载本软件系统。

十七、LabView/CVI 支持

如果您采用 Typical 安装选项，那么您一般可以得到我们为您提供的如下组件：

Hardware Help 硬件使用说明 Word 帮助文档；
ReadmeFile 安装目录等信息简介；
Setup 关于硬件参数设置的应用程序；
Software Help 软件使用说明 Word 帮助文档；
Test Application 基于 Microsoft Visual C++代码的硬件测试应用程序；
Visual C++ Sample Microsoft VC++演示程序（这个程序对驱动程序演示说 明最全面）；
Visual Basic Microsoft VB 演示及接口程序文件(ART2003.Bas)
C++ Builder Borland C++ Builder 演示程序；
Delphi Borland Delphi 演示及接口程序文件（ART2003.Pas）；
LabView 美国国家仪器公司(National Instrument)的虚拟仪器开发平台的演示程序及接口模块程序
UnInstallShield 本软件卸载应用程序；

第三章 设备驱动程序安装

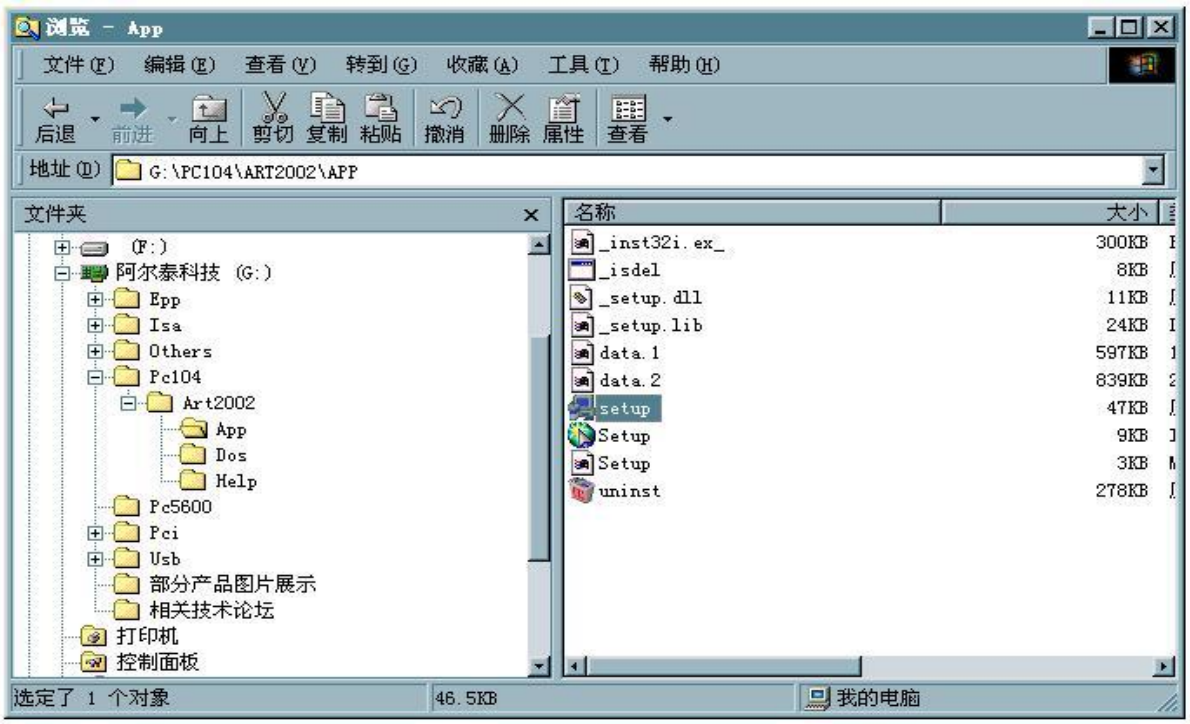
第一节 设备驱动程序及演示程序安装(以 ART2002 为例)

一、安装步骤

第一步 将设备按硬件要求插入计算机主板上的任意一个 ISA 插槽中，并将其固定好，连接好其外接设备后，打开计算机电源，启动 Windows95/98/Me 系统。

第二步 启动 Windows95/98/Me 后，将安装光盘放入光驱中，稍等几秒钟后，会自动弹出安装向导窗体，如下图，然后用户点击[驱动安装]，即可进行安装。

或者在资源管理器中，将当前位置定位到安装光盘的 PC104\ART2003\App 目录下，如下图：



双击 Setup.Exe 文件，即可安装。在 Win95/98/Me 环境下，安装后可直接进行设备的测试和其他正常工作，但在 Windows NT/2000/XP 环境下则需要重新复位计算机系统方可。

第二节 设备软件测试系统的介绍

1.怎样进入测试系统：当您正确完成了第四节中的工作，您便可以在 Windows 的系统菜单中找到“阿尔泰测控演

示系统”项，在其中即可找到相应设备的子菜单，最后点击其中的 VC 高级测试程序，即可进入测试系统。

2.怎样进行数据采集：在这个系统中，您可以先设置好各项硬件参数，然后再按[开始数据采集]按钮，即可开始采集 AD 数据。

3.怎样进行数据处理：在采集过程中，您可以随时在右下方的“数据处理方式”组合框中选择数据处理方式，或数字方式显示，或图形方式显示、数据存盘处理。在数字窗口中，单击鼠标右键，即可弹出浮动菜单，以实现电压显示等功能。

4.怎样进行精度分析：当您点击了“图形方式显示”单选框，便可进行精度分析。具体操作是：您必须先将采样通道设置为单通道方式，然后往此通道上加一个恒定的电压或接地，然后再在窗口右侧将屏幕量程逐渐改小，您便会发现图形窗口中的直线波形会由小变大，如果波形跳出窗口以外，您可以按动“F5”键，即可将波形平移到屏幕中央（如果再按一次“F5”键，又将波形恢复到本来的位置）。通过对宽大的波形上下分层情况，会可以确定设备的 AD 转换精度。当您平移波形后，在窗口右侧平移电压框中出现的值便是外加在这个通道上的电压值。如果您想对大批量数据进行全面精度统计，您可以在“触发电平”框中设置电压范围，此时您通过“分析总数”，“超限次数”、“超限比率”等框中的值了解 AD 精度。

5.怎样进行高速连续数据存盘处理：当您在“文件操作”菜单中选择“新建数据文件”后，数据存盘处理方式即刻有效，然后您再启动数据采集，即可进行数据连续高速大容量不间断存盘。

6.怎样进行存盘后的文件数据回放：当您在“文件操作”菜单中选择“打开数据文件”，即可对您指定的数据文件进行回放。此时会出现数据回放窗口。在这个窗口中单击“开始回放”按钮，即可开始自动回放。不管在什么情况下，您移动显示窗口滚动条或文件偏移滚动条即可进行定点搜寻。

7.怎样进行开关量测试,最好的办法是将板上的开关量输出端与开关量输入端一一对接起来，然后用户点击左边的按钮进行输出，则右边对应的输入则将发生相应变化，这种方法将开关量的输入和开关量输出同时进行测试，且也是一种最可靠的一种测试方法。

第三节 本驱动程序软件的关键文件

（WinDir 指 Windows 的系统根目录, UserDir 为本驱动软件的用户安装根目录）

文件名	文件类型及功能	适用的操作系统	文件位置
ART2003.VxD	动态虚拟设备驱动程序库	Window95/98	WinDir\System
ART2003.Sys	Win32 标准设备驱动 WDM 模式的设备驱动程序库	Windows NT/2000	WinDir\System32\Drivers
ART2003.Dll	底层驱动程序库的用户级函数接口封装所用的动态库。	所有操作系统	WinDir\System
ART2003.Lib	基于 Microsoft Visual C++工程开发环境的驱动程序函数接口输入库。	所有操作系统	UserDir\Include 或 UserDir\Samples\VC…
ART2003.Lib	基于 Borland C++ Builder 工程开发环境的驱动程序函数接口输入库。	所有操作系统	UserDir\Samples\C_Builder
ART2003.Bas	基于 Microsoft Visual Basic 工程开发环境的驱动程序函数接口输入模块文件	所有操作系统	UserDir\Samples\VB
ART2003.Pas	基于 Borland Delphi 工程开发环境的驱动程序函数接口输入单元文件。	所有操作系统	UserDir\Samples\Delphi
ART2003.VI	基于 National Instrument LabView 工程开发环境的驱动程序函数接口输入部件文件。**(只是外挂驱动接口)**	所有操作系统	UserDir\Samples\LabView

第四章 设备操作函数接口介绍

第一节 设备驱动接口函数列表（每个函数省略了前缀“ART2003_”）

函数名	函数功能	备注
设备对象操作函数		
CreateDevice	创建设备对象	上层及底层用户
ReleaseDevice	关闭设备，且释放总线设备对象	上层及底层用户
DA 操作函数		

WriteDeviceProDA	DA 输出函数	上层用户

使用需知：

Visual C++ & C++Builder：

要使用如下函数关键的问题是：

首先，必须在您的源程序中包含如下语句：

#include “C:\Art\ART2003\INCLUDE\ART2003.H”

注：以上语句采用默认路径和默认板号，应根据您的板号和安装情况确定 ART2003.H 文件的正确路径，当然也可以把此文件拷到您的源程序目录中。

其次，您还应该在 Visual C++编译环境软件包的 Project Setting 对话框的 Link 属性页中的 Object/Library Module 输入行中加入指令 C:\Art\ART2003\ART2003.LIB

或者：单击 Visual C++编译环境软件包的 Project 菜单中的 Add To Project 的菜单项，在此项中再单击 Files…，在随后弹出的对话框中选择 ART2003.Lib，再单击“确定”，即可完成。

注：以上语句采用默认路径和默认板号，应根据您的板号和安装情况确定 ART2003.LIB 的路径，当然也可以把此文件拷到您的源程序目录中。

另外，在 Visual C++演示工程的目录下，也有相应的 ART2003.h 和 ART2003.Lib 文件。

为了驱动程序和相关接口尽量精炼快速，所以没有加任何调试代码，因此用户在使用 VC 接口的时候应使用发行版本进行源代码编译（Win32 Release），而不应该使用调试版本（Win32 Debug）。具体方法是在源代码编译前，执行 Build 总菜单中的 Set Active Configuration 子菜单命令，便可实现其发行版的设置，然后再编译，即可生成发行版的应用程序。

另外，要在 VB 环境中用子线程以实现高速、连续数据采集与存盘，请务必使用 VB5.0 版本。当然如果您有 VB6.0 的最新版，也可以实现子线程操作。

C++ Builder:

要使用如下函数一个关键的问题是首先必须将我们提供的头文件(ART2003.H)写进您的源程序头部。如：#include “\Art\ART2003\Include\ART2003.h”

然后再将 ART2003.Lib 库文件分别加入到您的 C++ Builder 工程中。其具体办法是选择 C++ Builder 集成开发环境中的工程(Project)菜单中的“添加”（Add to Project）命令，在弹出的对话框中分别选择文件类型：Library file (*.lib)，即可选择 ART2003.Lib 文件。该文件的路径为用户安装驱动程序后其子目录 Samples\C_Builder 下

Visual Basic:

要使用如下函数一个关键的问题是首先必须将我们提供的模块文件(*.Bas)加入到您的 VB 工程中。其方法是选择 VB 编程环境中的工程(Project)菜单,执行其中的"添加模块"(Add Module)命令,在弹出的对话中选择 ART2003.Bas 模块文件，该文件的路径为用户安装驱动程序后其子目录 Samples\VB 下面。

请注意,因考虑 Visual C++和 Visual Basic 两种语言的兼容问题,在下列函数说明和示范程序中,所举的 Visual Basic 程序均是需要编译后在独立环境中运行。所以用户若在解释环境中运行这些代码，我们不能保证完全顺利运行。

Delphi:

要使用如下函数一个关键的问题是首先必须将我们提供的单元模块文件（*.Pas）加入到您的 Delphi 工程中。其方法是选择 Delphi 编程环境中的 View 菜单,执行其中的"Project Manager"命令,在弹出的对话中选择*.exe 项目，再单击鼠标右键，最后 Add 指令，即可将 ART2003.Pas 单元模块文件加入到工程中。或者在 Delphi 的编程环境中的 Project 菜单中，执行 Add To Project 命令，然后选择*.Pas 文件类型也能实现单元模块文件的添加。该文件的路径为用户安装驱动程序后其子目录 Samples\Delphi 下面。最后请在使用驱动程序接口的源程序文件中的头部的 Uses 关键字后面的项目中加入：“ART2003”。如：

```
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ART2003;  // 注意： 在此加入驱动程序接口单元 ART2003
```

LabView/CVI :

LabVIEW 是美国国家仪器公司(National Instrument)推出的一种基于图形开发、调试和运行程序的集成化环境，是目前国际上唯一的编译型的图形化编程语言。在以 PC 机为基础的测量和工控软件中，LabVIEW 的市场普及率仅次于 C++/C 语言。LabVIEW 开发环境具有一系列优点，从其流程图式的编程、不需预先编译就存在的语法检查、调试过程使用的数据探针，到其丰富的函数功能、数值分析、信号处理和设备驱动等功能，都令人称道。关于 LabView/CVI 的进一步介绍请见本文最后一部分关于 LabView 的专述。其驱动程序接口单元模块的使用方法如下：

一、在 LabView 中打开 ART2003.VI 文件,用鼠标单击接口单元图标,比如 CreateDevice 图标 

然后按 Ctrl+C 或选择 LabView 菜单 Edit 中的 Copy 命令，接着进入用户的应用程序 LabView 中，按

Ctrl+V 或选择 LabView 菜单 Edit 中的 Paste 命令，即可将接口单元加入到用户工程中，然后按以下函数原型说明或演示程序的说明连续该接口模块即可顺利使用。

二、根据 LabView 语言本身的规定，接口单元图标以黑色的较粗的中竖线为中心，以左边的方格为数据输入端，右边的方格为数据的输出端，如 ReadDeviceProAD 接口单元，左边为设备对象句柄、用户分配的数据缓冲区、要求采集的数据长度等信息从接口单元左边输入端进入单元，待单元接口被执行后，需要返回给用户的数据从接口单元右边的输出端输出，其他接口完全同理。

三、在单元接口图标中，凡标有“I32”为有符号长整型 32 位数据类型，“U16”为无符号短整型 16 位数据类型，“[U16]”为无符号 16 位短整型数组或缓冲区或指针，“[U32]”与“[U16]”同理，只是位数不一样。

第二节、设备对象管理函数原型说明

υ 创建设备对象函数

Visual C++ & C++Builder：

HANDLE CreateDevice (WORD BaseAddress)

Visual Basic：

Declare Function CreateDevice Lib “ART2003”(ByVal BaseAddress As Integer) As long

Delphi：

Function CreateDevice(BaseAddress : Word):Integer;
StdCall; External 'ART2003' Name ' CreateDevice';

LabView：

功能：该函数负责创建设备对象，并返回其设备对象句柄。

参数：

BaseAddress 象设备基地址，取值范围为 100H 至 3F0H，要求其值必须为 10H 的整数倍。

返回值：如果执行成功，则返回设备对象句柄；如果没有成功，则返回错误码 INVALID_HANDLE_VALUE。由于此函数已带容错处理，即若出错，它会自动弹出一个对话框告诉您出错的原因。您只需要对此函数的返回值作一个条件处理即可，别的任何事情您都不必做。

相关函数：ReleaseDevice

υ 释放设备对象所占的系统资源及设备对象

函数原型：

Visual C++ & C++Builder:

BOOL ReleaseDevice(HANDLE hDevice)

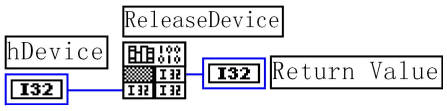
Visual Basic:

Declare Function ReleaseDevice Lib "ART2003" (ByVal hDevice As Long) As Boolean

Delphi：

Function ReleaseDevice(hDevice : Integer):Boolean;
StdCall; External 'ART2003' Name ' ReleaseDevice';

LabView：



功能：释放设备对象所占用的系统资源及设备对象自身。

参数：hDevice 设备对象句柄，它应由 CreateDevice 创建。

返回值：若成功，则返回 TRUE，否则返回 FALSE，用户可以用 GetLastError 捕获错误码。

相关函数：CreateDevice

应注意的是，CreateDevice 必须和 ReleaseDevice 函数一一对应，即当您执行了一次 CreateDevice 后，再一次执行这些函数前，必须执行一次 ReleaseDevice 函数，以释放由 CreateDevice 占用的系统软硬件资源，如 DMA 控制器，系统内存等。只有这样，当您再次调用 CreateDevice 函数时，那些软硬件资源才可被再次使用。

第二节、DA 操作函数原型说明

υ输出 DA 数据

函数原型:

Visual C++ & C++Builder:

BOOL WriteDeviceProDA (HANDLE hDevice,

```
                                WORD nDAData,
                                int nDAChannel)
```

Visual Basic:

```
Declare Function WriteDeviceProDA Lib "PCI2301" (
                                    ByVal hDevice As Long, _
                                    ByVal nDAData As Integer,_
                                    nDAChannel As Long) As Boolean
```

Delphi:

```
Function WriteDeviceProDA (hDevice : Integer;
                           nDAData: SmallInt;
                           nDAChannel: LongInt ):Boolean;
        StdCall; External 'ART2003' Name ' WriteDeviceProDA ';
```

LabView：

功能：向指定通道上输出一个点的 DA 数据

参数：

hDevice 设备对象句柄,它应由 CreateDevice 创建。

nDAData 准备输出的 DA 原始数据，注意它的换算关系请参考数据转换章节。

nDAChannel DA 通道，取值范围为 0-3， 若等于 0xFFFF，则本函数用同样的值 nDAData 同时输出到所有通道上。

返回值：如果成功，返回 TRUE，否则返回 FALSE，用户可用 GetLastError 捕获当前错误码,并加以分析。

相关函数： CreateDevice_ ReleaseDevice

第五章 共用函数介绍

这部分函数不参与本设备的实际操作，它只是为您编写数据采集与处理程序时的有力手段，使您编写应用程序更容易，使您的应用程序更高效。

第一节 公用接口函数列表

函数名	函数功能	备注
① 创建 Visual Basic 子线程，线程数量可达 32 个以上		
CreateVBThread	在 VB 环境中建立子线程对象	在 VB 中可实现多线程
TerminateVBThread	终止 VB 的子线程	
CreateSystemEvent	创建系统内核事件对象	用于线程同步或中断
ReleaseSystemEvent	释放系统内核事件对象	
② 文件对象操作函数		
CreateFileObject	初始设备文件对象	
WriteFile	请求文件对象写用户数据到磁盘文件	
ReadFile	请求文件对象读数据到用户空间	
SetFileOffset	设置文件指针偏移	
GetFileLength	取得文件长度	
ReleaseFile	释放已有的文件对象	
③ ISA 总线 I/O 端口操作函数		
WritePortByte	以字节(8Bit)方式写 I/O 端口	用户程序操作端口
WritePortWord	以字(16Bit)方式写 I/O 端口	用户程序操作端口
WritePortULong	以无符号双字(32Bit)方式写 I/O 端口	用户程序操作端口
ReadPortByte	以字节(8Bit)方式读 I/O 端口	用户程序操作端口
ReadPortWord	以字(16Bit)方式读 I/O 端口	用户程序操作端口
ReadPortULong	以无符号双字(32Bit)方式读 I/O 端口	用户程序操作端口
WritePortByteEx	以字节(8Bit)方式写 I/O 端口	NT 下直接操作端口
WritePortWordEx	以字(16Bit)方式写 I/O 端口	NT 下直接操作端口
WritePortULongEx	以无符号双字(32Bit)方式写 I/O 端口	NT 下直接操作端口
ReadPortByteEx	以字节(8Bit)方式读 I/O 端口	NT 下直接操作端口
ReadPortWordEx	以字(16Bit)方式读 I/O 端口	NT 下直接操作端口

ReadPortULongEx	以无符号双字(32Bit)方式读 I/O 端口	NT 下直接操作端口
④ 其他函数		
GetDiskFreeBytes	取得指定磁盘的可用空间(字节)	适用于所有设备

第二节 公用接口函数原型说明

一、创建 VB 子线程（如果您的 VB6.0 中线程无法正常运行，可能是 VB6.0 语言本身的问题，请选用 VB5.0）

υ 在 VB 环境中,创建子线程对象,以实现多线程操作

Visual Basic

```
Declare Function CreateVBThread Lib "ART2003" (hThread As Long, _
                                               ByVal RoutineAddr As Long _
                                              ) As Boolean
```

功能：该函数在 VB 环境中解决了不能实现或不能很好地实现多线程的问题.通过该函数用户可以很轻松地实现多线程操作。

参数：

hThread 若成功创建子线程，该参数将返回所创建的子线程的句柄，当用户操作这个子线程时将用到这个句柄，如启动线程,暂停线程,以及删除线程等。

RoutineAddr 作为子线程运行的函数的地址，在实际使用时,请用 AddressOf 关键字取得该子线程函数的地址,再传递给 CreateVBThread 函数。

返回值：当成功创建子线程时,返回 TRUE，且所创建的子线程为挂起状态,用户需要用 ResumeThread 函数启动它.若失败,则返回 FALSE，用户可用 GetLastError 捕获当前错误码。

相关函数： CreateVBThread
TerminateVBThread

注意: RoutineAddr 指向的函数或过程必须放在 VB 的模块文件中,如 ART2003.Bas 文件中.

Visual Basic 程序举例:

```
' 在模块文件中定义子线程函数(注意参考实例)
Function NewRoutine() As Long        ' 定义子线程函数
        :                            ' 线程运行代码
       NewRoutine = 1   ' 返回成功码
End Function
'
' 在窗体文件中创建子线程
        :
  Dim hNewThread As Long
If Not CreateVBThread(hNewThread, AddressOf NewRoutine) Then ' 创建新子线程
    MsgBox "创建子线程失败"
    Exit Sub
End If
ResumeThread (hNewThread)   '启动新线程              :
        :
```

υ 在 VB 中,删除子线程对象

Visual Basic:

```
Declare Function TerminateVBThread Lib "ART2003" (ByVal hThread As Long) As Boolean
```

功能：在 VB 中删除由 CreateVBThread 创建的子线程对象.

参数：hThread 指向需要删除的子线程对象的句柄，它应由 CreateVBThread 创建。

返回值：当成功删除子线程对象时,返回 TRUE.,否则返回 FALSE,用户可用 GetLastError 捕获当前错误码。

相关函数： CreateVBThread
TerminateVBThread

Visual Basic 程序举例:

```
        :
If Not TerminateVBThread (hNewThread)   ' 终止子线程
   MsgBox "创建子线程失败"
   Exit Sub
End If
        :
```

三、 创建内核系统事件

Visual C++:

```
HANDLE CreateSystemEvent(void);
```

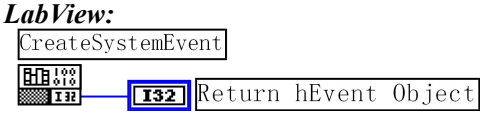
Visual Basic:

```
Declare Function CreateSystemEvent Lib " ART2003 " () As Long
```

Delphi:

Function CreateSystemEvent():Integer; StdCall; External 'ART2003' Name 'CreateSystemEvent';

LabView:



功能：创建系统内核事件对象,它将被用于中断事件响应或数据采集线程同步事件。

参数：无任何参数

返回值：若成功，返回系统内核事件对象句柄，否则返回－1(或 INVALID_HANDLE_VALUE)。

四、 释放内核系统事件

Visual C++:

BOOL ReleaseSystemEvent(HANDLE hEvent);

Visual Basic:

Declare Function ReleaseSystemEvent Lib " ART2003 " (ByVal hEvent As Long) As Boolean

Delphi:

Function ReleaseSystemEvent(hEvent : LongInt):Integer; StdCall; External 'ART2003' Name ' ReleaseSystemEvent ';

LabView:

功能：释放系统内核事件对象。

参数：hEvent 被释放的内核事件对象。它应由 CreateSystemEvent 成功创建的对象。

返回值：若成功，则返回 TRUE。

五、文件对象操作函数

υ 初始化设备文件对象

函数原型：

Visual C++:

```
Handle CreateFileObject (
                        HANDLE hDevice,
                        LPCTSTR NewFileName,
                        int Mode)
```

Visual Basic:

```
Declare Function CreateFileObjectLib "ART2003" (ByVal hDevice As Long, _
                                                ByVal NewFileName As String, _
                                                ByVal Mode As Long
                                              ) As Long
```

Delphi:

```
Function CreateFileObject (hDevice : Integer; NewFileName: string; Mode: Integer):LongInt;
        stdcall; external 'ART2003' name 'CreateFileObject';
```

LabView:

功能：初始化设备文件对象, 以期待 WriteFile 请求准备文件对象进行文件操作。

参数：

hDevice 设备对象句柄,它应由 CreateDevice 创建。

NewFileName 与新文件对象关联的磁盘文件名，可以包括盘符和路径等信息。在 C 语言中，其语法格式如：”C:\\ART2003\\Data.Dat”， 在 Basic 中，其语法格式如：”C:\ART2003\Data.Dat”

Mode 文件操作方式, 所用的文件操作方式控制字定义如下(可通过或指令实现多种方式并操作)

常量名	常量值	功能定义
ART2003_modeRead	0x0000	只读文件方式
ART2003_modeWrite	0x0001	只写文件方式
ART2003_modeReadWrite	0x0002	既读又写文件方式
ART2003_modeCreate	0x1000	如果文件不存可以创建该文件，否则重建此文件，并清 0
ART2003_typeText	0x4000	以文本方式操作文件

返回值：若成功，则返回文件对象句柄.

相关函数： CreateDevice CreateFileObject WriteFile ReadFile

ReleaseFile ReleaseDevice

υ 通过设备对象,往指定磁盘上写入用户空间的采样数据.

函数原型：

Visual C++:

```
BooL WriteFile( HANDLE hFileObject,
                PWORD pDataBuffer,
                 LONG nWriteSizeBytes)
```

Visual Basic:

```
Declare Function WriteFile Lib "ART2003" (ByRef hFileObject As Long,
                                          ByVal pDataBuffer As Integer,_
                                           ByVal nWriteSizeBytes As Long, _
                                          ) As Boolean
```

Delphi:

```
function WriteFile(hFileObject: Integer;
                   pDataBuffer:PWordArray;
                   nWriteSizeBytes: LongWord):Boolean;
                    stdcall; external 'ART2003' name 'WriteFile';
```

LabView:

功能：通过向设备对象发送“写磁盘消息”，设备对象便会以最快的速度完成写操作。注意为了保证写入的数据是可用的，这个操作将与用户程序保持同步，但与设备对象中的环形内存池操作保持异步，以得到更高的数据吞吐量，其文件名及路径应由 CreateFileObject 函数中的 NewFileName 指定。

参数：

hFileObject 设备对象句柄,它应由 CreateFileObject 创建。

pDataBuffer 用户数据空间地址。

nWriteSizeBytes 告诉设备对象往磁盘上一次写入数据的长度(以字节为单位)

返回值：若成功，则返回 TRUE，否则返回 FALSE，用户可以用 GetLastError 捕获错误码。

相关函数： CreateFileObject WriteFile

ReadFile ReleaseFile

υ 通过设备对象,从指定磁盘文件中读采样数据.

函数原型：

Visual C++:

```
BOOL ReadFile( HANDLE hFileObject,
               PVOID pDataBuffer,
               LONG OffsetBytes,
               LONG nReadSizeBytes)
```

Visual Basic:

```
Declare Function ReadFile Lib "ART2003" (ByVal hFileObject As Long, _
                                         ByRef pDataBuffer As Integer, _
                                         ByVal nOffsetBytes As Long, _
                                          ByVal nReadSizeBytes As Long, _
                                         ) As Boolean
```

Delphi:

```
Function ReadFile(hFileObject: Integer;
                  pDataBuffer:PWordArray;
                  nOffsetBytes:LongWord;
                  nReadSizeBytes:LongWord;):Boolean;
                  stdcall; external 'ART2003' name 'ReadFile';
```

LabView:

功能：通过向设备对象发送写磁盘消息，设备对象便会以最快的速度完成写操作。注意为了保证写入的数据是可用的，这个操作将与用户程序保持同步，但与设备对象中的环形内存池操作保持异步，以得到更高的数据吞吐量，其文件名及路径应由 CreateFileObject 函数中的 NewFileName 指定。

参数：

hFileObject 设备对象句柄,它应由 CreateFileObject 创建。

pDataBuffer 用于接受文件数据的用户缓冲区指针。

nOffsetBytes 指定从文件始端起所偏移的读位置。

nReadSizeBytes 告诉设备对象从磁盘上一次读入数据的长度(以字为单位)，其取值范围为[1, 65535]。
返回值：若成功，则返回 TRUE，否则返回 FALSE，用户可以用 GetLastError 捕获错误码。
相关函数： CreateFileObject WriteFile
ReadFile ReleaseFile

υ 设置文件偏移位置
函数原型：
Visual C++:
BOOL SetFileOffset (HANDLE hFileObject, int nOffsetBytes)
Visual Basic:
Declare Function SetFileOffset Lib "ART2003" (ByVal hFileObject As Long, ByVal nOffsetBytes As Long) As Boolean
Delphi:
Function SetFileOffset (hFileObject: Integer):Boolean;
stdcall; external 'ART2003' Name ' SetFileOffset ';
LabView:

功能：设置文件偏移位置，用它可以定位读写起点。
参数：hFileObject 文件对象句柄，它应由 CreateFileObject 创建。
返回值：若成功，则返回 TRUE，否则返回 FALSE，用户可以用 GetLastError 捕获错误码。
相关函数： CreateFileObject WriteFile
ReadFile ReleaseFile

υ 取得文件长度（字节）
函数原型：
Visual C++:
BOOL GetFileLength (HANDLE hFileObject)
Visual Basic:
Declare Function GetFileLength Lib "ART2003" (ByVal hFileObject As Long) As Boolean
Delphi:
function GetFileLength (hFileObject: Integer):Boolean;
stdcall; external 'ART2003' Name ' GetFileLength ';
LabView:

功能：取得文件长度。
参数：hFileObject 设备对象句柄，它应由 CreateFileObject 创建。
返回值：若成功，则返回 TRUE，否则返回 FALSE，用户可以用 GetLastError 捕获错误码。
相关函数： CreateFileObject WriteFile
ReadFile ReleaseFile

υ 释放设备文件对象
函数原型：
Visual C++:
BOOL ReleaseFile(HANDLE hFileObject)
Visual Basic:
Declare Function ReleaseFile Lib "ART2003" (ByVal hDevice As Long) As Boolean
Delphi:
Function ReleaseFile(hDevice : Integer):Boolean;
stdcall; external 'ART2003' Name 'ReleaseFile';
LabView:

功能：释放设备文件对象。
参数：hFileObject 设备对象句柄，它应由 CreateFileObject 创建。
返回值：若成功，则返回 TRUE，否则返回 FALSE，用户可以用 GetLastError 捕获错误码。
相关函数： CreateFileObject WriteFile
ReadFile ReleaseFile

四、I/O 端口读写函数
υ 以单字节(8Bit)方式写 I/O 端口
Visual C++ & C++ Builder:

BOOL WritePortByte (HANDLE hDevice, UINT nPort, BYTE Value)
BOOL WritePortByteEx (UINT nPort, BYTE Value)
(在 Windows NT/2000 用户模式程序中直接访问 I/O 端口)

Visual Basic:

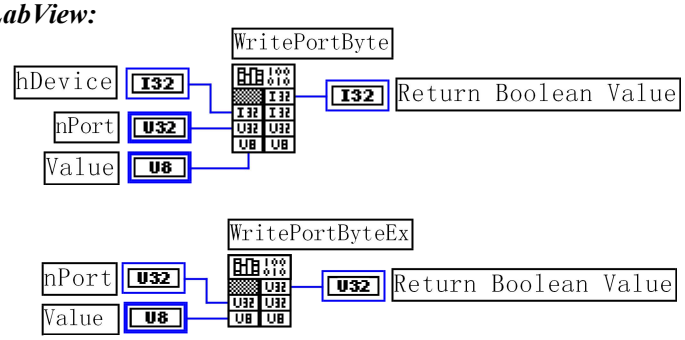
```
Declare Function WritePortByte Lib "ART2003" (ByVal hDevice As Long, _
                                              ByVal nPort As Long, _
                                              ByVal Value As Byte) As Boolean
Declare Function WritePortByteEx Lib "ART2003" (ByVal nPort As Long, _
                                              ByVal Value As Byte) As Boolean
```

(在 Windows NT/2000 用户模式程序中直接访问 I/O 端口)

Delphi:

```
Function WritePortByte(hDevice : Integer; nPort:LongWord; Value:Byte):Boolean;
        StdCall; External 'ART2003' Name 'WritePortByte';
Function WritePortByteEx(nPort:LongWord; Value:Byte):Boolean;
        StdCall; External 'ART2003' Name 'WritePortByte';
```

LabView:



功能：以单字节(8Bit)方式写 I/O 端口
参数：
hDevice 设备对象句柄,它应由 CreateDevice 或 CreateDeviceEx 创建。
nPort 设备的 I/O 端口号。
Value 写入由 nPort 指定端口的值。
返回值：若成功，返回 TRUE，否则返回 FALSE，用户可用 GetLastError 捕获当前错误码。

相关函数：

CreateDevice	WritePortByte
WritePortWord	WritePortULong
ReadPortByte	ReadPortWord
WritePortByteEx	WritePortWordEx
WritePortULongEx	ReadPortByteEx
ReadPortWordEx	ReadPortULongEx
ReadPortULongEx	ReleaseDevice

υ 以双字(16Bit)方式写 I/O 端口

Visual C++ & C++ Builder:

BOOL WritePortWord (HANDLE hDevice, UINT nPort, WORD Value)
BOOL WritePortWordEx (UINT nPort, WORD Value)
(在 Windows NT/2000 用户模式程序中直接访问 I/O 端口)

Visual Basic:

```
Declare Function WritePortWord Lib "ART2003" (ByVal hDevice As Long, _
                                              ByVal nPort As Long, _
                                              ByVal Value As Integer) As Boolean
Declare Function WritePortWordEx Lib "ART2003" (ByVal nPort As Long, _
                                              ByVal Value As Integer) As Boolean
```

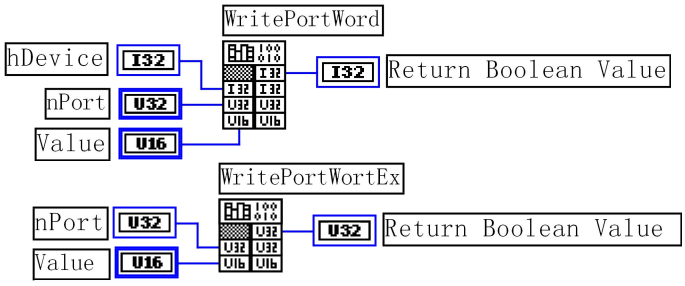
(在 Windows NT/2000 用户模式程序中直接访问 I/O 端口)

Delphi:

```
Function WritePortWord(hDevice : Integer; nPort:LongWord; Value:Word):Boolean;
        StdCall; External 'ART2003' Name 'WritePortWord';
Function WritePortWordEx(nPort:LongWord; Value:Word):Boolean;
```

StdCall; External 'ART2003' Name 'WritePortWord';

LabView:

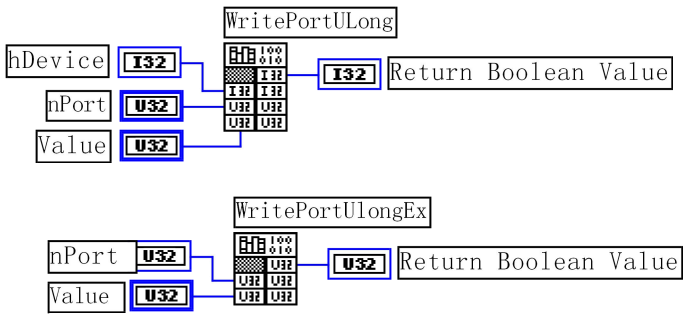


功能：以双字(16Bit)方式写 I/O 端口
参数：
hDevice 设备对象句柄,它应由 CreateDevice 创建。
nPort　设备的 I/O 端口号。
Value　写入由 nPort 指定端口的值。
返回值：若成功，返回 TRUE，否则返回 FALSE，用户可用 GetLastError 捕获当前错误码。
相关函数：

CreateDevice	WritePortByte
WritePortWord	WritePortULong
ReadPortByte	ReadPortWord
WritePortByteEx	WritePortWordEx
WritePortULongEx	ReadPortByteEx
ReadPortWordEx	ReadPortULongEx
ReadPortULongEx	ReleaseDevice

υ 以四字节(32Bit)方式写 I/O 端口
Visual C++ & C++ Builder:
BOOL WritePortULong(HANDLE hDevice, UINT nPort, ULONG Value)
BOOL WritePortULongEx(UINT nPort, ULONG Value)
(在 NT 用户模式程序中直接访问 I/O 端口)
Visual Basic:
```
Declare Function WritePortULong Lib "ART2003" (ByVal hDevice As Long, _
                                               ByVal nPort As Long, _
                                               ByVal Value As Long ) As Boolean
Declare Function WritePortULongEx Lib "ART2003"(ByVal nPort As Long, _
                                               ByVal Value As Long ) As Boolean
```
(在 Windows NT/2000 用户模式程序中直接访问 I/O 端口)
Delphi:
```
Function WritePortULong(hDevice : Integer; nPort:LongWord; Value:LongWord):Boolean;
        StdCall; External 'ART2003' Name 'WritePortULong';
Function WritePortULongEx(nPort:LongWord; Value:LongWord):Boolean;
        StdCall; External 'ART2003' Name 'WritePortULong';
```

LabView:



功能：以四字节(32Bit)方式写 I/O 端口
参数：

hDevice 设备对象句柄,它应由 CreateDevice 或 CreateDeviceEx 创建。
nPort 设备的 I/O 端口号。
Value 写入由 nPort 指定端口的值。
返回值：若成功，返回 TRUE，否则返回 FALSE，用户可用 GetLastError 捕获当前错误码。
相关函数： CreateDevice WritePortByte WritePortWord WritePortULong ReadPortByte ReadPortWord WritePortByteEx WritePortWordEx WritePortULongEx ReadPortByteEx ReadPortWordEx ReadPortULongEx ReleaseDevice

υ 以单字节(8Bit)方式读 I/O 端口

Visual C++ & C++ Builder:

```
BYTE ReadPortByte(HANDLE hDevice, UINT nPort)
BYTE ReadPortByteEx(UINT nPort)
(在 Windows NT/2000 用户模式程序中直接访问 I/O 端口)
```

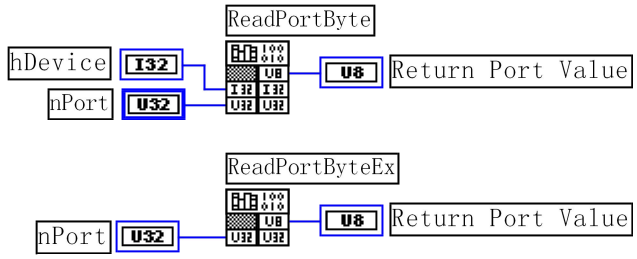
Visual Basic:

```
Declare Function ReadPortByte Lib "ART2003" (ByVal hDevice As Long, _
                                             ByVal nPort As Long ) As Byte
Declare Function ReadPortByteEx Lib "ART2003" ( ByVal nPort As Long ) As Byte
(在 Windows NT/2000 用户模式程序中直接访问 I/O 端口)
```

Delphi:

```
Function ReadPortByte(hDevice : Integer; nPort:LongWord):Byte;
        StdCall; External 'ART2003' Name 'ReadPortByte';
Function ReadPortByteEx(nPort:LongWord):Byte;
        StdCall; External 'ART2003' Name 'ReadPortByte';
```

LabView:



功能：以单字节(8Bit)方式读 I/O 端口
参数：
hDevice 设备对象句柄,它应由 CreateDevice 或 CreateDeviceEx 创建。
nPort 设备的 I/O 端口号。
返回值：返回由 nPort 指定的端口的值
相关函数： CreateDevice WritePortByte WritePortWord WritePortULong ReadPortByte ReadPortWord WritePortByteEx WritePortWordEx WritePortULongEx ReadPortByteEx ReadPortWordEx ReadPortULongEx ReleaseDevice

υ 以双字节(16Bit)方式读 I/O 端口

Visual C++ & C++ Builder:

```
WORD ReadPortWord(HANDLE hDevice, UINT nPort)
WORD ReadPortWordEx(UINT nPort)
(在 Windows NT/2000 用户模式程序中直接访问 I/O 端口)
```

Visual Basic:

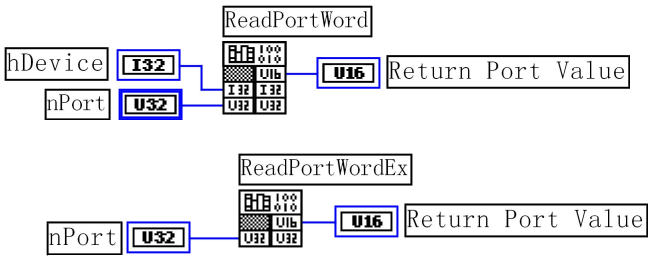
```
Declare Function ReadPortWord Lib "ART2003" (ByVal hDevice As Long, _
                                             ByVal nPort As Long ) As Integer
```

Declare Function ReadPortWordEx Lib "ART2003" (ByVal hDevice As Long, _
ByVal nPort As Long) As Integer
(在 Windows NT/2000 用户模式程序中直接访问 I/O 端口)

Delphi:

Function ReadPortWord(hDevice : Integer; nPort:LongWord):Word;
StdCall; External 'ART2003' Name 'ReadPortWord';
Function ReadPortWordEx(nPort:LongWord):Word;
StdCall; External 'ART2003' Name 'ReadPortWord';

LabView:



功能：以双字节(16Bit)方式读 I/O 端口
参数：
hDevice 设备对象句柄,它应由 CreateDevice 或 CreateDeviceEx 创建。
nPort 设备的 I/O 端口号。
返回值：返回由 nPort 指定的端口的值
相关函数：

CreateDevice	WritePortByte
WritePortWord	WritePortULong
ReadPortByte	ReadPortWord
WritePortByteEx	WritePortWordEx
WritePortULongEx	ReadPortByteEx
ReadPortWordEx	ReadPortULongEx
ReadPortULongEx	ReleaseDevice

υ 以四字节(32Bit)方式读 I/O 端口

Visual C++ & C++ Builder:

WORD ReadPortULong(HANDLE hDevice, UINT nPort)
WORD ReadPortULongEx(UINT nPort)
(在 Windows NT/2000 用户模式程序中直接访问 I/O 端口)

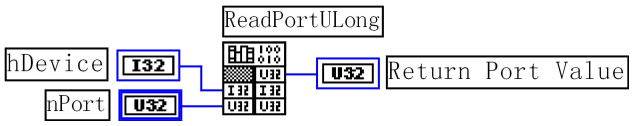
Visual Basic:

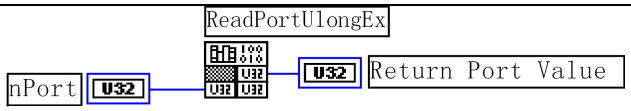
Declare Function ReadPortULong Lib "ART2003" (ByVal hDevice As Long, _
ByVal nPort As Long) As Long
Declare Function ReadPortULongEx Lib "ART2003" (ByVal nPort As Long) As Long
(在 Windows NT/2000 用户模式程序中直接访问 I/O 端口)

Delphi:

Function ReadPortULong(hDevice : Integer; nPort:LongWord):LongWord;
StdCall; External 'ART2003' Name 'ReadPortULong';
Function ReadPortULongEx(nPort:LongWord):LongWord;
StdCall; External 'ART2003' Name 'ReadPortULong';

LabView:





功能：以四字节(32Bit)方式读 I/O 端口
参数：
hDevice 设备对象句柄,它应由 CreateDevice 创建。
nPort　设备的 I/O 端口号。
返回值：返回由 nPort 指定端口的值
相关函数： CreateDevice　WritePortByte
WritePortWord　WritePortULong
ReadPortByte　ReadPortWord
WritePortByteEx　WritePortWordEx
WritePortULongEx　ReadPortByteEx
ReadPortWordEx　ReadPortULongEx
ReadPortULongEx　ReleaseDevice

第三节 其他函数

一、取得指定磁盘的可用空间

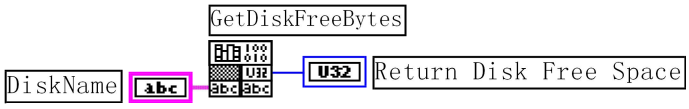
Visual C++:

ULONGLONG GetDiskFreeBytes（LPCTSTR DiskName)

Visual Basic:

Declare Function GetDiskFreeBytes Lib "ART2003" (ByVal DiskName As String) As Currency

LabView:



功能：取得指定磁盘的可用剩余空间(以字为单位)。
参数：需要访问的盘符,若为 C 盘为"C:\\", D 盘为"D:\\"，以此类推。
返回值：若成功，返回大于或等于 0 的长整型值, 否则返回零值，用户可用 GetLastError 捕获错误码。注意使用 64 位整型变量。

第七章　数据格式转换与排列规则

DA 的电压值如何转换成输出到 DA 转换器的 LSB 原码数据？

量程(伏)	计算机语言换算公式	Lsb 取值范围
±5000mV	Lsb = Volt / (10000 / 65536) + 32768	[0，65535]
0～10000mV	Lsb = Volt/ (10000 / 65536)	[0，65535]
±10000mV	Lsb = Volt/ (20000 / 65536) + 32768	[0，65535]
0～5000mV	Lsb = Volt/ (5000 / 65536)	[0，65535]

请注意这里求得的 LSB 数据就是用于 WriteDeviceProDA 中的 DAData 参数的。

附录 A　LabView/CVI 图形语言专述

第一章 图形化编程语言 LabVIEW 环境及其开放性

图形化编程语言 LabVIEW 是著名的虚拟仪器开发平台。LabVIEW 是美国国家仪器公司(National Instrument)推出的一种基于图形开发、调试和运行程序的集成化环境，是目前国际上唯一的编译型的图形化编程语言。在以 PC 机为基础的测量和工控软件中，LabVIEW 的市场普及率仅次于 C++/C 语言。LabVIEW 开发环境具有一系列优点，从其流程图式的编程、不需预先编译就存在的语法检查、调试过程使用的数据探针，到其丰富的函数功能、数值分析、信号处理和设备驱动等功能，都令人称道。本文对 LabVIEW 开发环境及其开放性作一简述。

第一节 LabVIEW 概述

LabVIEW 使用了一种称为 G 的数据流编程模式，它有别于基于文本语言的线性结构。在 LabVIEW 中执行程序的顺序是由块之间的数据流决定的，而不是传统文本语言的按命令行次序连续执行的方式。

LabVIEW 程序称为虚拟仪表(Vitual Instrument)程序，简称 VI。VI 包括 3 个部分：前面板、框图程序和图标/连接口。前面板用于输入数值和观察输出量。

输入量被称为 Controls，输出量被称为 Indicators。用户可以使用许多图标，如旋钮、开关、文本框和刻度盘等来使前面板易看易懂。如图 1 所示，它是一个温度计程序(Thermometer VI)的前面板。

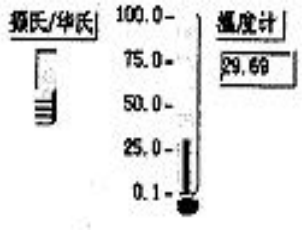


图 1　温度计的前面板

每一个前面板都伴有一个对应的框图(block diagram)程序。框图程序使用图形编程语言编写，可以把它理解成传统程序的源代码。框图中的程序可以看成程序节点，如循环控制、事件控制和算术功能等。这些部件用连线联接，以定义框图内的数据流动方向。上述温度计程序的框图程序如图 2 所示，框图程序的编写过程与人的思维过程非常接近。LabVIEW 提供的 3 类可移动的图形化工具模板用于创建和运行程序，它们是工具(Tools Palette)、控制(Controls Palette)和功能(Functions Palette)等。工具模板用于创建、修改和调试程序(如连线、着色等)；控制模板用来设计仪器的前面板(如增加输入控制量和输出指示量等)；功能模板用来创建相当于源代码的 LabVIEW 框图程序(如循环、数值运算、文件 I/O 等)。LabVIEW 平台的特点可归结为以下几个方面：

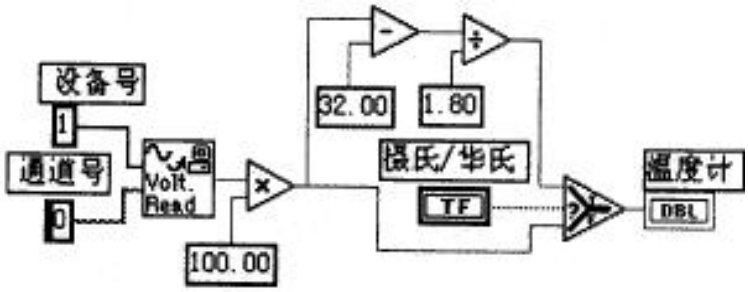


图 2　温度计的框图程序

(1)图形编程方式：使用直观形象的数据流程图式的语言书写程序源代码；

(2)提供程序调试功能，如设置断点或探针，单步执行，语法检查等；

(3)拥有数据采集、仪器控制、分析、网络、 ActiveX 等集成库；

(4)继承传统编程语言结构化和模块化的优点，这对于建立复杂应用和代码的可重用性来说是至关重要的；

(5)提供 DLL 库接口、CIN 节点以及大量的仪器驱动器、网络通信 VIs 与其它应用程序或外部设备进行连接；

(6)采用编译方式运行 32 位应用程序；

(7)支持多种系统平台，如 Macintosh、HP-UX、SUN SPARC 和 Windows 3.x/95/NT 等，LabVIEW 应用程序能在上述各平台之间跨平台进行移植；

(8)提供大量的函数库及附加工具。如数学函数、字串处理函数、数组运算函数、文件 I/O、高级数字信号处理函数、数据分析函数、仪器驱动和通信函数等。

第二节　程序设计结构

(1)层次化结构

LabVIEW 是模块化程序设计语言，用户可以把一个 VI 程序创建成自己的一个图标/接口(即 VI 子程序) ，然后被其它 VI 程序所调用。用这种方法可设计出一个有层次关系的 VIs 或子 VIs ，而且调用阶数是无限制的。

(2)并行工作

LabVIEW 是一个多任务的软件系统，当创建具有同步工作的程序块时，就可交互地运行并行 VIs 程序。

(3)常规语法结构：While Loops，For Loop，Case 结构，顺序结构等;

(4)基于文本的公式结(Formula Node)

公式结是一种用于书写数学公式的文本编辑框。

第三节　LabVIEW 的运算形式

(1)模块化图标运算

LabVIEW 中的图标/连接口表示一定的函数功能，将若干个图标/连接口组合起来就可进行有关运算，如算术、布尔逻辑、比较和数组运算、数值运算(三角函数、对数等)、字符串运算和文件 I/O 等；

(2)公式运算

使用公式结运行数学公式。公式结包含一个或多个公式表达式，各公式之间用分号"；"隔开。公式表达式使用了一种类似于大多数基于文本编程语言(如 BASIC 语言)的算术表达式的语法。如图 3 所示，输入变量为 m、b 和 x，经公式结运算后的输出变量为 y1 和 y2。公式结中使用的变量或公式的数量是无限制的。

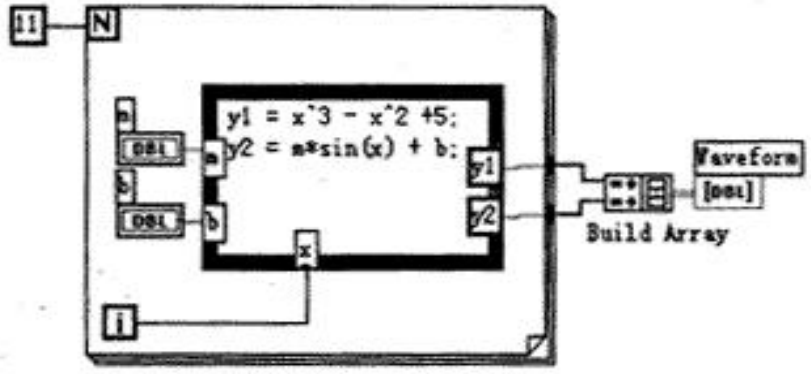


图 3　公式结运算例子

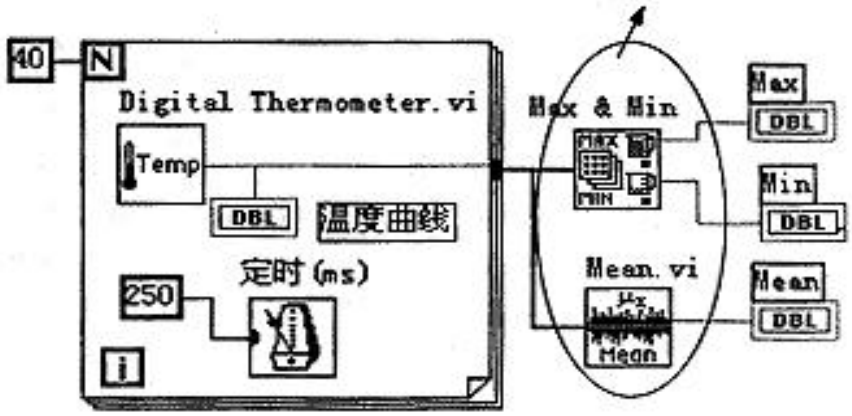


图 4　使用功能子模块进行温度曲线分析

(3)　使用集成库的功能子模板完成运算

LabVIEW 中集成了大量的生成图形界面的模板，丰富实用的数值分析、数字信号处理功能，以及多种硬件设备驱动器(包括 RS232、GPIB、VXI、DAQ 卡和网络等)。用户不需了解有关运算细节就能直接使用这些功能子模块，这对于编程工作来说，可节省了大量的时间开销。如图 4 所示，使用两个功能子模块进行温度曲线分析，以求出数组的最大值、最小值和平均值。

(4)通过链接 DLL 形式的代码进行运算

LabVIEW 提供 DLL 库接口和 CIN 节点来使用户有能力在该平台上使用其它软件开发平台生成的模块。即用户可通过其它开发平台(如 BC++)建立一个子例程，并生成动态链接库 DLL，然后与 LabVIEW 框图程序进行链接。LabVIEW 的这一开放性，为用户自行编写某些软件模块提供了方便。如用户可通过 C++/C 语言为某一新设备开发通信及驱动程序，或编写一控制算法软件，然后链入 LabVIEW 程序。

第四节　LabVIEW 的开放性

LabVIEW 是开放型的开发环境，它拥有大量的与其它应用程序进行通信的 VI 库。因此，LabVIEW 可从众多的外部设备获取或传送数据，这些设备包括 GPIB、VXI、PXI、串行设备、PLCs、和插件式 DAQ 板等；LabVIEW 甚至可以通过 Internet 取得外部数据源。

(1)DLLs

在 Windows 或其它平台下调用内部或外部的 DLL 形式的代码或分享其它平台(包括 Windows)中的库资源；使用 CodeLink，同样可自动分享在 LabWindows/CVI 中开发的 C 程序库；

(2)ActiveX，DDE,SQL

使用自动化 ActiveX、DDE 和 SQL，与其它 Windows 应用程序一起集成用户的应用程序；

(3)远程通信：Internet，TCP/IP

使用 TCP/IP 和 UDP 网络 VIs，与远程应用程序进行通信；在用户的应用程序中融入 e-mail、FTP 和浏览器等；通过远程自动控制 VIs，可远程操作其它机器上的分散 VIs 的执行。

第五节 调试工具

(1)语法检查：如果程序有错，则无需编译，工具栏的运行按钮就会出现一个折断的箭头。点击该箭头，就会给出错误列表信息。

(2)运行灯高亮：运行灯高亮用于在单步模式下跟踪框图程序中的数据流动。

(3)单步执行：按顺序一个节点一个节点地执行程序。

(4)探针：探针工具用来查看程序流经某一根连线上的数据。

(5)断点：设置断点可在程序的某一地方终止程序的执行，以观察调试部分的执行结果。

综上所述，列出 LabVIEW 的开发环境表，如表 1 所示。

第六节 工具软件包

NI 公司及其协作单位提供众多的软件工具箱和支持软件，用于扩展支持 LabVIEW。这些工具软件包有：

(1)常用工具箱

Application Builder:创建可单独运行的应用程序；

控件与指示器

按钮/开关 LED，滑块/数显，计量器/刻度盘/旋钮，水槽/温度表，曲线图/图表，表格/数组，密度图，菜单/列表/环，文本框；

仪器控制

GPIB，VXI，Serial，CAMAC，PLC 等 600 多种仪器驱动器；

文件 I/O 电子表格，二进制，ASCII 码，日志；

开放性联接

Internet，SQL，TCP/IP，Activex，DLLs，DDE 等；

数据采集

DAQ，单点输入/输出，波形采集/发生，图像采集，信号调理，触发/定时，TTL/CMOS 输入/输出，数字图案发生，数字握手，脉冲发生，事件计数，边界检测，周期和脉宽测量等；

程序设计结构

While Loops，For Loop，Case 结构，顺序结构，基于文本的公式结；

程序设计原则

算术运算，布尔逻辑，数组处理，串函数，时间/日期函数，多数据类型结构，用户子例程；

分析

信号发生，信号处理，图象处理，曲线拟合，窗体，过滤，线性，统计等；

优化与应用程序管理

用于存储管理和执行时间跟踪的 Profiler，在所有平台上的 TURE 编译性能，源代码控制，文档打印等；

调试

断点，探针，单步模式，执行高亮，帮助窗口，在线帮助 Test Suite:包括 600 多个仪器驱动程序软件包、连接到 30 多个本地或远程数据库的数据库连接工具、程序性能测试和分析软件等；

Test Executive:

多用途的附加软件包。使用该软件包，可以控制程序执行的次序，生成应用程序。按照自己的特定要求和标准来设置应用程序。在保持扩展升级兼容性的前提下，允许用户增强操作和人机接口；

SQL:用于与本地或远程数据库的直接访问；

SPC:过程控制中统计方法应用程序库；

Internet:把 VI 程序转换成可在 Internet 上执行的应用程序；

PID:给 LabVIEW 加入复杂的控制算法。该软件包带有许多误差反馈及外部复位的 PID 算法，同时含有超前-滞后补偿和设置点斜率生成等功能；

Picture Control: 一个多功能的图形软件包，用于生成前面板显示，如特殊的棒形图、饼形图和 Smith 图表等。

7 (2)分析工具箱

HIQ：一个交互式的工作环境，可以对数学、科学计算和工程问题的数据进行组织、可视化处理。HIQ 集成了数学运算用户接口控制、数值分析、矩阵运算及二维、三维和四维图形处理；

Signal Processing Suite: 提供数据处理功能和高级信号处理工具。如数字滤波器、1/3 倍频程分析和动态信号分析等；

G Math:算术运算、数据分析和数据可视化。如常微分方程、最优化、变换和过程控制模拟等；

Image Processing:提供图象处理功能和机器视觉功能等。

第七节 总结

LabVIEW 是开放型模块化程序设计语言，使用它可快速建立自己的仪器仪表系统，而又不用担心程序的质量和运行速度。LabVIEW 既适合编程经验丰富的用户使用，也适合编程经验不足的工程技术人员使用，所以被誉为工程师和科学家的语言。