LabVIEW™

LabVIEW 基础

2005 年 10 月
374029A-0118

NATIONAL INSTRUMENTS™

全球技术支持及产品信息

ni.com

National Instruments Corporate 总部

11500 North Mopac Expressway Austin, Texas 78759-3504 USA 电话 : 512 683 0100

全球办事处

澳大利亚 1800 300 800, 奥地利 43 0 662 45 79 90 0, 比利时 32 0 2 757 00 20,
巴西 55 11 3262 3599, 加拿大 800 433 3488, 中国 86 21 6555 7838, 捷克共和国 420 224 235 774,
丹麦 45 45 76 26 00, 芬兰 385 0 9 725 725 11, 法国 33 0 1 48 14 24 24, 德国 49 0 89 741 31 30,
印度 91 80 51190000, 以色列 972 0 3 6393737, 意大利 39 02 413091, 日本 81 3 5472 2970,
韩国 82 02 3451 3400, 黎巴嫩 961 0 1 33 28 28, 马来西亚 1800 887710, 墨西哥 01 800 010 0793,
荷兰 31 0 348 433 466, 新西兰 0800 553 322, 挪威 47 0 66 90 76 60, 波兰 48 22 3390150,
葡萄牙 351 210 311 210, 俄罗斯 7 095 783 68 51, 新加坡 1800 226 5886, 斯洛文尼
亚 386 3 425 4200, 南非 27 0 11 805 8197, 西班牙 34 91 640 0085, 瑞典 46 0 8 587 895 00,
瑞士 41 56 200 51 51, 台湾 866 02 2377 2222, 泰国 662 992 7519, 英国 44 0 1635 523545

如需更多关于技术支持的信息，请查阅 “技术支持及专业服务”附录。 如需对 National Instruments 文档提出任何意见或建议，请登录 National Instruments 网站 ni.com/info 并输入代码 feedback。

重要信息

保证书

发货日起 90 天内， National Instruments 保证其软件载体不会因材料或制作方面的问题导致无法执行编程指令。发货日以发票或其它有关证明文件为准。在此期间内，如 National Instruments 收到有关该问题的通知，将选择进行维修或更换无法执行编程指令的软件载体。 National Instruments 不保证软件的运行不中断或完全无误。

任何设备获取保证服务前，必须在外包装上明确标注有从厂家获取的商品返修授权 （RMA）编号。对于保证书担保的货物，National Instruments 将承担货物返还的运费。

National Instruments 确保本文件中信息的准确性。本文件已经严格审阅以确保其技术方面的准确性。如出现技术或印刷错误，National Instruments 保留对本文件后续版本的修改权，而毋须事先通知本版本的持有人。如发现错误，用户应垂询 National Instruments。 National Instruments 在任何情况下均无须对由本文件或本文件中信息所引起或与之相关的任何损害承担责任。

除本文另有明确规定， National Instruments 不作其它任何明示或暗示的保证并明确拒绝适销性或针对特定目的适用性的任何保证。因 National Instruments 的过错或疏忽而导致的赔偿应限于客户所支付的金额范围之内。即使已被告知相关可能性，National Instruments 也不对数据丢失、利润损失、使用产品导致的损害，偶然或间接损害承担责任。 National Instruments 的此项有限责任条款适用于任何形式的法律程序，无论是违反合同、侵权行为 （包括疏忽）或其它。任何针对 National Instruments 的诉讼必须在诉讼事由发生起一年内提起。National Instruments 对其有效控制外的原因引起的任何行事延误不承担责任。本文中规定的保证不包含由以下原因引起的损害、缺陷、故障或服务方面的问题：用户未能遵守 National Instruments 有关安装、操作或维护方面的指示；用户对产品进行修改；用户对产品的滥用、误用或疏忽行为、停电或功率骤增、火灾、洪灾、事故、第三方行为，或有效控制以外的其它事件。

版权

商标

专利权

关于 National Instruments 产品的专利权，见软件中 **Help»Patents**， CD 中 `patents.txt` 文档，或登录 `ni.com/patents`。

使用 NATIONAL INSTRUMENTS 产品注意事项

(1) 对某些外科移植手术设备或关键救生系统而言，运行故障可能导致严重的人身伤害。 National Instruments 产品设计中未涵盖适用于上述外科移植手术设备或任何关键救生系统的组件，也未经与此相关的可靠性测试。

(2) 在包括上述情况在内的任何实际应用中，软件产品运行的可靠性可能受到不利因素影响，包括但不限于以下因素：供电不稳定、计算机硬件故障、计算机操作系统与软件的兼容性、编码器与应用软件开发工具的兼容性、安装错误、软硬件兼容性问题、电子监控或控制设备故障或失灵、电子设备的短暂性故障 （硬件和 / 或软件）、意外使用或误用、用户或应用设计师操作失误 （这些不利因素以下统称 “系统故障”）。在任何应用中，如系统故障将可能对财产或人身安全造成伤害 （包括人身伤害和死亡），考虑到其可能存在的系统故障风险，不应仅依赖于某一种电子系统。为避免受损、伤害或死亡，用户或应用设计师必须采取合理谨慎的措施对系统故障采取保护措施，包括备份或关闭机制等。由于每套最终用户的系统均为定制并与 National Instruments 的测试平台有差异，且由于用户或应用设计师可能将 National Instruments 产品与其它产品一起使用，而 National Instruments 之前未对此进行测试或预计，因此当 National Instruments 产品与其它系统或程序共同使用时，用户或应用设计师应对测试和验证 National Instruments 产品的适用性承担最终责任，包括但不限于该系统和程序的合理设计、流程和安全等级。

目录

第 4 章
创建前面板

第 5 章
创建程序框图

第 6 章
运行和调试 VI

第 7 章
创建 VI 和子 VI

第 8 章
循环和结构

关于本用户手册

在阅读本手册之前，请先参考 *《LabVIEW 入门》(Getting Started with LabVIEW)*，这将有助于您初步了解 LabVIEW 图形化编程环境，掌握 LabVIEW 中创建数据采集和仪器控制程序的各项功能。

本文档包括 LabVIEW 的编程理论、技巧和功能，介绍了用于创建测试测量、数据采集、仪器控制、数据记录、测量分析和报表生成等各类应用程序的 VI 和函数。

LabVIEW Help 中包含本手册的全部内容。阅读本手册时如需查找某个内容更为详细的介绍，请参阅 *LabVIEW Help*。

本手册不会对每个选板、工具、菜单、对话框、输入控件、显示控件、内置 VI 和函数作详尽的描述。如需详细内容，或需了解 LabVIEW 各项功能的分步操作指导以及如何创建具体应用程序，请参阅 *LabVIEW Help*。关于 *LabVIEW Help* 的更多信息见第 1 章 *LabVIEW 简介*中的 *LabVIEW 文档资源 (LabVIEW Documentation Resources)* 一节。

行文规范

本手册中的行文规范：

»	» 表示通过嵌套菜单和对话框选项进行选择。**File»Page Setup»Options** 顺序，表示先下拉 **File** 菜单，再选择 **Page Setup**，然后从对话框中选择 **Options**。
	该提示符号提醒您注意参考信息。
	该提示符号提醒您注意重要信息。
	该警告符号表示提醒采取预防措施以防受伤、避免数据丢失或系统崩溃。
粗体	粗体文本表示软件中的必选项，如菜单和对话框选项。粗体文本也表示参数名称、前面板上的输入控件和显示控件、对话框、对话框的一部分、菜单名称和选板名称。
斜体	斜体表示变量、强调、交叉引用或重要概念介绍。同时它也可以为占位符，表示须由用户填写的文字或数值。
等宽字体	等宽字体表示用户必须从键盘输入的文字、部分代码、程序范例和语法范例。该字体也用于对磁盘驱动器名称、路径、目录、程序、子程序、设备名、运算、变量、文件名和扩展名的命名。

等宽粗体	等宽粗体表示由计算机在屏幕上自动生成的消息和响应。同时用于强调不同于其他范例的代码行。
等宽斜体	等宽斜体为占位符，表示须由用户填写文字或数值。
平台	平台字体表示特定的平台，下文所提的内容只应用于该平台。
单击右键	**(Mac OS)** 按 <Command> 键并单击，相当于单击右键。

1

LabVIEW 简介

LabVIEW (Laboratory Virtual Instrument Engineering Workbench) 是一种用图标和连线代替文本行来创建应用程序的图形化编程语言。传统文本编程语言根据语句和指令的先后顺序决定程序执行顺序，而 LabVIEW 则采用数据流编程方式，即在程序框图中节点之间的数据流向决定 VI 及函数的执行顺序。 VI(virtual instruments) 指虚拟仪器，是 LabVIEW 的程序模块。

LabVIEW 提供很多外观与传统仪器 （如示波器、万用表）类似的控件，可用来方便地创建用户界面。用户界面在 LabVIEW 中被称为前面板 (front panel)。使用图标和连线，可以通过编程对前面板上的对象进行控制。这就是图形化源代码，又称 G 代码。 LabVIEW 的图形化源代码在某种程度上类似于流程图，因此又被称作程序框图 (block diagram) 代码。

如需开发特定程序，可购买各类附加软件工具包。所有工具包都与 LabVIEW 无缝集成。关于工具包的详细信息，请查询 National Instruments 网站 `ni.com\toolkits`。

LabVIEW 文档资源

LabVIEW 附带全面的参考文档，均有网页和印刷品两种版本，供 LabVIEW 初级或高级用户使用。

LabVIEW 帮助

LabVIEW 帮助 (LabVIEW Help) 包含 LabVIEW 编程理论、编程分步指导以及 VI、函数、选板、菜单和工具的参考信息。

LabVIEW Help 中详细列出 National Instruments 网站中技术支持方面的链接，如 NI 开发者园地 (NI Developer Zone)、知识库 (KnowledgeBase)、产品手册文库等。

请在菜单中选择 **Help»Search the LabVIEW Help** 打开 *LabVIEW Help*。同时还可以在 *LabVIEW Help* 中选择打印您所需的一个或多个帮助主题。

关于打印帮助主题的详细信息见 *LabVIEW Help*。

注　**(Mac OS)** 建议使用 Safari 1.0 或更高版本、 Firefox 1.0.2 或更高版本浏览 *LabVIEW Help*。**(Linux)** 建议使用 Netscape 6.0 或更高版本、Mozilla 1.2 或更高版本、 Firefox 1.0.2 或更高版本浏览 *LabVIEW Help*。

安装 LabVIEW 附加软件 （如工具包、模块、驱动程序）后，附加软件的相关文档将出现在 *LabVIEW Help* 中，或者将自带帮助文件存在 **Help»*Add-On Help*** 目录下 （***Add-On Help*** 指该附件软件的帮助文件名）。

印刷文档

在使用 LabVIEW 时，可参考以下文档。

- *《LabVIEW 入门》(Getting Started with LabVIEW)* －本手册可帮助您初步了解 LabVIEW 图形化编程环境，掌握一些创建数据采集和仪器控制程序的 LabVIEW 功能。
- *《LabVIEW 快速参考指南》(LabVIEW Quick Reference Card)* －本指南提供帮助文档资源、快捷键、数据类型及编辑、执行、调试工具的相关信息。
- *《LabVIEW 基础》(LabVIEW Fundamentals)*－本文档包括LabVIEW的编程理论、技巧、作用、 VI 和功能，用于创建测试测量、数据采集、仪器控制、数据记录、测量分析和报表生成等各类程序。 *LabVIEW Help* 中涵盖本手册的所有内容。
- *《LabVIEW 发行说明》(LabVIEW Release Notes)* －介绍如何安装和卸载 LabVIEW，以及 LabVIEW 软件 （包括 LabVIEW Application Builder）的系统要求。
- *《LabVIEW 升级说明》(LabVIEW Upgrade Notes)* －说明如何在 Windows、 Mac OS 或 Linux 上升级 LabVIEW。同时还介绍升级版本的新功能和升级时可能出现的问题。

这些文档均有印刷版本和 PDF 版本， PDF 版本在 `labview\ manuals` 目录下。正常显示该 PDF 文档需 Adobe Reader 5.0.5 或更高版本。如需查询所有 LabVIEW 用户手册的 PFD 文档，请安装带有 Search and Accessibility 6.*x* 或更高版本的 Adobe Reader。 **(Mac OS)** 需安装带 Search and Accessibility 6.*x* 或更高版本的 Adobe Reader 才能正常显示该 PDF 文档。

请登录 Adobe Systems Incorporated 网站 `www.adobe.com` 下载 Acrobat Reader。关于文档资源更新的详细信息，请查阅 `ni.com/manuals`。

自述文件

在使用 LabVIEW 时，可参考以下自述文件。

- *LabVIEW 自述文件 (Readme)* —介绍 LabVIEW 的最新信息，包括安装和升级、兼容信息、升级改动以及现存问题的记录。请选择**开始 » 程序 »National Instruments»LabVIEW 8.0 Readme**，打开 `readme.html`，或者在 `labview\readme` 目录下直接打开 `readme.html` 文件，查看 *LabVIEW Readme*。
- *LabVIEW 应用程序生成器自述文件 (LabVIEW Application Builder Readme)* —说明如何安装 LabVIEW Application Builder。LabVIEW 专业版开发系统 (LabVIEW Professional Development System) 附带 LabVIEW Application Builder；也可单独购买。请选择**开始 » 程序 »National Instruments»LabVIEW 8.0 Readme**，打开 `readme_AppBldr.html`，或者在 `labview\readme` 目录下直接打开 `readme_AppBldr.html` 文件，查看 *LabVIEW Application Builder Readme*。

LabVIEW VI 模板、 VI 范例和工具

初学 VI 设计可以先从 LabVIEW VI 模板 (LabVIEW VI templates)、VI 范例 (Example VIs) 和工具开始。

LabVIEW VI 模板

LabVIEW 内置的 VI 模板包括用来创建通用测量应用程序所必需的子 VI、函数、结构和前面板对象。可以通过修改这些 VI 模板来构建所需的应用程序。请选择 **File»New** 打开 **New** 对话框，上面列出所有内置 VI 模板。也可在 **Getting Started** 窗口中单击 **New** 链接，打开 **New** 对话框。

LabVIEW VI 范例

LabVIEW 包含了数百个 VI 范例，这些范例可用来了解 LabVIEW 的编程技巧、功能、 VI 和函数的使用方法、应用领域等。通过对一个 VI 范例进行修改，或从一个或多个范例中复制并粘贴到新的 VI 中可以创建自己的应用程序。请选择 **Help»Find Examples**，使用 NI Example Finder 查看或搜索 VI 范例。

关于其它 VI 范例请查阅 NI 开发者园地 （`ni.com/zone`）。

请打开 *LabVIEW Help*，在 VI 和函数介绍 (VI and function reference) 目录下找到相应的 VI，在页面下方点击 **Open example** 或 **Browse related examples**，打开 VI 范例。单击 **Open example** 可打开与该主题相关的 VI 范例。单击 **Browse related examples** 打开 NI 范例搜索器 (NI Example Finder)，显示相关 VI 范例。

还可以在程序框图或锁定选板中右键单击 VI 或函数，在快捷菜单中选择 (**Examples**)，打开帮助主题，其中包含了该 VI 或函数的范例链接。

配置 DAQ 所需的 LabVIEW 工具 (Windows)

LabVIEW 中的 Measurement & Automation Explorer (MAX) 可用于配置测量设备。请选择 **Tools»Measurement & Automation Explorer** 打开 MAX，配置 NI 软硬件。请从 NI 设备驱动光盘中安装 MAX。

关于管理其它类型仪器的相关信息，请打开 *LabVIEW Help*，在 **Contents** 一栏中查阅 **Controlling Instruments**。

请使用 DAQ Assistant Express VI 通过图形界面配置通道或测量任务。只有在安装 NI-DAQmx 后，DAQ Assistant Express VI 才会在**函数**选板上显示。关于安装 NI-DAQmx 的详细信息见 DAQ 快速入门指南 (*DAQ Quick Start Guide*)。可用以下几种方式打开 DAQ Assistant：

- 将 DAQ Assistant Express VI 置于程序框图中。
- 右键单击 DAQmx 全局通道控件，在快捷菜单中选择 **New Channel (DAQ Assistant)**。右键单击 DAQmx 任务控件，在快捷菜单中选择 **New Task (DAQ Assistant)**。右键单击 DAQmx 测量刻度控件，在快捷菜单中选择 **New Scale (DAQ Assistant)**。
- 打开 Measurement & Automation Explorer，在 **Configuration** 目录树下选择 **Data Neighborhood** 或 **Scales**。单击 **Create New** 按钮。配置 NI-DAQmx 通道、任务和测量单位。

2

虚拟仪器简介

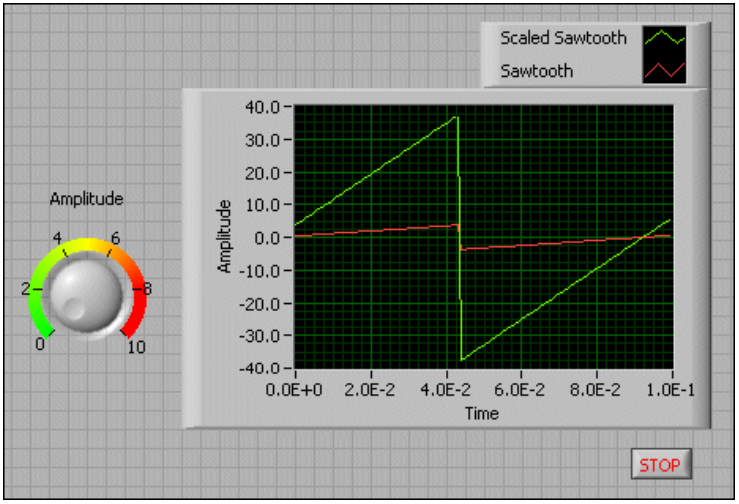
LabVIEW 程序又称虚拟仪器 (virtual instruments)，即 VI，其外观和操作类似于真实的物理仪器（如示波器和万用表）。VI 从用户界面或其它资源获取信息输入，然后显示或传输至其它文件或计算机。

每个 VI 都包含以下三个组件：

- **前面板 (Front panel)** —即用户界面。
- **程序框图 (Block diagram)** —包含用以定义 VI 功能的图形化源代码。
- **图标和接线器 (Icon and connector pane)** —用以识别 VI 的接口，以便在创建 VI 时调用另一个 VI。当一个 VI 应用在其它 VI 中，则称为子 VI。子 VI 相当于文本编程语言中的子程序。

前面板

前面板是 VI 的用户界面。前面板示例如下：



输入控件 (controls) 和显示控件 (indicators) 用于创建前面板，它们分别是 VI 的交互式输入输出端口。输入控件是指旋钮、按钮、转盘等输入装置。显示控件是指图表、指示灯等显示装置。输入控件模拟仪器输入装置，为 VI 的程序框图提供数据。显示控件模拟仪器输出装置，用以显示程序框图所获取或生成的数据。

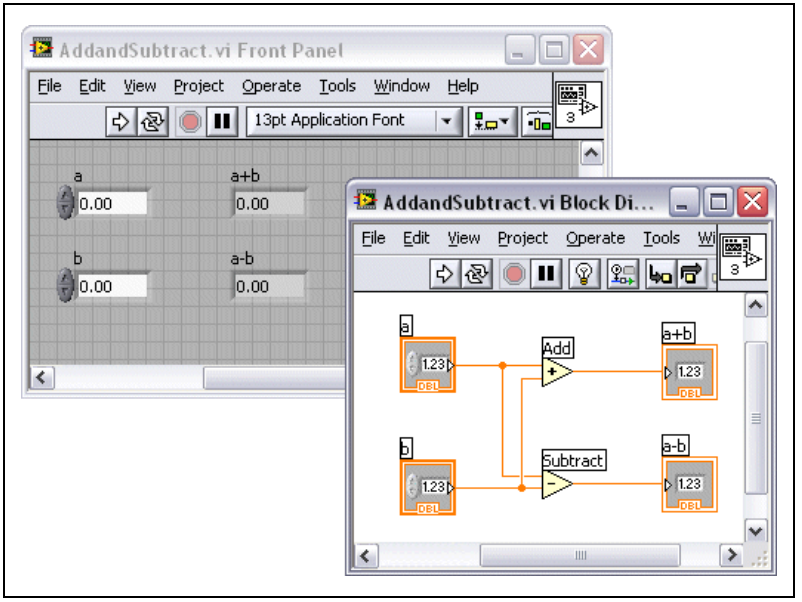
关于程序框图的更多信息参见第 4 章*创建前面板*。

程序框图

创建前面板后，可使用图形化的函数添加源代码，用以控制前面板上的对象。程序框图是图形化源代码的集合。图形化源代码又称 G 代码，或程序框图代码。前面板上的对象 (objects) 在程序框图中显示为接线端 (terminals)。

关于程序框图的更多信息参见第 5 章*创建程序框图*。

下图中的 VI 包含一些基本的程序框图对象：接线端、函数和连线。



接线端

接线端 (terminal) 用以表示输入控件或显示控件的数据类型。在程序框图中可将前面板的输入控件或显示控件显示为图标或数据类型接线端。默认状态下，前面板对象以图标形式显示。比如：旋钮接线端代表前面板上的一个旋钮，如下所示。

接线端底部 DBL 代表双精度浮点数的数据类型。如下所示的 DBL 接线端代表一个双精度浮点数输入控件。

关于数据类型的更多信息见第 5 章*创建程序框图*的*输入控件和显示控件的数据类型*一节。

接线端是在前面板和程序框图之间交换信息的输入输出端口。在前面板控件中输入的数据 (如上图中的 **a** 和 **b**) 将通过控件接线端传输至程序框图。然后进行数据的加减运算。加减运算结束后，将输出新的数据值。数据将传输至显示控件接线端，更新前面板显示控件中的数据 (如上图中的 **a+b** 和 **a-b**)。

节点

节点 (Nodes) 是程序框图上的对象，带有输入输出端，在 VI 运行时进行运算。节点相当于文本编程语言中的语句、运算、函数和子程序。上图中的加减运算即是一个节点。

关于节点的详细信息见第 5 章*创建程序框图*中的*程序框图节点*一节。

连线

LabVIEW 通过连线 (wire) 在程序框图对象之间传递数据。在上图中，输入控件和显示控件接线端通过连线实现加减运算。每根连线都只有一个数据源，但可以与多个读取该数据的 VI 和函数连接。不同数据类型的连线有不同的颜色、粗细和样式。断开的连线显示为黑色的虚线，中间有个红色的 x。出现断线的原因有很多，比如连接数据类型不兼容的两个对象时就会产生断线。

关于连线的详细信息见第 5 章*创建程序框图*中的*使用连线连接程序框图各对象*一节。

结构

结构 (Structures) 是传统编程语言中的循环和选择语句的图形化表示。使用程序框图中的结构可对代码组进行重复操作、有条件执行或按特定顺序执行。

关于结构的详细信息见第 8 章*循环和结构*。

图标和接线器

创建 VI 的前面板和程序框图后，请设置图标 (icon) 和接线器 (connector pane)，以便该 VI 作为子 VI 在其它 VI 中调用。图标和接线器相当于文本编程语言中的函数原型。每个 VI 都有一个图标，位于前面板和程序框图窗口的右上角。如下图所示。

图标是 VI 的图形化表示，可包含文字、图形或图文组合。如果将一个 VI 当作子 VI 使用，程序框图上将显示代表该子 VI 的图标，可双击图标进行修改或编辑。

关于图标的详细信息见第 7 章*创建 VI 和子 VI* 中的*创建图标*一节。

如需使用子 VI，还需要创建接线器，如下所示。

接线器用于显示 VI 中所有输入控件和显示控件接线端，类似于文本编程语言中调用函数时使用的参数列表。接线器标明了可与该 VI 连接的输入和输出端，以便将该 VI 作为子 VI 调用。接线器在其输入端接收数据，然后通过前面板的控件传输至程序框图的代码中，并从前面板的显示控件中接收运算结果传输至其输出端。

关于创建接线器的详细信息见第 7 章*创建 VI 和子 VI* 中的*创建接线器*一节。

注　一个 VI 的接线端应尽量控制在 16 个以内。接线端太多将影响 VI 的可读性和可用性。

VI 和子 VI 的应用和自定义

创建好一个 VI 并设定图标和接线器后，该 VI 将可作为子 VI 调用。

关于子 VI 的详细信息见第 7 章*创建 VI 和子 VI* 中的*创建子 VI* 一节。

VI 的外观和运行方式都可以进行自定义。

关于自定义子 VI 的详细信息见第 7 章*创建 VI 和子 VI* 中的*自定义子 VI* 一节。

3

LabVIEW 编程环境

LabVIEW 选板、工具、菜单用于创建 VI 的前面板 (front panel) 和程序框图 (block diagram)。 LabVIEW 中包含三种选板：**控件 (Controls)** 选板、**函数 (Functions)** 选板和**工具 (Tools)** 选板。LabVIEW 中还有**启动向导 (Getting Started)** 窗口、**即时帮助 (Context Help)** 窗口、**项目浏览 (Project Explorer)** 和**导航 (Navigation)** 窗口。您可以自定义**控件**和**函数**选板，同时还可以设置多个工作环境选项。

启动向导窗口

打开 LabVIEW 时将显示 **Getting Started** 窗口。在这个窗口中可以创建新 VI、选择最近打开的 LabVIEW 文件、查找范例以及打开 *LabVIEW 帮助 (LabVIEW Help)*。同时还可查看各种信息和资源，如 《用户手册》、帮助主题以及 National Instruments 网站 `ni.com` 上的各种资源等。

打开文件或创建新文件时 **Getting Started** 窗口会隐藏。关闭了所有已打开的前面板和程序框图后 **Getting Started** 窗口会再次显现。选择 **View»Getting Started Window**, 可以显示该窗口。

控件选板

控件 (Controls) 选板在前面板中。**控件**选板中集合了创建前面板所需的输入控件和显示控件。根据类型的不同，输入控件和显示控件位于不同的子选板上。

关于输入控件和显示控件类型的详细信息见第 4 章*创建前面板*中的*前面板输入控件和显示控件*一节。

如需显示**控件**选板，请选择 **View»Controls Palette** 或在前面板活动窗口单击右键。 LabVIEW 将记住**控件**选板的位置和大小，因此当 LabVIEW 重启时选板的位置和大小不变。您可以自定义**控件**选板中需要显示的控件。

关于自定义**控件**选板的详细信息见本章中*自定义控件和函数选板*一节。

函数选板

函数 (Functions) 选板在程序框图中。**函数**选板中包含创建程序框图所需的 VI 和函数。根据类型的不同， VI 和函数位于不同的子选板上。

如需显示**函数**选板，请选择 **View»Functions Palette** 或在程序框图活动窗口单击右键。 LabVIEW 将记住**函数**选板的位置和大小，因此当 LabVIEW 重启时选板的位置和大小不变。您可以自定义**函数**选板中需要显示的函数和 VI。

关于自定义**函数**选板的详细信息见本章中 *自定义控件和函数选板*一节。

浏览控件和函数选板

单击选板上的对象，使光标指向该对象，将其拖放到前面板或程序框图上。也可在选板的 VI 图标上单击右键，从快捷菜单中选择 **Open VI**。

单击**控件**或**函数**选板中各栏目左边的黑色箭头可展开或缩进子选板。只有将选板模式设为 **Category (Standard)** 或 **Category (Icons and Text)** 时这些黑色箭头才会显示。

控件和**函数**选板工具栏上的图标用于查看、配置选板，搜索控件、 VI 和函数，如下所示。

	**Up** －转到选板的上级目录。单击该按钮并将光标停在上面，将显示该子选板的所有下级选板列表。单击快捷菜单上的子选板名称进入子选板。只有当选板模式设为 **Icons**、 **Icons and Text** 或 **Text** 时这些按钮才会显示。
	**Search** －用于将选板转换为搜索模式，可通过文字输入来查找选板上的控件、 VI 或函数。选板处于搜索模式时可单击 **Return** 按钮，将退出搜索模式，重新显示选板。
	**View** －用于选择当前选板的视图模式，显示或隐藏所有选板目录、在 **Text** 和 **Tree** 模式下按字母顺序对栏目进行排序。在快捷菜单中选择 **Options**，显示 **Options** 对话框中的 **Controls/Functions Palettes** 选项卡，可为所有选板选择显示模式。只有当点击选板左上方的图钉标识将选板锁住时，该按钮才会显示。
	**Restore Palette Size** －将选板恢复至默认大小。只有在**控件**或**函数**选板大小被改变后该选项才会显示。

工具选板

在前面板和程序框图中都可看到**工具 (Tools)** 选板。工具选板上的每一个工具对应于鼠标的一个操作模式。鼠标的光标随所选择的工具图标变化。可选择适当的工具对前面板和程序框图上的对象进行操作和修改。

如果 “自动工具选择”已打开，当光标移到前面板或程序框图的对象上时，LabVIEW 将自动从**工具**选板中选择相应的工具。

请选择 **View»Tools Palette** 打开**工具**选板。 LabVIEW 将记住**工具**选板的位置和大小，因此当 LabVIEW 重启时选板的位置和大小不变。

提示　按 <Shift> 键并单击右键，光标处将显示**工具**选板。

菜单和工具栏

菜单和工具栏用于操作和修改前面板和程序框图上的对象。

菜单

VI 窗口顶部的菜单为通用菜单，对其它 VI 程序同样适用，如 **Open**、**Save**、 **Copy** 和 **Paste**，以及其它 LabVIEW 特定功能的菜单项。有些菜单选项有快捷键。

(Mac OS) 菜单在屏幕最上方。

(Windows and Linux) 默认状态下，菜单只显示最近常用的菜单项。单击菜单底部的箭头可显示所有菜单选项。如需设置默认状态下显示所有菜单选项，请选择 **Tools»Options**，在 **Category** 中选择 **Environment**，取消勾选 **Use abridged menus** 复选框。

注　VI 运行时，有些菜单项不可用。

快捷菜单

所有 LabVIEW 对象均有相关的快捷菜单。创建 VI 时，可使用快捷菜单上的选项改变前面板和程序框图上对象的外观或操作。右键单击对象可打开快捷菜单。

(Mac OS) 按 <Command> 键并单击相当于右键单击。

运行模式下的快捷菜单

VI 处于运行模式时，所有前面板的对象都有一套常用的快捷菜单选项。可使用常用快捷菜单来剪切、复制、粘贴对象的内容、将对象的值恢复为默认值或查看该对象的说明。

一些更复杂的控件具有更多选项。例如：旋钮快捷菜单中包含 “添加指针”、“修改刻度显示”等功能。

VI 工具栏

工具栏按钮用于运行、中断、终止、调试 VI、修改字体、对齐、分组、分布对象。

关于工具栏按钮的详细信息见第 6 章*运行和调试 VI*，或参阅 *LabVIEW Help* 中完整的工具栏按钮列表和说明。

项目浏览器窗口工具栏 (Project Explorer Window Toolbar)

Standard、**Project**、**Build** 和 **Source Control** 工具栏中的各个按钮可用于执行 LabVIEW 项目的各种操作。工具栏位于 **Project Explorer** 窗口顶端。有时可能需要扩展 **Project Explorer** 窗口查看所有工具栏。

关于 LabVIEW 项目的详细信息参见本章的*项目浏览器窗口*一节。

即时帮助窗口

将光标移至 LabVIEW 对象上时，**即时帮助 (Context Help)** 窗口会显示每一对象的基本信息。VI、函数、常数、结构、选板、属性、方式、事件、对话框和 **Project Explorer** 中的项目均包含即时帮助信息。**Context Help** 窗口还可用于帮助确定 VI 或函数的连线位置。

关于如何使用 **Context Help** 为对象连线的详细信息见第 5 章*创建程序框图*中的*使用连线连接程序框图各对象*一节。

选择 **Help»Show Context Help** 显示 **Context Help** 窗口。在工具栏中选择 **Show Context Help Window**，也可显示 **Context Help**，如下所示。

(Windows) 按 <Ctrl-H> 键显示该窗口。**(Mac OS)** 按 <Command-H> 键。**(Linux)** 按 <Alt-H> 键。

Context Help 窗口可根据内容的多少自动调整大小。也可调整 **Context Help** 窗口的大小使之最大化。LabVIEW 将记住 **Context Help** 窗口的位置和大小，因此当 LabVIEW 重启时该窗口的位置和最大尺寸不变。

如果 **Context Help** 窗口中提及的对象在 *LabVIEW Help* 中有描述，则 **Context Help** 窗口中会出现一个蓝色的 **Detailed help** 链接。同时也可单击 **Context Help** 中的 **Detailed help** 图标，如下所示。单击该链接或图标可查看更多关于对象的信息。

项目浏览器窗口

项目浏览器窗口 (Project Explorer Window) 用于创建和编辑 LabVIEW 项目。项目 (project) 可用于集合 LabVIEW 文件和非 LabVIEW 文件、创建程序生成规范以及在终端中部署或下载文件。选择 **File»New Project** 显示 **Project Explorer** 窗口。

导航窗口

导航 (Navigation) 窗口显示活动前面板 （编辑模式下）或程序框图的全局概况。**Navigation** 窗口可用于浏览较大的前面板或程序框图。单击 **Navigation** 窗口的某一图像区域，在前面板和程序框图上将显示该区域的相应位置。同时也可单击并拖动 **Navigation** 窗口上的图像，这时前面板和程序框图上的图像也随之移动。前面板和程序框图中不可见的部分在 **Navigation** 窗口中显示为阴影。

选择 **View»Navigation Window** 打开 **Navigation** 窗口。**(Windows)** 按 <Ctrl-Shift-N> 键显示该窗口。**(Mac OS)** 按 <Command-Shift-N> 键。**(Linux)** 按 <Alt-Shift-N> 键。

注　只有在 LabVIEW 完整版和专业版开发系统 （Full and Professional Development Systems）中才有 **Navigation** 窗口。

Navigation 窗口的大小可调节。LabVIEW 将记住 **Navigation** 窗口的位置和大小，因此当 LabVIEW 重启时该窗口的位置和大小不变。

自定义工作环境

您可以对**控件**和**函数**选板进行自定义，也可用 **Options** 对话框选择选板视图并配置其它工作环境选项。

自定义控件和函数选板

以下方法可用于自定义**控件**和**函数**选板。

- 配置 **Edit Controls and Functions Palette Set** 对话框，重新排列内置选板、创建或移动子选板。选择 **Tools»Advanced»Edit Palette Set**，显示 **Edit Controls and Functions Palette Set** 对话框。右键单击需修改的选板，从快捷菜单中进行选择。
- **函数**选板中的项目可以添加到 Favorites 目录中。在锁定的**函数**选板上，右键单击对象并从快捷菜单中选择 **Add Item to Favorites**。在 **Category (Standard)** 和 **Category (Icons and Text)** 格式中，也可以展开一个选板以显示其子选板，右键单击该子选板名称并从快捷菜单中选择 **Add Subpalette to Favorites**。

工作环境设置

选择 **Tools»Options** 可自定义 LabVIEW。 **Options** 对话框中可设置前面板、程序框图、路径、性能和磁盘相关选项、对齐网格、选板、撤消操作、调试工具、颜色、字体、打印、**修订记录**等各项 LabVIEW 功能。

在 **Options** 对话框左栏中的 **Category** 列表中列出了可以进行设置的各类选项。

4

创建前面板

前面板 (front panel) 是 VI 的用户界面。通常，首先设计前面板，然后设计程序框图 (block diagram) 执行在前面板上所创建的输入输出任务。

关于程序框图的更多信息参见第五章*创建程序框图*。

输入控件 (controls) 和显示控件 (indicators) 可用来创建前面板，它们分别是 VI 的交互式输入输出端口。输入控件是指旋钮 (knobs)、按钮 (push buttons)、转盘 (dials) 等输入装置。显示控件是指图形 (graphs)、指示灯 (LEDs) 等显示装置。输入控件模拟仪器输入装置，为 VI 的程序框图提供数据。显示控件模拟仪器输出装置，用以显示程序框图所获取或生成的数据。

选择 **View»Controls Palette** 可以显示**控件 (Controls)** 选板，从中选出输入控件和显示控件，将它们放置在前面板上。

前面板输入控件和显示控件

位于**控件**选板上的前面板输入控件和显示控件可用于创建前面板。输入控件和显示控件的种类有：数值输入控件和显示控件如滑动杆和旋钮 (slides and knobs)、图形 (graphs)、图表 (charts)、布尔输入控件和显示控件如按钮和开关 (buttons and switches)、字符串 (strings)、路径 (paths)、数组 (arrays)、簇 (clusters)、列表框 (listboxes)、树形控件 (tree controls)、表格 (tables)、下拉列表控件 (ring controls)、枚举控件 (enumerated type controls) 和容器控件 (containers) 等等。

输入控件和显示控件的式样

前面板输入控件和显示控件具有新式、经典和系统式样。

新式及经典输入控件和显示控件

许多前面板对象具有高彩外观。为了获取对象的最佳外观，显示器至少要设置成 16 色。

位于**新式 (Modern)** 选板上的输入控件和显示控件除了具有高彩对象外，同时也具有相应的低彩对象。而位于**经典 (Classic)** 选板上的输入控件和显示控件可用于创建适合 256 色和 16 色显示器的 VI。

系统输入控件和显示控件

位于**系统 (System)** 选板上的系统输入控件和显示控件可应用于用户创建的对话框中。系统输入控件和显示控件特别为在对话框中使用而设计，包括下拉列表和滚动选择控件 (ring and spin controls)、数值滑动杆 (numeric slides)、进度条 (progress bars)、滚动条 (scroll bars)、列表框 (listboxes)、表格 (tables)、字符串和路径控件 (string and path controls)、 tab 控件 (tab controls)、树形控件 (tree controls)、按钮 (buttons)、复选框 (checkboxes)、单选按钮 (radio buttons) 和自动匹配父对象背景色的不透明标签 (opaque label)。这些控件仅仅在外观上与前面板控件不同。它们的颜色与系统设置的颜色保持一致。

由于系统控件根据不同的运行 VI 的平台而改变其外观，在 VI 中创建的控件外观与所有 LabVIEW 平台都兼容。在不同的平台上运行 VI 时，系统控件将改变其颜色和外观使其与该平台的标准对话框控件相匹配。

关于设计对话框的信息参见本章*设计对话框*一节。

数值显示框、滑动杆、滚动条、旋钮、转盘和时间标识

位于**数值 (Numeric)** 和**经典数值 (Classic Numeric)** 选板上的数值对象可用于创建滑动杆 (slides)、滚动条 (scroll bars)、旋钮 (knobs)、转盘 (dials) 和数值显示框 (numeric displays)。该选板也包含用来设置颜色值的颜色盒 (color boxes) 和颜色梯度 (color ramp)，以及用来设置时间和日期值的时间标识 (time stamps)。数值对象用于输入和显示数值数据。

数值输入控件和显示控件

数值输入控件和显示控件 (Numeric controls and indicators) 是输入和显示数值数据最简单的方式。您可以水平改变这些前面板对象的大小以容纳更多的数据位。下述方法可用来改变数值控件的值：

- 用操作工具 (Operating tool) 或标注工具 (Labeling tool) 在数字显示框 (digital display) 内单击然后从键盘上输入数字。
- 用操作工具单击数值控件的递增或递减箭头按钮。
- 用操作工具或标注工具将光标放置于欲改变的数字右边，然后在键盘上按向上或向下箭头键。

默认状态下， LabVIEW 像计算器一样显示和存储数字。在自动切换为科学计数法前，数值输入控件或显示控件最多显示 6 位数字。右键单击数值对象并从快捷菜单中选择 **Format and Precision** 可以显示 **Numeric Properties** 对话框的 **Format and Precision** 选项卡，从中配置 LabVIEW 在切换到科学计数法之前所显示的数字位数。

滑动杆输入控件和显示控件

滑动杆输入控件和显示控件 (Slide controls and indicators) 是带有刻度的数值对象。滑动杆输入控件和显示控件包括垂直和水平方向滑动杆 (vertical and horizontal slides)、液罐 (tank) 和温度计 (thermometer)。下述方法可用来改变滑动杆控件的值：

- 用操作工具 (Operating tool) 单击或拖曳滑块至一个新的位置。
- 向数字显示框中输入数据，就如同对数值输入控件和显示控件进行操作时一样。

滑动杆输入控件和显示控件可以显示一个以上的值。右键单击该对象并从快捷菜单中选择 **Add Slider** 可以添加多个滑块。含有多个滑块的控件的数据类型为簇，簇中包含每一个滑块所对应的数值。

关于簇的更多信息参见第 9 章 *使用字符串、数组和簇将数据分组*中的 *簇*一节。

滚动条输入控件和显示控件

与滑动杆控件相类似，滚动条控件 (Scroll bar controls) 是数值对象，可以用来滚动数据。滚动条控件包括水平和垂直滚动条。通过操作工具单击或拖曳滚动框至一个新的位置，单击递增和递减箭头，或单击滚动框和箭头之间的空间都可以改变滚动条的值。

旋转型输入控件和显示控件

旋转型输入控件和显示控件 (rotary controls and indicators) 包括旋钮 (knobs)、转盘 (dials)、量表 (gauges) 和仪表 (meters)。旋转型对象操作起来与滑动杆输入控件和显示控件相似，因为它们都是带有刻度的数值对象。下述方法可用来改变旋转型控件的值：

- 用操作工具单击或拖曳指针至一个新的位置。
- 向数字显示框中输入数据，就如同对数值输入控件和显示控件进行操作时一样。

旋转型输入控件或显示控件可以显示一个以上的值。右键单击对象然后选择 **Add Needle** 可以添加新的指针。含有多个指针的控件的数据类型为簇，簇中包含每一个指针所对应的数值。

关于簇的更多信息参见第 9 章 *使用字符串、数组和簇将数据分组*中的 *簇*一节。

时间标识输入控件和显示控件

时间标识输入控件和显示控件 (time stamp control and indicator) 用于向程序框图发送或者从程序框图获取时间和日期的值。可以通过以下任一方法改变时间标识控件的值：

- 右键单击控件并从快捷菜单中选择 **Format & Precision**。
- 如下所示，单击 **Time/Date Browse** 按钮来显示 **Set Time and Date** 对话框。

- 右键单击控件并从快捷菜单中选择 **Data Operations»Set Time and Date** 来显示 **Set Time and Date** 对话框。
- 右键单击控件，从快捷菜单中选择 **Data Operations»Set Time to Now**。

图形和图表

位于**图形 (Graph)** 和**经典图形 (Classic Graph)** 选板上的图形输入控件和显示控件用于以图形和图表 (chart) 形式绘制数值数据。

关于在 LabVIEW 中使用图形和图表的更多信息见第 10 章*图形和图表*。

按钮、开关和指示灯

位于**布尔 (Boolean)** 和**经典布尔 (Classic Boolean)** 选板上的布尔输入控件和显示控件可用于创建按钮 (buttons)、开关 (switches) 和指示灯 (lights)。布尔输入控件和显示控件可用于输入并显示布尔值 (TRUE/FALSE)。例如，在监控一个实验的温度时，可以在前面板上放置一个布尔报警指示灯以显示温度在何时超出指定值。

布尔控件具有六种机械动作，可用于自定义布尔对象，创建与物理仪器行为更为相似的前面板。使用快捷菜单可以定制布尔对象的外观以及单击时这些控件相应的动作。

单选按钮控件

单选按钮控件 (radio buttons control) 用于向用户提供一个列表，用户一次只能从中选择一项。如果要使用户可以不选或选择一项，右键单击该控件然后在快捷菜单中选择 **Allow No Selection**，这样在该菜单项旁边就放置了一个选中标记。

由于单选按钮控件的数据类型为枚举型 (enumerated type), 可以用它来选定条件结构 (Case structure) 的条件分支。

关于枚举控件的更多信息见本章*枚举控件*一节。关于条件结构的更多信息见第 8 章*循环和结构*中的*条件结构*一节。

关于使用单选按钮控件的范例见 `labview\examples\general\controls\booleans.llb` 中的 Radio Buttons Control VI 和 Radio Buttons with Event Structure VI。

文本输入框、标签和路径显示

位于**字符串和路径 (String & Path)** 及**经典字符串和路径 (Classic String & Path)** 选板上的字符串和路径输入控件和显示控件可用于创建文本输入框 (text entry boxes) 和标签 (labels)，输入或返回文件或目录的地址。

字符串输入控件和显示控件

操作工具或标注工具可用于输入或编辑前面板上某个字符串控件中的文本。在默认状态下，新文本或编辑过的文本在编辑操作结束之前不会传送给程序框图。在运行时，单击面板的其他位置，切换到另一窗口，单击工具栏上的 **Enter** 按钮，或按数值键区的 <Enter> 键，都可以结束编辑操作。在键盘上按 <Enter> 键输入的是回车字符。

右键单击字符串输入控件或显示控件 (string control or indicator) 可以为输入控件或显示控件中的文本选择显示类型，如密码显示或以十六进制数显示。

关于字符串显示类型的更多信息参见第 9 章*使用字符串、数组和簇将数据分组*中的*前面板上的字符串*一节。

组合框控件

组合框控件 (combo box control) 可用来创建一个字符串列表，在前面板上可以循环浏览该列表。组合框控件类似于文本或菜单下拉列表控件。但是，组合框控件的值和数据类型是字符串，而下拉列表控件的值和数据类型是数字。

关于下拉列表控件的更多信息参见本章*下拉列表控件*一节。

关于条件结构的更多信息见第 8 章*循环和结构*中的*条件结构*一节。

路径输入控件和显示控件

路径输入控件和显示控件 (path controls and indicators) 可用于输入或返回文件或目录的地址。**(Windows and Mac OS)** 如果运行时允许拖动，也可以从 Windows Explorer 拖曳一个路径、文件夹或文件放置在路径控件中。

路径输入控件和显示控件与字符串输入控件和显示控件行为相似，但是 LabVIEW 会根据用户所使用操作平台的标准句法对路径进行格式化。

数组、矩阵及簇输入控件和显示控件

位于**数组、矩阵和簇 (Array, Matrix & Cluster)** 及**经典数组、矩阵和簇 (Classic Array, Matrix & Cluster)** 选板上的数组、矩阵和簇输入控件及显示控件可用来创建数组、矩阵和簇。数组将相同类型的数据元素归为一组。簇将不同类型的数据元素归为一组。矩阵按实数或复数标量数据的行或列分组，用于线性代数等数学操作中。

关于数组和簇的更多信息参见第 9 章 *使用字符串、数组和簇将数据分组*中的 *使用数组和簇将数据分组*一节。

列表框、树形控件和表格

位于**列表和表格 (List & Table)** 及**经典列表和表格 (Classic List & Table)** 选板上的列表框控件可用于向用户提供一个可以选择的条目列表。

列表框

您可以将列表框 (listboxes) 配置为接受单个或多个选择。多列列表用于显示条目的更多信息，如大小和创建日期等。

树形控件

树形控件 (tree control) 可用于向用户提供一个可以选择的层次结构列表。它通过条目组或节点组的形式组织用户在树形控件中所输入的内容。单击节点旁边的展开符号可以展开节点，显示节点中的所有条目。单击节点旁边的缩进符号可以缩进节点。

注　只有在 LabVIEW 完全版和专业版开发系统中才可以创建和编辑树形控件。如果 VI 中含有树形控件，那么您可以在所有 LabVIEW 版本中运行该 VI，但不能在基础版中配置树形控件。

使用树形控件的范例请参见 `labview\examples\general\controls\Directory Tree Control.llb` 中的 Directory Hierarchy in Tree Control VI。

表格

表格控件 (table control) 可用于在前面板上创建表格。

关于使用表格控件的更多信息参见第 9 章 *使用字符串、数组和簇将数据分组*中的 *表格*一节。

下拉列表及枚举输入控件和显示控件

位于**下拉列表和枚举 (Ring & Enum)** 及**经典下拉列表和枚举 (Classic Ring & Enum)** 选板上的下拉列表及枚举输入控件和显示控件用于创建用户可以循环浏览的字符串列表。

下拉列表控件

下拉列表控件 (Ring controls) 是为字符串或图片指定数值的数值对象。下拉列表控件以下拉菜单的形式出现，用户可以循环浏览作出选择。

下拉列表控件可用于选择互斥条目，如触发模式。例如，下拉列表控件可使用户从连续、单次和外部触发模式中进行选择。

枚举控件

枚举控件 (enumerated type controls) 可用于向用户提供一个可以选择的条目列表。枚举控件类似于文本或菜单下拉列表控件。但是，枚举控件的数据类型包括控件中所有条目的数值和字符串标签的相关信息。而下拉列表控件的数据类型为数值型。

容器控件

位于**容器 (Containers)** 和**经典容器 (Classic Containers)** 选板上的容器控件 (container controls) 用于将输入控件和显示控件分组或在当前 VI 的前面板上显示另一个 VI 的前面板。**(Windows)** 容器控件还可用于在前面板上显示 .NET 和 ActiveX 对象。

关于 .NET 和 ActiveX 控件的更多信息见本章 *.NET 和 ActiveX 控件 (Windows)* 一节。

Tab 控件

Tab 控件 (tab controls) 可用于将前面板输入控件和显示控件重叠放置在一个较小的区域内。 Tab 控件由页 (pages) 和选项卡 (tabs) 组成。您可以将前面板对象放置在 Tab 控件的每一页上并用选项卡作为显示不同页的选择器。

当要一起使用几个前面板对象或在操作过程的某个特定阶段存在几个前面板对象时，您可以使用 Tab 控件。例如，有一个 VI 在测试开始前可能要求用户先设置几个选项，然后在测试过程中允许用户修改测试的某些方面，最后只允许用户显示和存储相关数据。

在程序框图上， Tab 控件在默认情况下属枚举控件。放置在 Tab 控件上的输入控件和显示控件接线端的外观跟任何其他程序框图接线端没有任何区别。

关于枚举控件的更多信息见本章*枚举控件*一节。

子面板控件

子面板控件 (subpanel control) 用于在当前 VI 的前面板上显示另一个 VI 的前面板。例如，子面板控件可用于设计一个类似向导的用户界面。具体方法是在顶层 VI 前面板上放置 **Back** 和 **Next** 按钮，并用子面板控件加载向导中每一步的前面板。

注　只能在 LabVIEW 完整版和专业版开发系统中创建和编辑子面板控件。如果一个 VI 含有子面板控件，那么可以在所有 LabVIEW 版本中运行该 VI，但不能在基础版中配置子面板控件。

使用子面板控件的范例请参见 `labview\examples\general\controls\subpanel.llb`。

I/O 名称输入控件和显示控件

位于 **I/O** 和 **Classic I/O** 选板上的 I/O 名称输入控件和显示控件 (I/O name controls and indicators) 可将所配置的 DAQ 通道名称、VISA 资源名称和 IVI 逻辑名称传送给 I/O VI，使其与仪器或者数据采集设备进行通讯。

I/O 名称常量位于**函数**选板上。常量是程序框图上向程序框图提供固定数据值的接线端。

注　在所有的平台上都可以使用所有的 I/O 名称控件或常量。这样，用户可以在任何平台上开发与特定平台下设备进行通讯的 I/O VI。但是，如果试图在一个不支持该设备的平台上运行带有特定平台 I/O 控件的 VI 将会出错。

(Windows) Tools 菜单中的测量和自动化浏览器 (Measurement & Automation Explorer) 可用于配置 DAQ 通道名称、 VISA 资源名称和 IVI 逻辑名称。

(Mac OS 和 Linux) 仪器配置应用程序 (configuration utilities) 可用于配置 VISA 资源名称和 IVI 逻辑名称。关于配置应用程序的更多信息参见相关仪器文档。

波形控件

波形控件 (waveform control) 用于对波形中的单个数据元素进行操作。波形数据类型包括波形的数据、起始时间和时间间隔 ($\triangle t$)。

关于波形数据类型的更多信息见第 10 章*图形和图表*中的*波形数据类型*一节。

数字波形控件

数字波形控件 (digital waveform control) 用于对数字波形中的单个数据元素进行操作。

关于数字波形数据类型的更多信息见第 10 章*图形和图表*中的*数字波形数据类型*一节。

数字数据控件

数字数据控件 (digital data control) 用于显示排列在行和列中的数字数据。数字数据控件可用于创建数字波形或显示从数字波形中提取的数字数据。将数字波形数据输入控件连接至数字数据显示控件，可以查看数字波形的采样和信号。

对象或应用程序的引用

位于**引用句柄 (Refnum)** 和**经典引用句柄 (Classic Refnum)** 选板上的引用句柄控件可用于对文件、目录、设备和网络连接进行操作。控件的引用句柄可将前面板对象信息传送给子 VI。

引用句柄 (reference number 或 refnum) 是一个对象的唯一标识符，这些对象包括文件、设备和网络连接等。打开一个文件、设备或网络连接时，LabVIEW 就会生成一个指向该文件、设备或网络连接的引用句柄。对打开的文件、设备或网络连接所进行的操作都通过引用句柄来识别每一个对象。引用句柄控件可将一个引用句柄传进或传出 VI。例如，引用句柄控件可以在不关闭或不必重新打开文件的情况下修改引用句柄所指向的文件内容。

由于引用句柄是一个打开对象的临时指针，因此它仅在对象打开期间有效。如果关闭对象， LabVIEW 会把引用句柄与对象分开，引用句柄将失效。如果再次打开对象， LabVIEW 将生成一个与第一个引用句柄不同的新引用句柄。 LabVIEW 也为引用句柄所指的对象分配内存空间。关闭引用句柄，该对象就会从内存中释放出来。

由于 LabVIEW 可以记住每个引用句柄所指的信息，如读取或写入的对象的当前地址和用户访问情况，因此可以对单一对象执行并列但相互独立的操作。如果一个 VI 多次打开同一个对象，那么每次的打开操作都将返回一个不同的引用句柄。 VI 结束运行时 LabVIEW 会自动关闭引用句柄，不过用户在结束使用引用句柄时就将其关闭可以最有效地利用内存空间和其他资源，这是一个很好的编程习惯。关闭引用句柄是以打开时相反的顺序进行的。例如，如果使对象 A 获得了一个引用句柄，然后要对对象 A 操作使对象 B 也获得一个引用句柄，请先关闭对象 B 的引用句柄然后再关闭对象 A 的引用句柄。

.NET 和 ActiveX 控件 (Windows)

位于 **.NET & ActiveX** 选板上的 .NET 和 ActiveX 控件可用于对常用的 .NET 或 ActiveX 控件进行操作。您还可以添加更多的 .NET 或 ActiveX 控件至该选板，供以后使用。选择 **Tools».NET & ActiveX»Add .NET Controls to Palette** 或 **Tools».NET & ActiveX Add ActiveX» Controls to Palette**，可以分别将 .NET 或 ActiveX 控件集转换成为自定义控件并将这些控件添加至 **.NET & ActiveX** 选板。

注　要创建 .NET 对象并与之通讯需安装 .NET Framework 1.1 Service Pack 1 或更高版本。 National Instruments 强烈建议您只在 LabVIEW 项目中使用 .NET 对象。

配置前面板对象

属性 (Properties) 对话框或快捷菜单可用来配置输入控件和显示控件在前面板上的外观和行为。 **Properties** 对话框带有即时帮助 (context help)，并且可以同时为一个对象设置多个属性来配置前面板上的控件。使用快捷菜单也可以快速配置输入控件和显示控件的通用属性。不同前面板对象的 **Properties** 对话框和快捷菜单选项也不同。任何用快捷菜单设置的选项会反映在 **Properties** 对话框里，任何用 **Properties** 对话框设置的选项也会在快捷菜单里反映出来。

右键单击前面板上的输入控件或显示控件，然后在快捷菜单中选择 **Properties** 进入该对象的 **Properties** 对话框。当 VI 运行时，不能进入输入控件或显示控件的 **Properties** 对话框。

您也可以创建自定义输入控件或显示控件来扩展前面板对象集。右键单击控件并从快捷菜单中选择 **Advanced» Customize**，便可以自定义输入控件或显示控件。您可以将创建的自定义输入控件或显示控件保存在一个目录或 LLB 中，以便用于其它前面板中。

显示和隐藏可选条目

前面板输入控件和显示控件中的一些条目可以被显示或隐藏，如标签、标题和数字显示框。在前面板对象 **Properties** 对话框的 **Appearance** 选项卡可以为前面板上的输入控件或显示控件设置显示条目。您也可以通过右键单击对象并从快捷菜单中选择 **Visible Items**，然后在可选项中选择来设置显示条目。

输入控件和显示控件的相互转换

LabVIEW 最初会根据**控件**选板中对象的典型用途将对象配置为输入控件或显示控件。例如，如果将扭动开关 (toggle switch) 放置于前面板上，它会显示为输入控件，因为扭动开关通常是一种输入设备。如果将指示灯 (LED) 放置于前面板上，它会显示为显示控件，因为指示灯通常是一种输出设备。

有些选板既包含输入控件又包含显示控件。例如，**Numeric** 子选板即包含数值输入控件和又包含数值显示控件，因为同时可以有数值输入和数值输出。

右键单击对象，并在快捷菜单中选择 **Change to Indicator**，可以将输入控件转换成显示控件。右键单击对象，并在快捷菜单中选择 **Change to Control**，可以将显示控件转换成输入控件。

替换前面板对象

您可以用不同的输入控件或显示控件来替换前面板对象。右键单击对象并从快捷菜单中选择 **Replace** 时，会出现一个临时的**控件**选板。从这个临时的**控件**选板中选择一个输入控件或显示控件便可以替换前面板上的当前对象。

配置前面板

您可以自定义前面板，如更改前面板对象的颜色、对齐和分布前面板对象等等。

为对象上色

许多对象的颜色可以改变，但不是所有对象。绝大多数前面板对象、前面板和程序框图工作区的颜色可以被改变。但是不能改变系统输入控件和显示控件的颜色，因为这些对象的颜色跟系统设置的颜色是一致的。

使用上色工具 (Coloring tool) 右键点击某个对象或工作区域可以改变前面板对象或前面板和程序框图工作区域的颜色。选择 **Tools»Options**，并从 **Category** 列表中选择 **Colors**，可以改变一些对象的的默认颜色。

颜色可能会转移用户的注意力使其无法注意到重要信息，所以应尽量合理地上色、用少量颜色并保持颜色的一贯性。

对齐和分布对象

选择 **Edit»Enable Panel Grid Alignment** 可以启用前面板网格对齐功能，在放置对象时自动对齐对象。选择 **Edit»Disable Panel Grid Alignment** 可以禁用网格对齐功能，而通过可见网格手动对齐对象。按 <Ctrl-#> 键也可以启用或禁用网格对齐。在法语键盘上，请按 <Ctrl-"> 键。

(Mac OS) 请按 <Command-*> 键。 **(Linux)** 请按 <Alt-#> 键。

在程序框图上也可以使用对齐网格。

选择 **Tools»Options** 然后从 **Category** 列表中选择 **Alignment Grid** 可以隐藏或自定义网格。

要在放置对象后对齐对象，请先选中该对象然后选择工具栏上的 **Align Objects** 下拉菜单或选择 **Edit»Align Items**。要均匀间隔对象，请先选中该对象然后选择工具栏上的 **Distribute Objects** 下拉菜单或选择 **Edit»Distribute Items**。

分组和锁定对象

定位工具 (Positioning tool) 可用来选择要一起分组 (group) 和锁定 (lock) 的前面板对象。单击工具栏上的 **Reorder** 按钮然后从下拉菜单中选择 **Group** 或 **Lock**。用定位工具移动或改变组内对象的大小时，这些对象将保持其相对位置和大小。锁定的对象保持它们在前面板上的位置，只有在解锁时才能删除这些对象。对象可以同时被分组并锁定。除了定位工具以外的其它工具都能对被分组或锁定的对象进行正常的操作。

改变对象大小

您可以改变大多数前面板对象的大小。将定位工具移动到某个大小可变的对象上时，调节柄或调节圈会出现在对象的大小可以被改变的地方。在改变对象大小时，对象的字体大小不会变化。改变一组对象的大小将同时改变组内所有对象的大小。

有些对象的大小只能在水平或垂直方向上被改变，如数值输入控件和显示控件。有些对象的长宽会成比例改变，如旋钮。改变这些对象的大小时，定位光标在外观上没有什么不同，但是对象周围的虚边框只能在一个方向移动。

改变对象大小时，可以手动限定对象变动的方向。要限定对象的大小只能在垂直或水平方向发生变化，或者要保持对象的当前比例，需在选中并拖曳调节柄或调节圈时同时按住 <Shift> 键。要以对象中心为参考点来改变大小，需在选中并拖曳调节柄或调节圈的同时按住 <Ctrl> 键。

(Mac OS) 请按 <Option> 键。 **(Linux)** 请按 <Alt> 键。

要将多个对象调整到同样大小，请选定这些对象然后选择工具栏上的 **Resize Objects** 下拉菜单。所有选定对象的大小都可以调整为最大或最小的那个对象的宽度或高度，所有选定对象也可以被调整到以象素为单位的特定尺寸。

在不改变窗口大小的情况下增加前面板空间

您无需改变窗口大小就可以为前面板增加空间。要为挤在一块或分组紧密的对象增加空间间隔，请按住 <Ctrl> 键然后用定位工具单击前面板工作区。在按下该组合键的时候，用鼠标拖出一块空白区域。

(Mac OS) 请按 <Option> 键。 **(Linux)** 请按 <Alt> 键。

带有虚线边框的矩形可限定所要插入的空间大小。释放鼠标按钮和组合键便可以增加空间。

添加标签

标签 (label) 用于识别前面板和程序框图上的对象。

LabVIEW 有两种标签－对象标签 (owned labels) 和自由标签 (free labels)。对象标签属于某一特定对象，并随对象的移动而移动，它仅注释该对象。对象标签可以被独立地移动，不过移动拥有该标签的对象时，标签将随之移动。您可以隐藏对象标签，但是无法离开对象标签的对象而复制或删除对象标签。右键单击数值输入控件和显示控件，从快捷菜单中选择 **Visible Items»Unit Label**，可以显示数值输入控件和显示控件的独立对象标签，即单位标签。

自由标签不附属于任何对象，可以独立地创建、移动、旋转或删除自由标签。自由标签可用于对前面板和程序框图进行注释。

自由标签也可用于注释程序框图上的代码以及在前面板上列出用户指令。双击空白区域或使用标注工具 (Labeling tool)，可以创建自由标签 , 或者编辑对象标签或是自由标签。

文本属性

LabVIEW 使用用户计算机上已经安装的字体。工具栏上的 **Text Settings** 下拉菜单可用于改变文本属性。

Text Settings 下拉菜单包含以下内置字体：

- **应用程序字体 (Application Font)** －用于**控件**和**函数**选板及新控件中的文本的默认字体
- **系统字体 (System Font)** －用于菜单
- **对话框字体 (Dialog Font)** －用于对话框中的文本

如果在 **Text Settings** 下拉菜单中作出选择前就已经选择了对象或文本，那么所选的全部对象或文本都将更改。如果未选任何对象或文本，那么只有默认字体会改变。改变默认字体并不会改变已有标签的字体。它仅影响此后创建的标签。

将含有某平台内置字体的 VI 安装到另一个平台上时，该字体会变为新的平台上最相近的字体。

Text Settings 下拉菜单也包含 **Size**、 **Style**、 **Justify** 和 **Color** 子菜单项。

设计用户界面

如果要将一个 VI 用作用户界面或对话框，那么前面板的外观和布局很重要。设计的前面板应该使用户能够轻松识别所要进行的操作。可以将前面板设计成类似于仪器或其它设备的外观。

使用前面板输入控件和显示控件

输入控件和显示控件是前面板的主要组成部分。在设计前面板时，须考虑用户与 VI 进行交互的方式，并对输入控件和显示控件进行逻辑分组。如果几个控件是相关的，可以为它们加上修饰边框，或者将它们放置在一个簇内。位于 **Decorations** 选板上的修饰控件可用于以方框，线条或箭头分组或分隔前面板上的对象。这些对象仅用于修饰对象，不能显示数据。

设计对话框

选择 **File»VI Properties**，然后从 **Category** 下拉菜单中选择 **Window Appearance**，可以隐藏菜单栏和滚动条，并为每个平台创建外观和行为如同标准对话框的 VI。

如果一个 VI 在屏幕的同一位置连续出现几个对话框，应该重新组织一下这些对话框，不要让第一个对话框中的按钮与后续对话框中的按钮排成一行。因为用户可能不经意中在双击第一个对话框中的按钮时单击了下一个对话框中的按钮。

位于 **System** 选板上的系统控件可应用到用户创建的对话框中。

5

创建程序框图

创建了前面板 (front panel) 之后，您可使用图形化的函数添加源代码来控制前面板上的对象。程序框图 (block diagram) 是图形化源代码的集合。图形化源代码又称 G 代码，或程序框图代码。

程序框图对象

程序框图对象包括接线端 (terminals) 和节点 (nodes)。通过连线连接各个对象，便创建了程序框图。每一个接线端的颜色和符号表明了相应输入控件或显示控件的数据类型。常量是程序框图上向程序框图提供固定数据值的接线端。

程序框图接线端

前面板上的对象 (objects) 在程序框图中显示为接线端。双击程序框图接线端，前面板上相应的输入控件或显示控件将突出显示。

接线端是在前面板和程序框图之间交换信息的输入输出端口。在前面板输入控件中输入的数据值通过输入控件接线端进入程序框图。在程序运行期间，输出数据值流进显示控件接线端，并通过程序框图输出重新进入前面板的显示控件中。

LabVIEW 包含输入控件和显示控件接线端、节点接线端、常量和结构上的专用接线端。用连线连接接线端便可以将数据传送给其它接线端。右键单击一个程序框图对象，从快捷菜单中选择 **Visible Items»Terminals**，可以显示接线端。用右键再次单击一个程序框图对象，从快捷菜单中选择 **Visible Items»Terminals**，可以隐藏接线端。但可扩展 VI 和函数没有这样的快捷菜单。

在程序框图中可将前面板的输入控件或显示控件显示为图标或数据类型接线端 (icon or data type terminals)。默认状态下，前面板对象以图标形式显示。比如，旋钮 (knob) 图标接线端代表前面板上的一个旋钮输入控件，如下所示。

接线端底部 DBL 代表的是双精度浮点数数据类型。如下所示的 DBL 接线端代表一个双精度浮点数输入控件。

右键单击一个接线端，取消勾选快捷菜单项 **View As Icon** 旁边的选中标记，便可以数据类型的形式显示接线端。图标接线端不仅可以显示前面板对象的数据类型还可以显示程序框图上前面板对象本身的类型。而数据类型接线端则可以节省程序框图空间。

注　由于图标接线端比数据类型接线端大，因此将一个数据类型接线端转换为图标接线端时，可能会在无意中覆盖其它的程序框图对象。

输入控件接线端的边框要比显示控件接线端的边框粗。箭头在前面板接线端的出现位置也表明了该接线端是输入控件还是显示控件。输入控件接线端的箭头在右边，显示控件接线端的箭头在左边。

输入控件和显示控件的数据类型

常用的输入控件 (Control) 和显示控件 (Indicator) 数据类型有浮点数 (floating-point numeric)、整数 (integer numeric)、时间标识 (time stamp)、枚举 (enumerated)、布尔 (Boolean)、字符串 (string)、数组 (array)、簇 (cluster)、路径 (path)、动态 (dynamic)、波形 (waveform)、引用句柄 (refnum) 和 I/O 名称。关于输入控件和显示控件的数据类型及其符号和用途的完整列表见 *LabVIEW Help*。

每一个接线端的颜色和符号表明了相应输入控件或显示控件的数据类型。许多数据类型都有相应的一套函数，用以对数据进行操作，如位于 **String** 子选板、用于字符串数据类型的字符串函数。

符号值

未定义或非预期数据 (Undefined or unexpected data) 会使后续的所有操作都无效。浮点数操作返回以下两种符号值来表明错误的计算或无意义的结果：

- `NaN` (not a number) 表示无效的操作产生的浮点数值，如对负数取平方根。
- `Inf`（无穷）表示超出某数据类型值域的浮点数值。例如，1 被 0 除时产生 `Inf`。

  LabVIEW 可以返回 `+Inf` 或 `-Inf`。`+Inf` 表示某数据类型的最大值，`-Inf` 表示某数据类型的最小值。

LabVIEW 不检查整数的上溢或下溢条件。

常量

常量 (constants) 是程序框图上向程序框图提供固定数据值的接线端。通用常量是那些有固定值的常量，如 pi (π) 和 Inf (∞)。用户定义的常量是那些在 VI 运行之前可以定义和编辑的常量。

绝大多数常量位于所在选板的底部或顶部。

右键单击某个 VI 或函数的输入接线端并从快捷菜单中选择 **Create» Constant** 可以创建一个用户自定义常量。

操作工具 (Operating tool) 或标注工具 (Labeling tool) 可用来单击常量并编辑常量的值。如果自动工具选择有效，双击该常量将切换到标注工具从而可以编辑常量的值。

程序框图节点

节点 (Nodes) 是程序框图上的对象，带有输入输出端，在 VI 运行时进行运算。节点相当于文本编程语言中的语句、运算、函数和子程序。LabVIEW 中含有以下类型的节点：

- **函数 (Functions)** －内置的执行元素，相当于操作符、函数或语句。
- **子 VI (SubVIs)** －用于另一个 VI 程序框图上的 VI，相当于子程序。

  关于在程序框图中使用子 VI 的更多信息参见第 7 章*创建 VI 和子 VI* 中的*创建子 VI* 一节。
- **Express VI** －协助常规测量任务的子 VI。Express VI 是在配置对话框中配置的。

  关于使用 Express VI 的更多信息见第本章 *Express VI* 一节。
- **结构 (Structures)** －执行控制元素，如 For 循环 (For Loops)、While 循环 (While Loops)、条件结构 (Case structures)、平铺式和层叠式顺序结构 (Flat and Stacked Sequence structures)、定时结构 (Timed structures) 和事件结构 (Event structures)。

  关于使用结构的更多信息参见第八章*循环和结构*。

关于程序框图节点的完整列表请参见 *LabVIEW Help*。

多态 VI 和函数

多态 VI 和函数会根据输入数据类型的不同自动调整数据类型。绝大多数的 LabVIEW 结构为多态，一些 VI 和函数也为多态。

函数多态性 (polymorphic) 的程度各不相同－可以是全部或部分输入多态，也可以是完全没有多态输入。有一些函数输入可接收数值或布尔值。有一些函数输入可接收数值或字符串。有一些函数输入不仅接收标量数值还接收数值数组，数值簇或数值簇数组等等。还有一些函数输入仅仅接收一维数

组，虽然该数组元素可以是任意数据类型。另外一些函数输入则接收所有数据类型，包括复数值。

关于数组和簇的更多信息参见第 9 章*使用字符串、数组和簇将数据分组*中的*使用数组和簇将数据分组*一节。

函数概述

函数 (Functions) 是 LabVIEW 中最基本的操作元素。**函数**选板上的函数图标具有淡黄色背景色和黑色前景色。函数没有前面板或程序框图，但具有接线器 (connector panes)。用户不能打开或编辑函数。

为接线端添加函数

某些函数接线端的数目可以改变。比如，要创建一个含有十个元素的数组，就必须向创建数组 (Build Array) 函数添加十个接线端。

使用定位工具 (Positioning tool) 分别向上或向下拖动函数的顶部或底部可以为函数添加接线端。定位工具也用于删除函数的接线端，但不能删除已经连好线的接线端。要删除这些接线端必须先删除已存在的连线。

关于连接程序框图对象的更多信息参见本章的*使用连线连接程序框图各对象*一节。

内置 VI 和函数

函数选板还包含 LabVIEW 自带的 VI。用户可以在应用程序中以子 VI 的形式使用这些 VI 和函数，从而减少开发时间。单击**函数**选板中的 **View** 按钮，并从快捷菜单中选择 **Always Visible Categories»Show All Categories**，可以在**函数**选板中显示所有类别。

关于使用内置 VI 和函数的更多信息参见第 7 章*创建 VI 和子 VI* 中的*使用内置 VI 和函数*一节。

关于所有内置 VI 和函数的详细信息见 *LabVIEW Help*。

Express VI

Express VI 用于完成常规测量任务。由于 Express VI 可以在对话框内配置，它在所有节点中需要的连线数最少。 Express VI 的输入输出取决于它的配置。在程序框图上 Express VI 以可扩展节点的形式出现，显示为蓝色背景图标。

关于使用 Express VI 的更多信息见 *《LabVIEW 入门》(Getting Started with LabVIEW)*。

使用连线连接程序框图各对象

LabVIEW 通过连线在程序框图对象之间传递数据。每根连线只有一个数据源，但可以与多个读取该数据的 VI 和函数连接，这与在文本编程语言中传递必要参数相似。必须为所有必需连接的程序框图接线端连线。否则， VI 是断开的并且不能运行。打开 **Context Help** 窗口可以知道某个程序框图节点的哪些接线端必需连接。必需连接接线端的标签在 **Context Help** 窗口显示为粗体字。

关于断开的 VI 的更多信息参见第 6 章*运行和调试 VI* 中的*纠正断开的 VI* 一节。

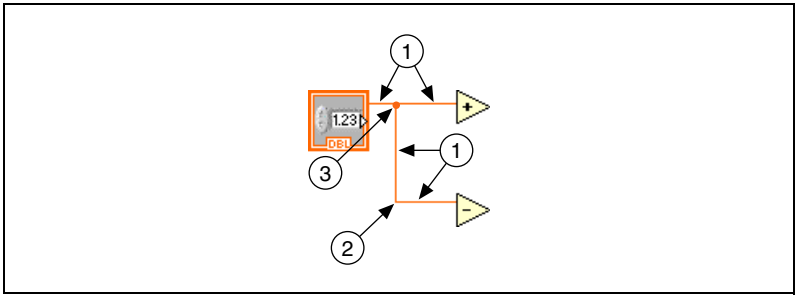
连线的外观和结构

随着数据类型的不同，连线也有不同的颜色、式样和粗细，这与接线端用不同颜色和符号来表示相应输入控件或显示控件的数据类型相似。断开的连线显示为黑色的虚线，中间有个红色的 x。出现断线的原因有很多，比如试图连接数据类型不兼容的两个对象时就会产生断线。断线中间红色 x 任意一边的箭头表明了数据流的方向，而箭头的颜色表明了流过连线的数据类型。

关于数据类型的更多信息参见本章的*输入控件和显示控件的数据类型*一节。关于数据流的更多信息参见本章的*程序框图数据流*一节。

当连线工具 (Wiring tool) 移到 VI 或函数节点上时，如接线端尚未连线，将出现接线头。它们表明了每个接线端的数据类型。同时还会出现一个提示框，列出接线端的名称。为一个接线端连好线后，该接线端的接线头在连线工具移到其节点时将不再出现。

一段连线是一单条水平或垂直的连线。连线中的弯折是两段连线交叉的地方。交叉点是两段或多段连线的相交点。一个连线分支包含所有从交叉点到交叉点、接线端到交叉点或中间没有交叉点的接线端到接线端的连线段。下图分别显示了连线段、弯折和交叉点。



1	连线段 (Segment)	2	弯折 (Bend)	3	交叉点 (Junction)

连接对象

连线工具可以手动方式为程序框图上不同节点的接线端连线。该工具的光标点到之处即为连线开始的位置。将连线工具移到某个接线端上时，该接线端会不停地闪烁。将连线工具移到某个 VI 或函数接线端时，也会出现一个提示框，里面描述了接线端的名称。为接线端连线时可能会产生断线 (broken wire)。在运行 VI 前必须纠正这些断线。

关于纠正断线的更多信息参见本章的*纠正断线*一节。

Context Help 窗口用于确定准确的连线位置。将光标移到某个 VI 或函数接线端时， **Context Help** 窗口会列出该 VI 或函数的每一个接线端。但 **Context Help** 窗口不会显示可扩展 VI 和函数的接线端，比如 Build Array 函数。点击 **Context Help** 窗口中的 **Show Optional Terminals and Full Path** 按钮可以显示接线器的可选连接接线端。

将连线交叉时，第一根连线处会出现一个小小空白，表示第一根连线位于第二根连线下面。

弯折连线

用连线连接接线端时，在垂直或水平方向移动光标可以将连线进行一次 90 度的弯折。要在多个方向弯折连线，先点击鼠标按钮一次来定位连线，然后在新的方向移动光标。这样可以重复定位连线并在新的方向移动连线。

撤消连线

要取消最后的连线定位点，按 <Shift> 键并点击程序框图上的任意位置。要中止整个连线操作，右键单击程序框图中的任意位置。

(Mac OS) 按 <Option> 键并单击。 **(Linux)** 单击鼠标的中间按钮。

自动连接对象

将选中的对象移到程序框图上其它对象的旁边时， LabVIEW 会进行暂时连线以显示有效的连线方式。放开鼠标将对象放置在程序框图上时，LabVIEW 会自动进行连线。您也可以对程序框图上已经存在的对象进行自动连线。 LabVIEW 会对最匹配的接线端进行连线，对不匹配的接线端不予连线。

用定位工具 (Positioning tool) 移动一个对象时，按空格键可以切换到自动连线模式。

选中连线

使用定位工具单击、双击或连续三次点击连线可以选择相应的连线。单击连线选中的是连线的一个直线段。双击连线选中的是连线的一个分段。连续三次点击连线选中的是整条连线。

纠正断线

断开的连线显示为黑色的虚线，中间带有红色 x。出现断线的原因有很多，比如试图连接数据类型不兼容的两个对象时就会产生断线。将连线工具移到断线上将会显示一个描述产生断线原因的提示框 (tip strip)。此时 **Context Help** 窗口中也会出现这样的信息。右键单击该连线并从快捷菜单中选择 **List Errors** 可以打开 **Error list** 窗口。要显示关于连线断开原因的更多信息，请单击 **Help** 按钮。

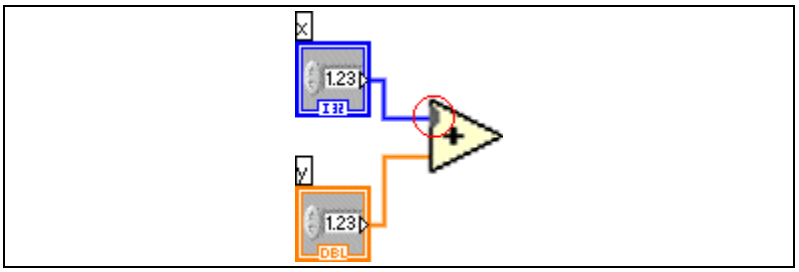
用定位工具连续三次点击连线并按 <Delete> 键可以删除断线。还可以右键单击连线，从快捷菜单中选择如 **Delete Wire Branch**、 **Create Wire Branch**、 **Remove Loose Ends**、 **Clean Up Wire**、 **Change to Control**、 **Change to Indicator**、 **Enable Indexing at Source** 和 **Disable Indexing at Source** 等选项。这些选项会随着产生断线的原因的不同而不同。

选择 **Edit»Remove Broken Wires** 或按 <Ctrl-B> 键，可以清除所有断线。**(Mac OS)** 按 <Command-B> 键。 **(Linux)** 按 <Meta-B> 键。

注意　应在清除所有断线时使用警告。有时在没有完成程序框图连线时也会产生断线。

强制转换点

将两个不同的数值数据类型连接在一起的时候，程序框图节点上会出现强制转换点 (Coercion Dots) 以示警告。强制转换点表示 LabVIEW 已经将传递给节点的数值转换成了不同的表示法。例如， Add 函数需要两个双精度浮点数输入。如果其中的一个输入为整数，该加法函数上就会出现一个强制转换点，如下图所示。



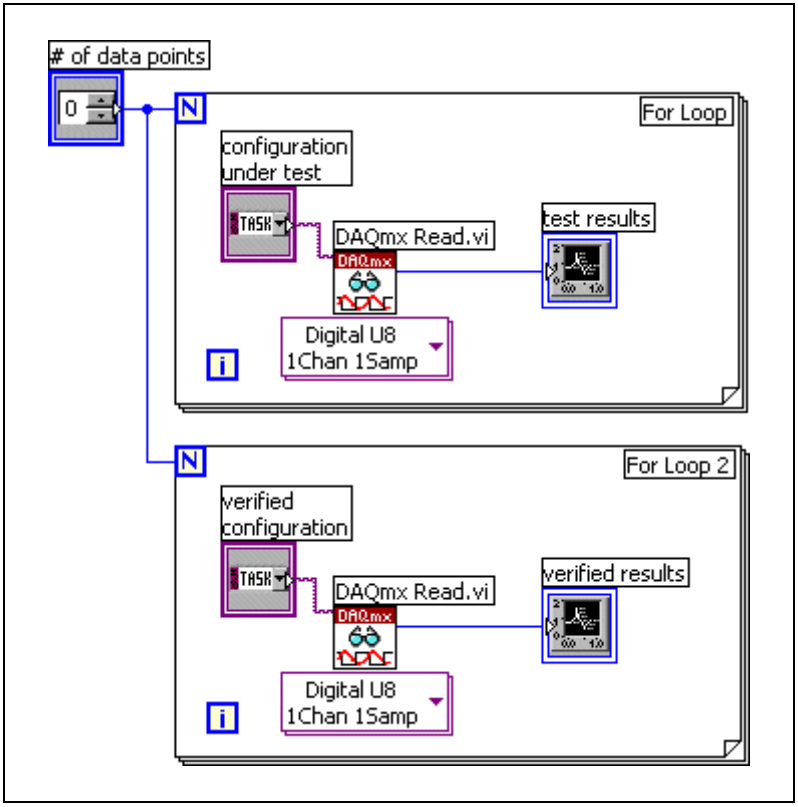
强制转换点会使 VI 消耗更多的内存，并增加其运行时间。因此，在所创建的 VI 中应尽量保持数据类型一致。

程序框图数据流

LabVIEW 按照数据流 (dataflow) 模式运行 VI。当所有需要的输入都存在时，程序框图节点将运行。节点在运行时产生输出端数据并将该数据传送给数据流路径中的下一个节点。数据流经节点的动作决定了程序框图上 VI 和函数的执行顺序。

Visual Basic、C++、JAVA 以及绝大多数其它文本编程语言都遵循程序执行的控制流模式。在控制流中，程序元素的先后顺序决定了程序的执行顺序。

在 LabVIEW 中，数据流代替命令的先后顺序决定了程序框图元素的执行顺序。因此可以创建具有并行操作的程序框图。例如，可以同时运行两个 For 循环 (For Loops)，并在前面板上显示其结果，如以下程序框图所示。



数据依赖性 (Data Dependency) 和人工数据依赖性 (Artificial Data Dependency)

控制流执行模式是由指令驱动的。而数据流执行模式是由数据驱动的，或称为是依赖于数据的。从其它节点接收数据的节点总是在其它节点执行完以后才开始执行。

没有连线的程序框图节点可以任意顺序执行。当自然的数据依赖关系不存在时，可以使用数据流参数 (flow-through parameters) 控制执行顺序。当数据流参数不可用时，可以使用顺序结构控制执行顺序。

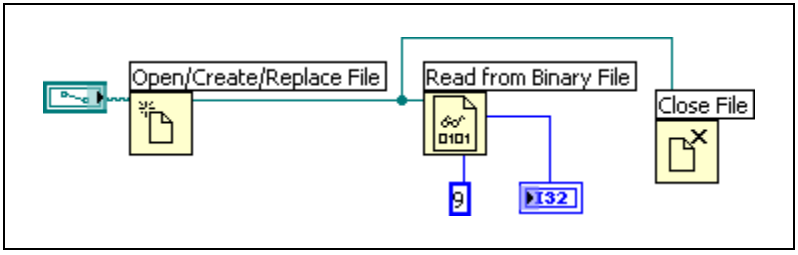
关于数据流参数的更多信息见本章*数据流参数*一节。关于顺序结构的更多信息见第 8 章*循环和结构*中的*顺序结构*一节。

您也可以创建人工数据依赖性，此时，接收节点并非真正使用接收到的数据，而是利用数据的到来触发该节点的执行。关于使用人工数据依赖性的范例见 `labview\examples\general\structs.llb` 中的 Timing Template (data dep) VI。

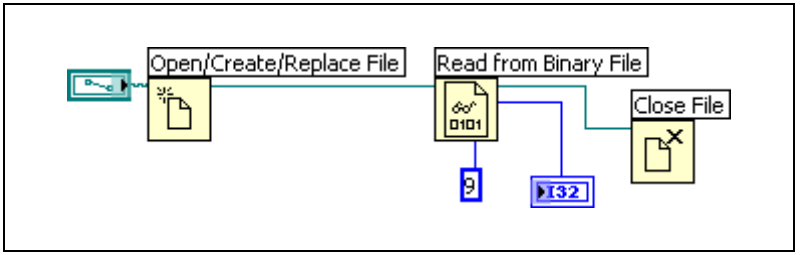
数据依赖性不存在

当数据依赖性不存在时，不要简单地认为程序的执行顺序是从左到右，自顶向下的。需要触发事件时，一定要通过数据流连线明确确定事件顺序。

在下面的程序框图中，Read from Binary File 函数和 Close File 函数之间不存在数据依赖性，因为 Read from Binary File 函数没有与 Close File 函数相连。该范例可能不会按所期望的顺序执行，因为无法确定哪个函数会先执行。如果 Close File 先运行，那么 Read from Binary File 函数将不会执行。



下面的程序框图通过将 Read from Binary File 函数的输出连接到 Close File 函数而建立了数据依赖关系。 Close File 函数只有在接收到 Read from Binary File 函数的输出后才能执行。



数据流参数

数据流参数 (Flow-Through Parameters) 通常为引用句柄 (refnum) 或错误簇 (error cluster)，它返回的是与相应的输入参数相同的值。当自然的数据依赖性不存在时，数据流参数可用来控制执行顺序。把要执行的第一个节点的数据流输出连接到要执行的下一个节点的相应输入，可以创建人工数据依赖性。如果没有这些数据流参数，必须使用顺序结构来确保数据操作按期望的顺序执行。

关于 error I/O 的更多信息请参见第 6 章*运行和调试 VI* 中的*错误处理*一节。关于顺序结构的更多信息，见第 8 章*循环和结构*中的*顺序结构*一节。

数据流和内存管理

相对于控制流执行模式，数据流执行模式使得内存管理更为简单。在 LabVIEW 中，无需为变量分配内存或给变量赋值。只需创建带有连线的程序框图以表示数据的传输就可以了。

生成数据的 VI 和函数会自动为该数据分配内存。当该 VI 或函数不再使用该数据时， LabVIEW 将释放相关内存。向数组或字符串添加新数据时，LabVIEW 将分配足够的内存来管理这些新数据。

设计程序框图

请使用下述指导方针设计程序框图：

- 请使用从左到右、自顶向下的布局模式。尽管程序框图元素的位置不会决定执行顺序，但应避免从右向左的连线方式，以使程序框图显得有结构，有条理，并且易于理解。在 LabVIEW 中，只有连线和结构才能决定执行顺序。
- 不要创建占用多于一个或两个屏幕的程序框图。如果程序框图又大又复杂，将会给理解或调试带来困难。
- 确定程序框图中是否有一些组成部分可以在其它 VI 中重复使用，或者程序框图的某一部分是否可以组合成一个逻辑单元。如果有，请将该程序框图分成几个执行特定任务的子 VI。使用子 VI 有利于管理变化，能够快速对程序框图进行调试。

关于子 VI 的更多信息参见第 7 章*创建 VI 和子 VI* 中的*创建子 VI* 一节。

- 使用错误处理 VI，函数和参数来管理程序框图上的错误。

  关于错误处理的更多信息参见第 6 章*运行和调试 VI* 中的*处理错误*一节。
- 避免在结构边框下面或交叉对象之间进行连线，因为 LabVIEW 可能会隐藏这些连线的部分线段。
- 不要将对象放置在连线上面。将接线端或图标放置在连线上面会使人感觉好像有连接存在但实际上并不存在。
- 应使用自由标签对程序框图上的代码进行注释。

  关于使用自由标签的更多信息参见第 4 章*创建前面板*中的*添加标签*一节。

6

运行和调试 VI

要运行 VI，必须得用正确的接线端数据类型连接所有的子 VI、函数和结构。有时 VI 并不会按预期的方式产生数据或运行。 LabVIEW 可以通过程序框图的组织形式或流经程序框图的数据找到问题所在。

运行 VI

运行 VI 便可以执行为 VI 所设计的操作。如果工具栏上的 **Run** 按钮是如下图所示的白色实心箭头，那么可以运行 VI。

白色实心箭头同时也表示如果为该 VI 创建接线器，则可以将其作为子 VI 使用。

关于创建接线器的更多信息请参见第 7 章*创建 VI 和子 VI* 中的*创建接线器*一节。

在单击了 **Run** 或 **Run Continuously** 按钮或者程序框图工具栏上的单步执行 (single-stepping) 按钮时， VI 便开始运行。当 VI 运行时， **Run** 按钮会变成如下图所示的黑色箭头，以表明该 VI 正在运行。

您不能对一个正在运行的 VI 进行编辑。

单击 **Run** 按钮， VI 只运行一次，在 VI 执行完数据流时就会停止运行。单击如下图所示的 **Run Continuously** 按钮， VI 将持续运行直到手动停止 VI 的运行为止。

单击单步执行按钮将以单步递增的方式运行 VI。

关于使用单步执行按钮调试 VI 的更多信息参见本章的*单步执行*一节。

纠正断开的 VI

如果一个 VI 不能执行，那么该 VI 是断开的或者是不可执行的。当您正在创建或编辑的 VI 存在错误时， **Run** 按钮是断开的，如下图所示。

如果在结束程序框图连线时，该按钮仍是断开的，则该 VI 是断开的，不能运行。

查找 VI 断开的原因

警告 (Warnings) 并不会阻碍 VI 的运行，仅用于帮助用户去避免 VI 中潜在的问题。但是错误会使一个 VI 断开。在运行 VI 前必须排除任何错误。

单击断开的 **Run** 按钮或选择 **View»Error List** 可以查找 VI 断开的原因。 **Error list** 列出了所有的错误。 **Items with errors** 部分列出了内存中所有存在错误的条目名称，如 VI 和项目库。如果两个或多个条目具有相同的名称，则这一部分会显示每一条目的特定应用程序实例。 **errors and warnings** 部分列出了在 **Items with errors** 部分选中的 VI 的错误和警告信息。 **Details** 部分描述了错误信息，有时还会建议如何纠正错误。单击 **Help** 按钮，可以显示 *LabVIEW Help* 中详细描述错误并包含纠正错误的步骤的主题。

单击 **Show Error** 按钮或双击错误描述，可以突出显示程序框图或前面板中包含错误的区域。

如下所示，如果 VI 中含有警告并且勾选了 **Error list** 窗口中的 **Show Warnings** 复选框，那么工具栏将包含如下图所示的 **Warning** 按钮。

VI 断开的常见原因

下表列出了一些在编辑 VI 时导致 VI 断开的常见原因：

- 由于数据类型不匹配或存在未连接的接线端，会导致程序框图含有断开的连线。

  关于纠正断线的更多信息参见第 5 章*创建程序框图*中的*纠正断线*一节。
- 必需连接的程序框图接线端没有连线。

  关于设定必需连接的输入端和输出端的信息参见第 5 章*创建程序框图*中的*使用连线连接程序框图各对象*一节。
- 子 VI 是断开的或将其放置到程序框图上后编辑了该子 VI 的接线器。

关于子 VI 的更多信息参见第 7 章*创建 VI 和子 VI* 中的*创建子 VI* 一节。

调试技术

如果 VI 没有断开，但得到了非预期的数据，则可使用各种调试方法识别并纠正 VI 或程序框图数据流中的问题。

执行过程高亮显示

单击如下图所示的 **Highlight Execution** 按钮可以查看程序框图的动态执行过程。

执行过程高亮显示 (Execution highlighting) 表明了程序框图上的数据通过沿着连线移动的圆点从一个节点移动到另一个节点的过程。将执行过程高亮显示与单步执行结合起来可以查看一个 VI 中数据值从一个节点移动到另一个节点的全过程。

注　执行过程高亮显示会大大降低 VI 的运行速度。

如果 **error out** 簇报告错误，该错误值会带上红色边框，出现在 **error out** 的旁边。如果没有错误发生， OK 会带上一个绿色边框，出现在 **error out** 的旁边。

关于错误簇的更多信息参见本章的*错误簇*一节。

单步执行

单步执行 (Single-step) VI 可用于查看 VI 运行时程序框图上 VI 的每次执行过程。如下图所示的这些单步执行按钮仅在单步执行模式下影响 VI 或子 VI 的运行。

单击程序框图工具栏上的 **Step Over** 或 **Step Into** 按钮可以进入单步执行模式。将鼠标移动到 **Step Over**, **Step Into** 或 **Step Out** 按钮上可以看到一个提示框，该提示框描述了单击该按钮时的下一步执行情况。可以单步执行子 VI，也可以正常运行子 VI。

如果单步执行 VI 而且同时高亮显示执行过程，如下图所示的执行符号将出现在当前运行的子 VI 的图标上。

探针工具

通用探针 (generic probe) 可用于查看流经连线的数据。右键单击连线并从快捷菜单中选择 **Custom Probe»Generic Probe**，便可以使用通用探针。

断点

如下图所示的断点工具 (Breakpoint tool) 可用于在程序框图上的 VI、节点或连线上放置一个断点，当程序运行到该处时暂停执行。

在连线上设置断点时，数据流经该连线后程序将暂停执行。在程序框图上放置一个断点，可以使程序框图上的所有节点执行以后程序暂停执行。

当 VI 在一个断点处暂停执行时， LabVIEW 会在前端显示程序框图并用选取框高亮显示含有断点的节点或连线。将光标移动到已有断点上时，断点工具光标的黑色区域会变成白色。

在程序执行中到达一个断点时， VI 会暂停执行，并且 **Pause** 按钮变为红色。这时可以采取以下措施：

- 使用单步执行按钮单步执行程序。
- 在连线上加探针以查看中间数据值。
- 改变前面板控件的值。
- 单击 **Pause** 按钮可以继续运行到下一个断点处或直到 VI 运行结束。

LabVIEW 将断点跟 VI 一起保存，但只有在 VI 运行时断点才起作用。选择 **Operate»Breakpoints** 然后单击 **Find** 按钮可以查看所有断点。

处理错误

无论您对自己创建的 VI 多么有信心，也很难预见到用户可能遇到的每一个问题。如果没有一个检查错误的机制，您所能知道的仅仅是 VI 是否在正常工作。错误检查可以告诉你为什么和在哪里产生了错误。

当进行任何形式的输入输出 (I/O) 操作时，都要考虑产生错误的可能性。几乎所有的 I/O 函数都会返回错误信息。在 VI 中包含错误检查，尤其是对于 I/O 操作，如文件、串口、仪器测量、数据采集和通讯，可以提供一个正确处理错误的机制。

默认状态下， VI 运行时 LabVIEW 会通过中断执行，高亮显示产生错误的子 VI 或函数，并显示错误对话框 (error dialog box) 来自动处理每一个错误。

选择 **File»VI Properties** 并从 **Category** 下拉菜单中选择 **Execution** 可禁用当前 VI 的自动错误处理。选择 **Tools»Options** 并从 **Category** 列表中选择 **Block Diagram**，可以禁用任何新创建的空白 VI 的自动错误处理。要禁用一个 VI 中的子 VI 或函数的自动错误处理，将它的 **error out** 参数连接到另一个子 VI 或函数的 **error in** 参数，或连接到一个 **error out** 显示控件。

您还可以选择其它的错误处理方法。例如，如果程序框图上的 I/O VI 超时，用户可能并不希望整个应用程序都停止运行并显示一个错误对话框。用户也可能希望在一段时间内 VI 进行重试。在 LabVIEW 中，可以在 VI 的程序框图上决定这些错误处理的设置。

位于 **Dialog & User Interface** 选板上的 LabVIEW 错误处理 VI 和函数以及大多数 VI 和函数的 **error in** 和 **error out** 参数可用于管理错误。例如，如果 LabVIEW 遇到了错误，可以在不同类型的对话框中显示错误信息。将错误处理和调试工具结合起来使用可以发现和管理错误。

VI 和函数通过以下任意一种方式返回错误－数值错误代码或错误簇。通常，函数使用数值错误代码，而 VI 使用包含错误输入输出的错误簇。

LabVIEW 中的错误处理遵循数据流模式。错误信息就像数据值一样流经 VI。将错误信息从 VI 的开始一直连接到结尾。将 error handler VI 连在一个 VI 之后可以确定该 VI 的运行是否未出差错。所使用或创建的每一个 VI 中的 **error in** 和 **error out** 簇可用于在 VI 中传递错误信息。错误簇属数据流参数 (flow-through parameters)。

关于数据流参数的更多信息参见第 5 章*创建程序框图*中的*数据流参数*一节。

当 VI 运行时， LabVIEW 会在每个执行节点检测错误。如果 LabVIEW 没有发现任何错误，该节点将正常执行。如果 LabVIEW 检测到错误，该节点会将错误传递到下一个节点而不执行那一部分代码。下面的节点也同样如此处理，直到最后一个节点。在执行流结束时， LabVIEW 报告错误。

错误簇

error in 和 **error out** 簇包括以下信息：

- **状态 (status)** 是一个布尔值，错误产生时报告 TRUE。
- **错误代码 (code)** 是一个 32 位有符号整数，它以数值方式识别错误。一个非零错误代码和 FALSE **status** 相结合可以表示警告但不是错误。
- **错误源 (source)** 是用于识别错误发生位置的字符串。

部分支持布尔数据的 VI、函数和结构也可识别错误簇。例如，可以将一个错误簇连接到 Select、Quit LabVIEW 或 Stop 函数的布尔输入端。如果错误发生，错误簇将 TRUE 值传递给该函数。

关于簇的更多信息参见第 9 章 *使用字符串、数组和簇将数据分组*中的*簇*一节。

使用 While 循环处理错误

可以将错误簇连接到 While 循环的条件接线端以停止 While 循环 (While Loops) 的运行。将错误簇连接到条件接线端上时，只有错误簇 **status** 参数的 TRUE 或 FALSE 值会传递到接线端。当错误发生时， While 循环停止执行。

将一个错误簇连接到条件接线端上时，快捷菜单项 **Stop if True** 和 **Continue if True** 将变为 **Stop on Error** 和 **Continue while Error**。

关于使用 While 循环的更多信息见第 8 章*循环和结构*中的 *While 循环*一节。

使用条件结构处理错误

将错误簇连接到条件结构 (Case Structures) 的选择器接线端时，分支选择器 (case selector) 标签将显示两个条件分支— `Error` 和 `No Error`，并且条件结构的边框会改变颜色— `Error` 时为红色， `No Error` 时为绿色。如果发生错误，条件结构将执行 `Error` 子程序框图。

关于使用条件结构的更多信息参见第 8 章*循环和结构*中的*条件结构*一节。

将子 VI 和 Error Handling template VI 相结合可以创建带有错误处理条件结构的 VI。

关于 VI 模板的更多信息见第 1 章 *LabVIEW 简介*中的 *LabVIEW VI 模板*一节。

7

创建 VI 和子 VI

VI 既是用户接口，也是程序中一项常用操作。了解了如何创建前面板 (front panel) 和程序框图 (block diagram) 后，您将可以开始创建新的 VI 和子 VI 并在此基础上进行自定义。

查找范例

在创建新 VI 之前，可考虑在 VI 范例中查找满足要求的 VI。选择 **Help»Find Examples** 打开 NI Example Finder。如果未找到合适的 VI 范例，请在 **New** 对话框中打开 VI 模板，模板中包含一些**函数 (Functions)** 选板中的内置 VI 和函数。

关于 VI 模板的详细信息见第 1 章 *LabVIEW 简介*中的 *LabVIEW VI 模板*、*VI 范例和工具*一节。

使用内置 VI 和函数

LabVIEW 包含多个 VI 和函数可以用来创建特定的应用程序，如数据采集 VI 和函数、访问其它 VI 的 VI、以及与其它应用程序通信的 VI。将这些 VI 作为子 VI 应用到自己的程序中可为您缩短开发时间。在创建新 VI 之前，可考虑在**函数**选板中查找类似 VI 和函数，以便在现有 VI 的基础上创建 VI。

创建子 VI

创建好一个 VI 后，可将其用于另一个 VI 中。在其它 VI 中被调用的 VI 称为子 VI。为使子 VI 在其它 VI 中重复使用，应为子 VI 创建接线器和图标。

一个子 VI 节点相当于文本编程语言中的一个子程序调用。节点并不是子 VI 本身，就像一个程序中的子程序调用指令并不是子程序本身一样。具有几个相同子 VI 节点的程序框图会重复几次调用该子 VI。

子 VI 的控件和函数从调用该 VI 的程序框图中接收和返回数据。单击**函数**选板上的 **Select a VI** 图标，找到所需 VI 并双击，然后将该 VI 放置在程序框图上就可以创建一个调用该 VI 的子 VI。

用操作 (Operating) 或定位 (Positioning) 工具双击程序框图上的子 VI 可以对该子 VI 进行编辑。保存子 VI 时，子 VI 的修改将影响到所有对该子 VI 的调用，而不仅仅是当前实例。

创建图标

每一个 VI 都在前面板和程序框图窗口的右上角有一个图标 (icon)，如图所示。

图标是 VI 的图形化表示，可包含文字、图形或图文组合。如果将一个 VI 当作子 VI 使用，程序框图上将显示代表该子 VI 的图标，

默认图标包含一个数字，表明自从运行 LabVIEW 以来已经打开第几个新 VI。右键单击前面板或程序框图右上角的图标并从快捷菜单中选择 **Edit Icon**，或双击前面板右上角的图标可以创建自定义图标来替换默认的图标。

也可以从文件系统的任何地方拖动一个图形并放置在前面板或程序框图的右上角。LabVIEW 会将该图形转换为 32 × 32 像素的图标。

关于 VI 图标的标准图形的详细信息，请登陆 National Instruments 网站 `ni.com/info` 并输入信息代码 `expnr7` 进行查询。

设置接线器

要将一个 VI 当作子 VI 使用，必须创建一个如图所示的接线器 (connector pane)。

接线器是各个对应于该 VI 输入控件和显示控件的接线端的集合，类似于文本编程语言中函数调用的参数列表。接线器标明了可与该 VI 连接的输入和输出端，以便将该 VI 作为子 VI 调用。接线器在其输入端接收数据，然后通过前面板的控件传输至程序框图的代码中，并从前面板的显示控件中接收运算结果传输至其输出端。

将前面板输入控件或显示控件分配给接线器的每一个接线端，这样可以确定数据的连接。用右键单击前面板窗口右上角的图标，并从快捷菜单中选择 **Show Connector** 显示接线器。图标的位置上将显示接线器。第一次打开接线器时，可看到接线器图案。右键单击接线器并从快捷菜单中选择 **Patterns** 可以为 VI 选择不同的接线器样式。

接线器上的每一个窗格代表一个接线端。各个窗格用于进行输入输出分配。默认的接线器模式为 4 x 2 x 2 x 4。可使用默认模式，保留多余的接线端，当需要为 VI 添加新的输入或输出端时再进行连接。

接线器中最多可设置 28 个接线端。如果前面板上希望通过编程使用的输入控件和显示控件的数目多于 28 个，这时可以将其中的一些对象分组成为一个簇，然后将该簇分配给接线器上的某个接线端。

关于使用簇为数据分组的详细信息参见第 9 章 *使用字符串、数组和簇将数据分组*中的*簇*一节。

右键单击接线器并从快捷菜单中选择 **Patterns** 可以为 VI 选择不同的接线端模式。例如：可以选择一个带有备用接线端的接线器模式。备用的接线端可以空置，到需要时再连接。这种灵活性可以减少接线器窗口的改变对 VI 的层次结构的影响。

选中部分程序框图创建子 VI

用定位工具选择希望重复使用的部分程序框图并选择 **Edit**»**Create SubVI** 可以将被选中的程序框图转换成子 VI。新的子 VI 图标将替换被选择的部分程序框图。LabVIEW 可为新的子 VI 创建输入控件和显示控件，并根据所选输入控件和显示控件的数目自动配置接线器，将子 VI 与现有的连线相接。

选中部分程序框图创建子 VI 是十分方便的，但仍需要仔细规划以便分清 VI 的逻辑层次结构。需仔细考虑选择哪些对象以避免改变 VI 的功能。

设计子 VI 的前面板

根据在接线器中的位置在前面板中放置输入控件和显示控件。请将输入控件放在前面板的左边，显示控件放在右边。在前面板中请将 **error in** 簇放在左下角位置， **error out** 簇放在右下角。

关于设置连接器的详细信息见本章的*设置连接器*部分。

查看 VI 的层次结构

VI Hierarchy 窗口以图形化的方式显示所有打开的 LabVIEW 项目和终端，以及内存中所有 VI 的调用结构，包括自定义类型、全局变量等。选择 **View**»**VI Hierarchy** 打开 **VI Hierarchy** 窗口。该窗口用于查看 VI 内存中所有子 VI 和节点并可搜索 VI。

关于 LabVIEW 项目的详细信息见第 3 章 *LabVIEW 编程环境*中的*项目浏览器窗口*一节。

VI Hierarchy 窗口顶部是上层图标，代表 LabVIEW 主程序实例，下面显示的是所有未包括在该项目或项目程序实例中的展开的 VI。如果在 LabVIEW 中添加了一个项目，**VI Hierarchy** 窗口中将显示代表该项目的上层图标。所有被添加到该项目中的对象均位于该上层图标之下。

将鼠标光标移至 **VI Hierarchy** 窗口的对象上，下方将显示 VI 的名称。使用定位工具拖动 **VI Hierarchy** 窗口中的 VI，可将其放到另一个 VI 的程序框图中作为子 VI 使用。也可以选择和拷贝一个或多个节点，将它们粘贴到其它程序框图上。在 **VI Hierarchy** 窗口中双击某个 VI 可以显示该 VI 的前面板。

含有子 VI 的 VI 在其底部边框上有个箭头按钮。单击该箭头按钮可以显示或隐藏它的子 VI。如果所有子 VI 均为隐藏状态则箭头显示为红色。如果所有子 VI 都已出现在结构图中则箭头显示为黑色。

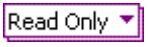
多态 VI

多态 VI(Polymorphic VI) 的单个输入或输出接线端可以接收不同的数据类型。多态 VI 是具有相同接线器模式的子 VI 的集合。该集合中的各个 VI 为多态 VI 中的一个实例。

例如， Read Key VI 就是一个多态 VI。其**默认**接线端可以接收的数据类型有布尔、双精度浮点数、 32 位有符号整数、路径、字符串或 32 位无符号整数。

对于绝大多数多态 VI，连接到多态 VI 输入端的数据类型将决定应该使用的程序实例。如果多态 VI 中没有任何实例与其连接的数据类型兼容，则会出现断线。如果连接到多态 VI 的连线不能决定使用哪个实例，则必须手动选择。如果为一个多态 VI 手动选择实例，该 VI 将不再是多态 VI，因为它将只接收和返回所选的数据类型。

如需手动选择实例，请右键单击多态 VI，在快捷菜单中选择 **Select Type**，然后选择所需实例。同时也可使用操作 （Operating）工具单击多态 VI 选择器 (polymorphic VI selector)，然后从快捷菜单中选择实例。如下所示。



用右键单击程序框图上的多态 VI 并从快捷菜单中选择 **Visible Items»Polymorphic VI Selector**，将显示该选择器。要使多态 VI 重新接收所有的可处理数据类型，用右键单击多态 VI 并从快捷菜单中选择 **Select Type»Automatic** 或使用操作工具单击多态 VI 选择器并从快捷菜单中选择 **Automatic**。

对不同的数据类型执行同样的操作时可以考虑创建多态 VI。

注　只有在 LabVIEW 专业版开发系统 (LabVIEW Professional Development System) 中才可以创建和编辑多态 VI。

例如，如果要对单精度浮点数、数值数组或波形执行同样的数学运算，需要创建三个独立的 VI：Compute Number、Compute Array 和 Compute Waveform VI。当要执行该操作时，根据不同的输入数据类型，选择其中的一个 VI 放置在程序框图上。

创建和使用单个多态 VI 可以代替手动选择 VI。

保存 VI

选择 **File»Save** 保存 VI。保存 VI 时请使用有意义的名称为 VI 命名以便于识别。可以将 VI 保存为 LabVIEW 的前期版本，以便升级 LabVIEW，以及必要时在两个不同的 LabVIEW 版本中维护这些 VI。

VI 命名

保存 VI 时，请使用有意义的名称。有意义的名称便于识别 VI 并了解该如何使用 VI，如：`Temperature Monitor.vi` 和 `Serial Write & Read.vi`。如果使用意义不明确的名称，将会引起混淆，尤其是在保存了好几个 VI 之后更难以识别，如：`VI#1.vi`。

同时应考虑到用户是否可能在其它平台上使用该 VI。请不要使用一些平台上用于特殊用途的符号，如：`\:/?*<>` 和 `#`。

注　如果在同一台计算机上有同名的 VI，请将这些 VI 分门别类放在相应目录或 LLB 下，以避免 LabVIEW 在打开上层 VI 时用错子 VI。

保存为前期版本

将 VI 保存为 LabVIEW 的前期版本，便于升级 LabVIEW，以及必要时在两个不同的 LabVIEW 版本中维护这些 VI。选择 **File»Save For Previous Version** 保存为 LabVIEW 前期版本格式。

将 VI 保存为前期版本时，LabVIEW 不是仅转换该 VI 而是转换层次结构中的所有 VI，但不包括 `labview\vi.lib` 目录下的 VI。

VI 可能经常使用 LabVIEW 前期版本中所没有的功能。这时 LabVIEW 将尽量保存该 VI 的现有功能并生成报告说明哪些是不能转换的。该报告会立即出现在 **Warning** 对话框中。单击 **OK** 确认已收到警告，并关闭对话框。单击 **Save to File** 按钮可将这些警告保存为文本文件以备今后查看。

自定义 VI

根据应用程序的要求可以对 VI 和子 VI 进行配置。例如，如果打算将一个 VI 作为要求用户输入的子 VI 使用，可以对该 VI 进行配置，使得该 VI 在每次被调用时显示其前面板。

选择 **File»VI Properties** 配置 VI 的外观和行为。使用 **VI Properties** 对话框顶部的 **Category** 下拉菜单可以从几个不同的选项中进行选择。

VI Properties 对话框包括以下选项：

- **General** －显示 VI 保存的当前路径、修订号、修订历史，以及自从该 VI 上次被保存以来所作的任何修改信息。同时可以编辑 VI 图标。
- **Documentation** －该选项用于添加 VI 说明，并链接相关帮助文件主题。

  关于文档选项的详细信息见第 12 章*编制 VI 文档和打印 VI* 中的*编制 VI 文档*一节。
- **Security** －该选项用于锁定 VI 或通过密码保护 VI。
- **Window Appearance** －该选项用于自定义 VI 的窗口外观，如窗口标题和式样。
- **Window Size** －该选项用于设置窗口的大小。
- **Execution** －该选项用于定义如何运行 VI。例如，可以将 VI 配置成在打开时立即运行或者在作为子 VI 被调用时暂停运行。
- **Editor Options** －该选项用于设置当前 VI 对齐网格的大小，改变输入控件或显示控件的式样。这里的输入控件和显示控件是指右键单击接线端并从快捷菜单中选择 **Create»Control** 或 **Create»Indicator** 所显示的控件。

  关于对齐网格的详细信息见第 4 章*创建前面板*中的*对齐和分布对象*一节。

8

循环和结构

结构是传统文本编程语言中的循环和条件语句的图形化表示。使用程序框图中的结构可对代码组进行重复操作、有条件执行或按特定顺序执行。

跟其它节点一样，结构也包含可与其它程序框图节点进行连线的接线端。当所有输入数据存在时结构会自动执行，执行结束后将数据提供给输出连线。

每个结构都含有一个特殊的可调整大小的边框用于包含根据结构规则执行的程序框图部分。结构边框中的程序框图部分称为子程序框图 (Subdiagram)。
结构中接收和输出数据的接线端称为隧道 (Tunnel)。隧道是结构边框上的连接点。

以下**结构 (Structures)** 选板中的结构可用于控制程序框图执行进程的方式：

- **For 循环 (For Loop)** －按设定的次数执行子程序框图。
- **While 循环 (While Loop)** －执行子程序框图直至条件满足。
- **条件结构 (Case structure)** －包括多个子程序框图，根据传递至该结构的输入值，每次只执行其中一个子程序框图。
- **顺序结构 (Sequence structure)** －包含一个或多个按顺序执行的子程序框图。
- **事件结构 (Event structure)** －包含一个或多个子程序框图，其中子程序框图的执行顺序取决于用户如何与 VI 进行交互操作。
- **定时结构 (Timed Structure)** －执行一个或多个包括时间限制和时间延迟的子程序框图。

右键单击结构的边框可以显示快捷菜单。

For 循环和 While 循环结构

For 循环和 While 循环可用于控制重复执行的操作。

For 循环

如下图所示， For 循环将按设定的次数执行子程序框图。

如下图所示，总数接线端 (count terminal) (输入接线端) 的值用于表示重复执行该子程序框图的次数。

通过将来自循环外部的数值连接到总数接线端的左边或顶部可以明确设定总数值，或者使用自动索引隐含地设定总数值。

关于隐含地设定总数值的详细信息见本章的*使用自动索引设置 For 循环总数值*一节。

如下图所示，计数接线端 (iteration terminal) (输出接线端) 包含已完成循环的次数。

循环计数器 (iteration count) 总是从零开始。在第一次循环期间，计数接线端返回 0。

总数和计数接线端都是 32 位有符号整数。如果将一个浮点数连接到总数接线端， LabVIEW 将对其进行舍入，并将其强制转换到 32 位有符号整数的范围内。如果将 0 或负数连接到总数接线端，该循环无法执行并在输出中包含该数据类型的默认值。

向 For 循环添加移位寄存器可以将当前循环中的数据传递到下一次循环。

关于在循环中添加移位寄存器的详细信息见本章的*移位寄存器*一节。

While 循环

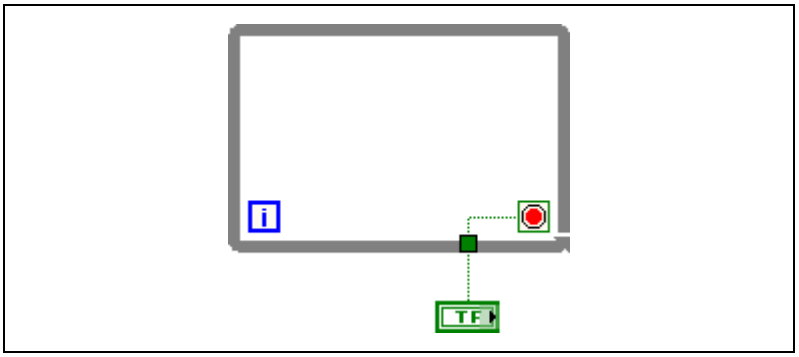
如下图所示，类似于文本编程语言中的 Do 循环或 Repeat-Until 循环，While 循环执行子程序框图直到满足某一条件。

While 循环执行子程序框图，直到条件接线端 (conditional terminal) 即输入接线端接收到某一特定的布尔值时才停止执行。如下图所示，条件接线端的默认行为和外观为 **Stop if True**。

当条件接线端为 **Stop if True** 时，While 循环将执行其子程序框图直到条件接线端接收到一个 TRUE 值。右键单击该接线端或 While 循环的边框，并选择如下图所示的 **Continue if True**，可以改变条件接线端的行为和外观。

当条件接线端为 **Continue if True** 时， While 循环将执行其子程序框图直到条件接线端接收到一个 FALSE 值。通过使用操作 (Operating) 工具单击条件接线端也可以改变条件。

如下图所示，将布尔输入控件的接线端放置在 While 循环的外部并将控件设置为 FALSE，如果条件接线端为 **Stop if True**，循环执行时会导致无限循环。如果循环外部的控件被设置为 TRUE 并且条件接线端为 **Continue if True**，也会导致无限循环。



由于输入控件的值只在循环开始前被读取一次，因此改变控件的值并不能停止无限循环。要停止一个无限循环，必须单击工具栏上的 **Abort Execution** 按钮中止整个 VI。

While 循环的条件接线端也可用于进行基本的错误处理。将错误簇 (error cluster) 连接到条件接线端时，只有错误簇中 **status** 参数的 TRUE 或 FALSE 值被传递到该接线端。并且 **Stop if True** 和 **Continue if True** 快捷菜单选项也相应改变为 **Stop if Error** 和 **Continue while Error**。

关于错误簇和错误处理的详细信息见第 6 章*运行和调试 VI* 的*错误处理*一节。

如下图所示，计数接线端 (输出接线端) 包含已完成循环的次数。

循环计数器总是从零开始。在第一次循环期间，计数接线端返回 0。

向 While 循环添加移位寄存器可以将当前循环的数据传递到下一次循环。

关于在循环中添加移位寄存器的详细信息见本章的*移位寄存器*一节。

控制定时时间

您可能需要控制进程的执行速度，比如数据绘制到图表上的速度。此时，在循环中使用等待 (Wait) 函数可以使循环在重新执行之前等待一定的时间，时间的单位为毫秒。

自动索引循环

如果将一个数组连接到 For 循环或 While 循环输入隧道，通过启用自动索引可以读取和处理数组中的每一个元素。

关于数组的详细信息见第 9 章*使用字符串、数组和簇分组数据*中的*数组*一节。

将数组连接到循环边框的输入隧道并且启用输入隧道的自动索引时，每次均有一个数组元素进入循环，并且从第一个数组元素开始。当禁用自动索引时，整个数组将一次全部传递到循环中。当自动索引数组输出隧道时，该输出数组可从每次循环中接收一个新元素。因此，自动索引的输出数组在大小上总是等于循环的次数。例如，如果循环执行 10 次，那么输出数组含有 10 个元素。如果在输出隧道上禁用自动索引，只有最后一次循环的元素被传递到程序框图上的下一个节点。

右键单击循环边框上的隧道，并从快捷菜单中选择 **Enable Indexing** 或 **Disable Indexing** 可以启用或禁用自动索引。默认情况下， While 循环已禁用自动索引。

循环边框上的方括号表示已启用自动索引。输出隧道和下一个节点之间连线的粗细也表示循环是否正在使用自动索引。当使用自动索引时，连线比较粗，因为此时连线上包含一个数组而不是一个标量。

循环可以从一维数组中索引标量元素，从二维数组中索引一维元素，以此类推。对于输出隧道情况正好相反，标量元素按自然顺序累积形成一维数组，一维数组累积形成二维数组，以此类推。

使用自动索引设置 For 循环总数值

如果将连接到 For 循环输入接线端的数组启用自动索引， LabVIEW 会将总数接线端设置为数组大小，因此用户无需为总数接线端连接数值。由于 For 循环可以每次处理数组中的一个元素，因此默认情况下， LabVIEW 对连接到 For 循环的每个数组均启用自动索引。如果不需要每次处理数组中的一个元素，可以禁用自动索引。

如果启用多个隧道的自动索引，或将数值连接至总数接线端，那么循环总数将取其中的最小值。例如，如果有两个启用自动索引的数组进入循环，并且两个数组分别含有 10 和 20 个元素，同时将值 10 连接到总数接线端，那么该循环只执行 10 次，并且第二个数组仅索引前 10 个元素。例如在一个图形上绘制两个数据源，并只需绘制前 100 个元素，这时可将值 100 连接到总数接线端。如果其中一个数据源只含有 50 个元素，那么循环将执行 50 次，并且只索引前 50 个元素。 Array Size 函数可以确定数组的大小。

自动索引 While 循环

如果自动索引一个进入 While 循环的数组， While 循环索引数组的方式与 For 循环相同。但是 While 循环只有满足特定条件时才会停止执行，因此 While 循环执行循环的次数不受该数组大小的限制。当 While 循环索引超过输入数组的大小时， LabVIEW 会将该数组元素类型的默认值输入循环。通过使用 Array Size 函数可以防止将数组默认值传递到 While 循环中。 Array Size 函数表示数组中元素的个数，将 While 循环设置为当循环次数与数组大小相同时停止执行。

注意　由于 While 循环不能提前确定输出数组的大小，因此自动索引 For 循环比自动索引 While 循环更有效。循环次数过多可能会引起系统内存溢出。

使用循环创建数组

循环不仅可以用于读取和处理数组元素，也可以通过 For 循环和 While 循环创建数组。将循环中 VI 或函数的输出连接到循环边框上。在 While 循环中，右键单击隧道并从快捷菜单中选择 **Enable Indexing**。使用 For 循环时，默认情况下已经启用自动索引。隧道的输出是一个数组，其中数组的每个元素就是每次循环结束后 VI 或函数返回的值。

关于数组的详细信息见第 9 章 *使用字符串、数组和簇分组数据*中的*数组*一节。

关于创建数组的范例见 `labview\examples\general\arrays.llb`。

循环中的移位寄存器和反馈节点

在 For 循环或 While 循环中，移位寄存器 (Shift Register) 或反馈节点 (Feedback Node) 可以将循环的值传递到下一次循环中。

移位寄存器

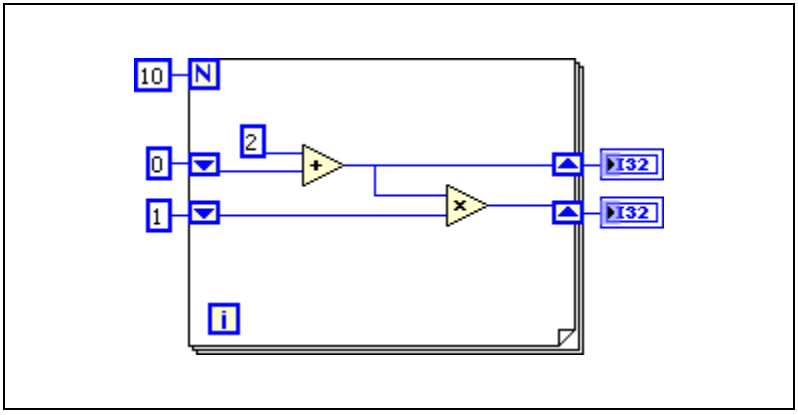
移位寄存器可用于将上一次循环的值传递到下一次循环中。如下图所示，移位寄存器以一对接线端的形式出现，并且以相反的方向分别位于循环两侧的垂直边框上。

右侧接线端含有一个向上的箭头，用于存储每次循环结束时的数据。LabVIEW 可将连接到右侧寄存器的数据传递到下一次循环中。循环执行后，右侧接线端将返回移位寄存器所保存的值。

右键单击循环的左侧或右侧边框，并从快捷菜单中选择 **Add Shift Register** 可以创建一个移位寄存器。

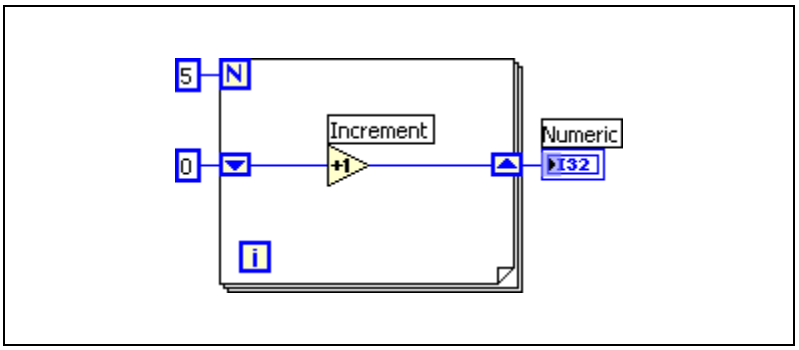
移位寄存器可以传递任何数据类型，并自动与连接到移位寄存器的第一个对象所属的数据类型保持一致。连接到各个移位寄存器接线端的数据必须属于同一数据类型。

在循环中可以添加多个移位寄存器。如下图所示，如果在循环中多个操作需使用之前循环的值，可以通过多个移位寄存器保存结构中不同操作的数据值。



初始化移位寄存器

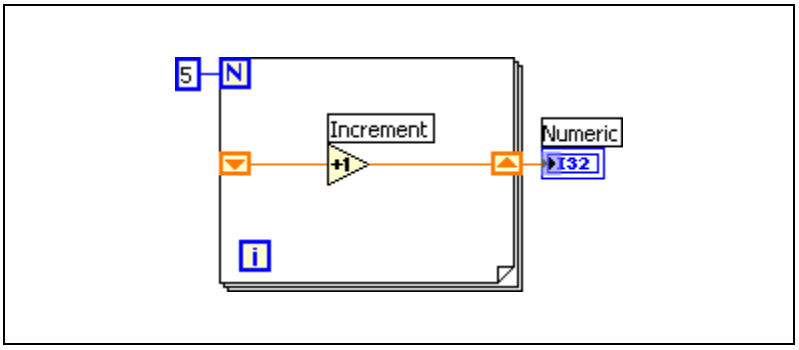
初始化移位寄存器后，VI 运行时可以将移位寄存器传递给第一次循环的值进行复位。如下图所示，将输入控件或常数连接到循环左侧的移位寄存器接线端，就可以对移位寄存器进行初始化。



上图中的 For 循环将执行 5 次，并且每次都将移位寄存器中的值加 1。 For 循环完成 5 次循环后，移位寄存器会将最终值 (5) 传递给显示控件并结束 VI 运行。每次执行该 VI，移位寄存器的初始值均为 0。

如果没有初始化移位寄存器，循环将使用最后一次执行时写入该寄存器的值，或在循环未执行过的情况下使用该数据类型的默认值。

使用没有进行初始化的移位寄存器还可以保留 VI 连续执行之间的状态信息。下图所示即为没有进行初始化的移位寄存器。

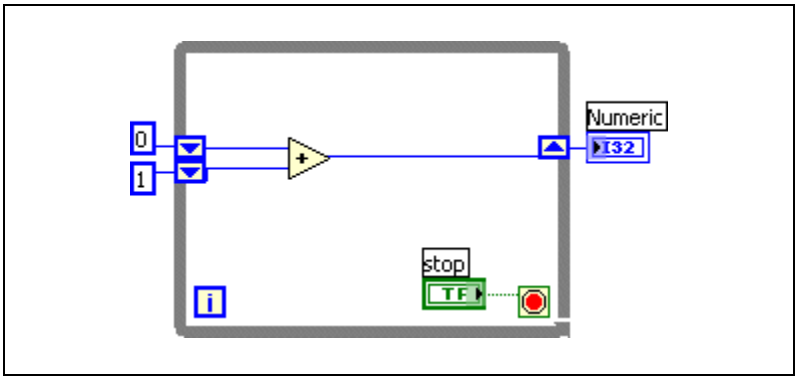


上图中的 For 循环将执行 5 次，并且每次都将移位寄存器中的值加 1。第一次运行 VI 时，移位寄存器的初始值为 0(即为 32 位整数的默认值)。 For 循环完成 5 次循环后，移位寄存器会将最终值 (5) 传递给显示控件并结束 VI 运行。而第二次运行该 VI 时，移位寄存器的初始值是上一次循环所保存的最终值 5。 For 循环执行 5 次后，移位寄存器会将最终值 (10) 传递给显示控件。如果再次执行该 VI，移位寄存器的初始值是 10，依此类推。关闭 VI 之前，未进行初始化的移位寄存器将保留上一次循环的值。

栈移位寄存器

通过栈移位寄存器 (Stacked Shift Register) 可以访问以前多次循环的数据。栈移位寄存器可以保存以前多次循环的值，并将值传递到下一次循环中。如需创建栈移位寄存器，右键单击左侧的接线端并从快捷菜单中选择 **Add Element**。

如下图所示，栈移位寄存器只位于循环左侧，因为右侧接线端只用于将当前循环的数据传递给下一次循环。



如果在上图中给左侧接线端添加另一个元素，上两次循环的值将传递至下一次循环中，其中最近一次循环的值保存在上面的寄存器中，而上一次循环传递给寄存器的值则保存在下面的接线端中。

反馈节点

如下图所示，在 For 循环或 While 循环中将节点或一组节点的输出连接到相同的节点或一组节点的输入时会自动出现反馈节点。

也可在**函数**选板上选择反馈节点并将其置于 For 循环或 While 循环中。使用反馈节点可以避免在循环中出现长连线。

右键单击反馈节点并从快捷菜单中选择 **Initializer Terminal**，在循环边框添加初始化接线端 (initializer terminal) 用于初始化循环。在**函数**选板中选择反馈节点或将已经初始化的移位寄存器转换为反馈节点，循环会自动生成初始化接线端。初始化反馈节点将使循环第一次执行时反馈节点所传递的初始值复位。如果没有初始化反馈节点，那么反馈节点将传递最后一次写入该节点的值，或者在循环从未执行过的情况下传递其数据类型的默认值。如果初始化接线端的输入端没有连接数值，每次 VI 运行时，反馈节点的初始输入都将是上一次执行的最终值。

通过右键单击移位寄存器并从快捷菜单中选择 **Replace with Feedback Node**，可以将移位寄存器替换为反馈节点。右键单击反馈节点并从快捷菜单中选择 **Replace with Shift Register**，可以将反馈节点替换为移位寄存器。

循环中的默认数据

移位寄存器没有进行初始化时， While 循环将产生默认数据。

如果将 0 连接到 For 循环的总数接线端，或者将空数组作为输入连接到 For 循环并且启用自动索引时， For 循环将产生默认数据。该循环将不会执行，任何禁用自动索引的输出隧道将包含该隧道数据类型的默认值。不管循环是否执行，移位寄存器都可用于在循环之间可以传输数据值。

关于数据类型的默认值的详细信息见 *LabVIEW 快速参考指南*。

条件、顺序和事件结构

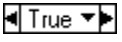
条件 (Case) 结构、层叠式 (Stacked Sequence) 和平铺式顺序 (Flat Sequence) 结构以及事件 (Event) 结构包含多个子程序框图。条件结构根据传递给该结构的输入值可执行相应的子程序框图；层叠式和平铺式顺序结构将按顺序执行所有的子程序框图；而事件结构根据用户与 VI 的交互情况执行其中的某一个子程序框图。

条件结构

如下图所示，条件结构包括两个或多个子程序框图或条件分支。

每次只能显示一个子程序框图，并且结构每次只执行一个条件分支。输入值将决定需要执行的子程序框图。条件结构类似于文本编程语言中的 switch 语句或 if...then...else 语句。

如下图所示，条件结构顶部的分支选择器 (Case Selector) 标签是由中间各个条件分支对应的选择器值的名称以及两边的递减和递增箭头组成。



单击递减和递增箭头可以滚动浏览已有条件分支。也可以单击条件分支名称旁边的向下箭头，并在下拉菜单中选择一个条件分支。

将一个输入值或选择器连接到如下图所示的选择器接线端即可以选择需执行的条件分支。

选择器接线端可以支持的数据类型有整数、布尔值、字符串或枚举类型的值。可以将选择器接线端置于条件结构左边框的任意位置。如果选择器接线端的数据类型是布尔型，该结构将包括 TRUE 和 FALSE 条件分支。如果选择器接线端的数据类型为整数、字符串或枚举类型，该结构可以使用任意个条件分支。

指定条件结构的默认条件分支以处理超出范围的数据值，否则必须明确列出所有可能的输入值。例如，如果选择器的数据类型是整数，并且已指定 1、2 和 3 条件分支，必须指定一个默认条件分支以便在输入数据为 4 或任何其它有效的整数值时执行。

分支选择器值和数据类型

在分支选择器的标签中可以输入单个值或数据值的列表和范围。如使用列表，使用逗号隔开数据值；如使用数值范围，指定一个类似 10..20 的范围可用于表示 10 到 20 之间的所有数字 (包括 10 和 20)。也可以使用开集范围，例如， ..100 表示所有小于或等于 100 的数， 100.. 表示所有大于或等于 100 的数。同时也可以将列表和范围结合起来使用，如 ..5, 6, 7..10, 12, 13, 14。在同一个分支选择器标签中输入包含重叠范围的数据值时，条件结构会以更紧凑的形式显示该标签。例如上述例子将被重新显示为 **..10, 12..14**。如使用字符串范围，范围 a..c 包括 a 和 b，但不包括 c。 a..c,c 才同时包括结束值 c。

如果输入选择器的值与连接到选择器接线端的对象不是同一数据类型，那么该值将变为红色以表示在该结构执行之前必须删除或编辑该值，并且 VI 不能运行。同样由于浮点算术运算可能存在四舍五入错误，因此不能将浮点数作为分支选择器值。如果将一个浮点值连接到条件分支， LabVIEW 将对其进行舍入并转换为最接近的整数。如果在分支选择器标签中输入浮点值，数值将变成红色以表示在执行结构前必须删除或编辑该值。

输入和输出隧道

条件结构可以创建多个输入输出隧道。所有输入都可供条件分支选用，但条件分支不需要使用所有输入。每一个条件分支必须定义各自的输出隧道。在某一个条件分支中创建一个输出隧道时，隧道将出现在所有其它条件分支边框的同一位置上。只要有一个输出隧道没有连线，该结构上的所有输出隧道将以白色正方形的形式出现。每个条件分支的同一输出隧道可以定义不同的数据源，但数据类型必须相互兼容。右键单击输出隧道并从快捷菜单中选择 **Use Default If Unwired**，所有未连接的隧道将使用隧道数据类型的默认值。

使用条件结构进行错误处理

将错误簇连接到条件结构的选择器接线端时，选择器标签将显示两个条件分支－ Error 和 No Error，并且也会相应改变条件结构的边框颜色－ Error 和 No Error 分别为红色和绿色。如果发生错误，条件结构将执行 Error 子程序框图。

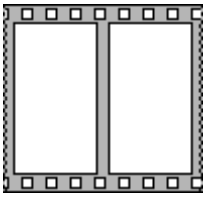
关于错误处理的详细信息见第 6 章*运行和调试 VI* 的*错误处理*一节。

顺序结构

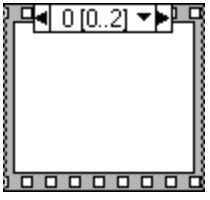
顺序结构包含一个或多个按顺序执行的子程序框图或帧。跟程序框图其它部分一样，在顺序结构的每一帧中，数据依赖性决定了节点的执行顺序。在 LabVIEW 中并不经常使用顺序结构。

顺序结构现包括两种类型：平铺式顺序结构和层叠式顺序结构。

如下图所示，平铺式顺序结构可以一次显示所有帧。当所连接的数据都传递至该帧时，将按照从左到右的顺序执行所有帧，直到执行完最后一帧。帧执行完毕后会将数据至传递至下一帧。



如下图所示，层叠式顺序结构将所有的帧堆积起来，因此每次只能看到其中的一帧，并且按照帧 0、帧 1、直到最后一帧的顺序执行。



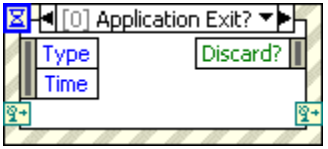
为了充分利用 LabVIEW 固有的并行机制，必须避免过度使用顺序结构。顺序结构虽然可以保证执行顺序但同时也阻止了并行操作。例如，如果没有使用顺序结构，PXI、GPIB、串口和 DAQ 设备等 I/O 设备的异步任务可以同其它操作并发运行。

当需要控制执行顺序时，可以考虑建立节点间的数据依赖性。例如，诸如错误 I/O (error I/O) 等数据流 (flow-through) 参数可用于控制执行顺序。

关于错误 I/O 的详细信息见第 6 章*运行和调试 VI* 的*错误处理*一节；关于数据流参数的详细信息见第 5 章*创建程序框图*的*数据流参数*一节。

事件结构

如下图所示，事件结构包括一个或多个子程序框图或事件条件分支，运行结构时将执行其中某一条件分支或子程序框图。



事件结构将等待直至某一事件发生，并执行相应条件分支以便处理该事件。事件可以源于用户界面、外部 I/O 或应用程序的其它部分。用户界面事件包括鼠标点击、键盘按键等动作。外部 I/O 事件包括硬件定时器或触发器在完成数据采集或发生错误情况时产生的信号。通过编程可以产生其它类型的事件，并实现与应用程序的不同部分进行通讯。 LabVIEW 可支持由用户界面和通过编程产生的事件，但不支持外部 I/O 事件。

注　只有 LabVIEW 完整版和专业版开发系统中才包括事件结构。在 LabVIEW 基础版中可以运行带有这些功能的 VI，但无法对事件处理部分进行重新配置。

9

使用字符串、数组和簇将数据分组

使用字符串 (strings)、数组 (arrays) 和簇 (clusters) 可以对数据进行分组。字符串将 ASCII 字符序列归为一组。数组将相同类型的数据元素归为一组。簇将不同类型的数据元素归为一组。

使用字符串将数据分组

字符串是可显示或不可显示的 ASCII 字符序列。字符串可以提供与平台无关的信息和数据的格式。一些常用的字符串应用包括：

- 创建简单的文本信息。
- 将数值数据以字符串的形式传送到仪器并将字符串转换为数值。
- 将数值数据存盘。要将数值数据存入 ASCII 文件，必须在将数据写入磁盘文件之前将这些数值数据转换成字符串。
- 以对话框指示或提示用户。

在前面板上，字符串以表格 (tables)、文本输入框 (text entry boxes) 和标签 (labels) 的形式出现。 LabVIEW 包括的内置 VI 和函数可用来对字符串进行操作，如将字符串格式化、解析字符串等编辑操作。

前面板上的字符串

使用字符串输入控件 (controls) 和显示控件 (indicators) 可以模仿文本输入框和标签。

关于字符串输入控件和显示控件的更多信息参见第 4 章*创建前面板*中的*字符串输入控件和显示控件*一节。

字符串显示类型

右键单击前面板上的字符串输入控件或显示控件，从下表所示的显示类型中进行选择。该表也显示了每个显示类型的示例信息。

显示类型	描述	信息
正常显示 (Normal Display)	以控件字体显示可打印字符。不可显示字符通常显示为一个小方框。	`There are four display types.` `\ is a backslash.`
代码显示 (Codes Display)	显示所有不可显示字符的反斜杠码。	`There\sare\sfour\sdisplay\stypes.\n\\\sis\sa\sbackslash.`
密码显示 (Password Display)	以星号 (*) 显示包括空格在内的每一个字符。	`**************************` `*****************`
十六进制显示 (Hex Display)	以十六进制而不是字符本身的形式显示每个字符的 ASCII 值。	`5468 6572 6520 6172 6520 666F 7572 2064 6973 706C 6179 2074` `7970 6573 2E0A 5C20 6973 2061 2062 6163 6B73 6C61 7368 2E`

表格

表格控件 (table control) 可用于在前面板上创建表格。表格中的每一个单元都是一个字符串，每一个单元为某一行和某一列的交叉处。因此，表格显示的是二维字符串数组。

关于数组的更多信息见本章*数组*一节。

编辑、格式化、解析字符串

字符串函数编辑字符串的方式大致如下：

- 查找、提取和替换字符串中的字符或子字符串。
- 将字符串中的所有文本转换为大写或小写。
- 在字符串中查找和提取匹配模式。
- 从字符串中提取一行。
- 将字符串中的文本旋转和反序。
- 连接两个或多个字符串。
- 删除字符串中的字符。

关于以编程方式编辑字符串时如何最少占用内存的更多信息见 *LabVIEW Help* 中的 *LabVIEW Style Checklist*。关于使用字符串函数编辑字符串的范例见 `labview\examples\general\strings.llb`。

格式化和解析字符串 (Formatting and Parsing Strings)

要在另一个 VI、函数或应用程序中使用数据，通常必须先将数据转换为字符串然后格式化该字符串，便于 VI、函数或应用程序读取。例如，Microsoft Excel 希望字符串中含有分隔符，如制表符、逗号与空格键。Excel 用分隔符将数字或单词分离开来再存入单元格中。

例如，要用 Write To Binary File 函数将一维数值数组写入电子表格，就必须将该数组格式化成字符串并用制表符等分隔符将各个数值分开。要使用 Write To Spreadsheet File VI 将一个数值数组写入电子表格，必须用 Array To Spreadsheet String 函数格式化该数组并指定格式和分隔符。

字符串函数可用于实现以下任务：

- 从一个字符串中提取字符串子集。
- 将数据转换为字符串。
- 格式化字符串以便用于文字处理或电子表格应用程序。

使用文件 I/O VI 和函数将字符串保存到文本和电子表格文件中。

格式化符

在许多情况下，必须在字符串函数的 **format string** 参数中输入一个或多个格式化符 (format Specifiers) 来格式化一个字符串。格式化符是一个表明如何将数值数据转换为字符串，或从字符串转换为数值的编码。 LabVIEW 使用转换码来确定参数的文本格式。例如，格式化符 `%x` 可将十六进制整数与字符串相互转换。

使用数组和簇将数据分组

数组、簇的控件和函数可用于对数据进行分组。数组将相同类型的数据元素归为一组。簇将不同类型的数据元素归为一组。

数组

数组由元素和维度组成。元素是组成数组的数据。而维数是数组的长度、高度或深度。一个数组可以是一维或多维的，而且每维在内存允许的情况下可以有多达 (2^{31}) -1 个元素。

可以创建数值 (numeric)、布尔 (Boolean)、路径 (path)、字符串 (string)、波形 (waveform) 和簇 (cluster) 等数据类型的数组。在对一组相似数据进行操作并重复计算时，可以考虑使用数组。数组还适用于存储从波形收集的数据或在循环中生成的数据 （每次循环生成数组中的一个元素）。

限制

在数组中不能再创建数组。但可以创建多维数组或创建每个簇中含有一个或多个数组的簇数组。用户也不能创建子面板控件 (subpanel controls)、tab 控件 (tab controls)、.NET 控件 (.NET controls)、ActiveX 控件 (ActiveX controls)、图表 (charts) 或多曲线 XY 图 (multiplot XY graphs) 的数组。

关于簇的更多信息见本章*簇*一节。

索引

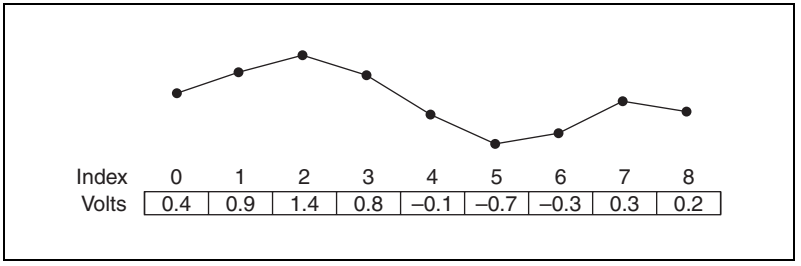
要定位数组中的某个特定元素需要为每一维建一个索引 (index)。在 LabVIEW 中，通过索引可以浏览整个数组，也可以从程序框图数组中提取元素、行、列和页。

数组举例

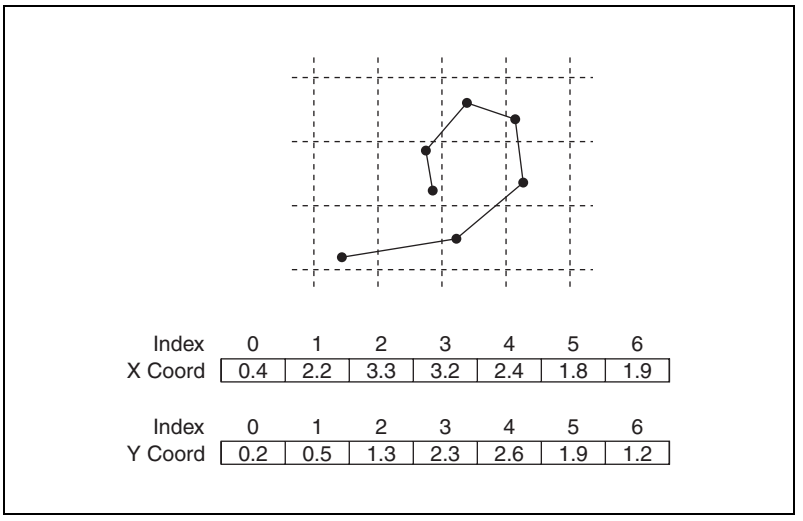
比如，一个列出太阳系中九大行星的文本数组就是一个简单的数组的例子。LabVIEW 会以含有九个元素的一维字符串数组来表述。

数组元素是有序的。数组使用索引，便于方便地访问数组中任意一个特定的元素。索引是以零开始的，也就是说索引的范围是 0 到 *n* 一 1，其中 *n* 是数组中元素的个数。例如，对于九个行星，*n* = 9，因此索引的范围是 0 到 8。地球是第三个行星，因此其索引为 2。

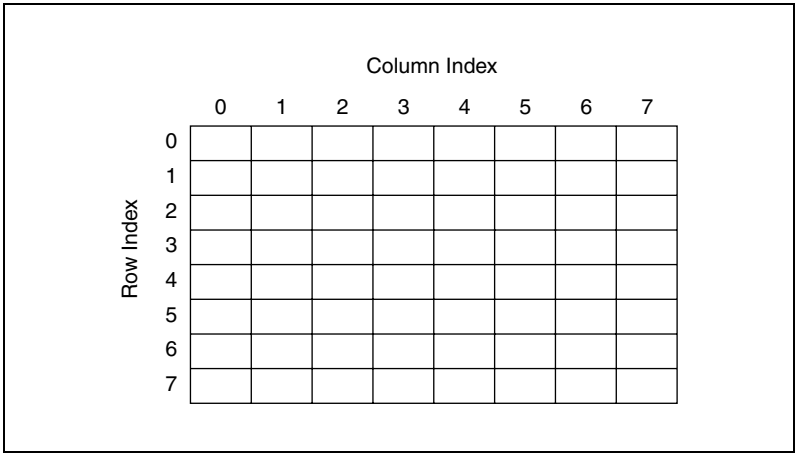
另一个数组的例子是以数值数组表示的波形，该数组中每一个连续元素是具有连续时间间隔的电压值，如下图所示。



一个更为复杂的数组的例子是以点数组表示的图形，其中每一个点是一对表示 X 坐标和 Y 坐标的数值簇，如下图所示。

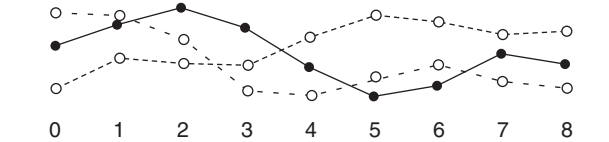


前面的例子都是一维数组。二维数组存储元素于网格中。这时，需要一个行索引和一个列索引来定位数组中的某一个元素，并且，这两个索引都是从零开始的。下图显示了一个 8 行 8 列的二维数组，其中包含 8 × 8 = 64 个元素。



例如，一个棋盘有八列和八行共 64 个位置。每个位置可以为空或有一个棋子。可以用一个二维字符串数组表示一个棋盘。每一个字符串是占据棋盘上相应位置的一个棋子的名称，或是空字符串 （当位置为空时）。

可以在前面的一维数组例子中添加一个行将数组统一成二维数组。下图显示了一个以二维数值数组表示的一组波形。其中行索引用于指定波形，列索引指定波形上的点。

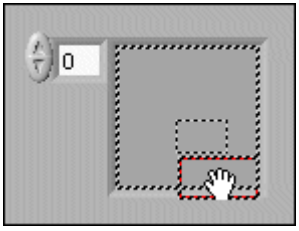


	0	1	2	3	4	5	6	7	8
0	0.4	0.9	1.4	0.8	–0.1	–0.7	–0.3	0.3	0.2
1	–0.1	0.6	0.4	0.2	0.8	1.6	1.4	0.9	1.1
2	1.6	1.4	0.7	0.5	–0.5	–0.6	–0.2	0.3	0.5

关于使用数组的范例见 `labview\examples\general\arrays.llb`。

创建数组输入控件、显示控件和常量

如下图所示，可以通过以下方式在前面板上创建一个数组输入控件或显示控件：先在前面板上放置一个数组壳 (array shell)，然后将一个数据对象 (object) 或元素 (element) 拖放到该数组壳中，数据对象或元素可以是数值 (numeric)、布尔值 (Boolean)、字符串 (string)、路径 (path)、引用句柄 (refnum)、簇输入控件或显示控件 (cluster control or indicator)。



数组壳会自动改变大小以容纳新对象。

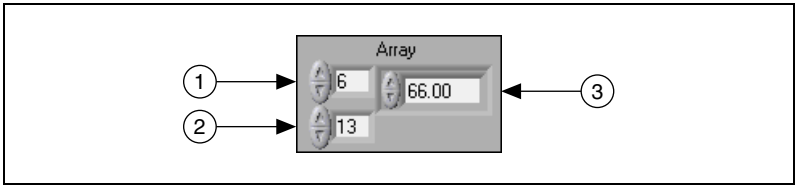
要在程序框图中创建数组常量，先从**函数**选板上选择数组常量，并将数组壳放置于程序框图上，然后将字符串常量、数值常量或簇常量放置到数组壳中。数组常量用于存储常量数据或用于同另一个数组进行比较。

创建多维数组

要在前面板上创建一个多维数组控件，右键单击索引框并从快捷菜单中选择 **Add Dimension**。也可以改变索引显示器的大小直到所希望的维数的出现。要一次性删除数组中的维数，右键单击索引框并从快捷菜单中选择 **Remove Dimension**。也可以改变索引框的大小来删除维数。

要在前面板上显示某个特定的元素，可以在索引框中输入索引数字或使用索引框上的箭头找到该数字。

例如，一个二维数组中包含行和列。如下图所示，左边的两个方框中上面的索引为行索引，下面的索引为列索引。而行和列右边的组合显示为指定位置的数据值。如下图所示，位置在第六行、第十三列处的值为 **66**。



1　行索引	3　行列定位取值
2　列索引	

行和列是从零开始的，也就是说第一列为列 0，第二列为列 1，依此类推。将下面数组中的索引显示器中的值改为行 1，列 2 将显示 **6**。

0	1	2	3
4	5	6	7
8	9	10	11

若试图显示超出数组维数范围的某一行或某一列，数组控件将变暗以表示该数据没有定义，同时 LabVIEW 将显示该数据类型的默认值。而数据类型的默认值取决于该数组的数据类型。

定位工具可用于改变数组的大小，以便一次显示多行或多列。

数组函数

使用数组函数 (Array Functions) 可以创建数组并对其操作。比如，可以执行以下操作：

- 从数组中提取单个数据元素。
- 在数组中插入、删除或替换数据元素。
- 分解数组。

Build Array 函数可通过编程方式创建数组。也可以通过循环来创建数组。

关于使用循环创建数组的信息参见第 8 章*循环和结构*中的*使用循环创建数组*一节。

关于在循环中使用数组函数时如何最少占用内存的更多信息参见 *LabVIEW Help* 中的 *LabVIEW Style Checklist*。

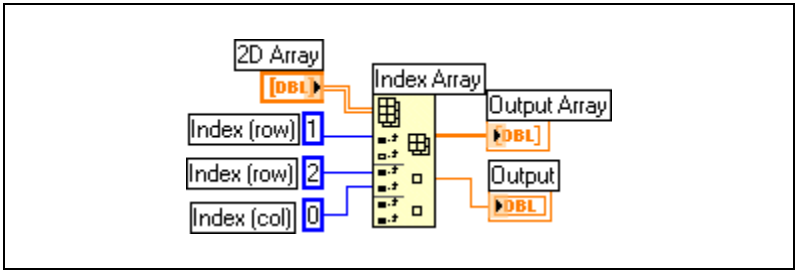
自动改变数组大小的函数

Index Array、Replace Array Subset、Insert Into Array、Delete From Array 和 Array Subset 等函数可以自动改变大小以匹配所连接的输入数组的维数。例如，如果将一个一维数组连接到以上某一个函数，该函数将只显示单个索引输入。如果将一个二维数组连接到同样的函数，它将显示两个索引输入，其中一个用于行索引，另一个用于列索引。

使用定位工具手动改变这些函数的大小，便可以通过这些函数访问多个数组元素或子数组（行、列或页）。扩展这些函数中的某个函数时，该函数将根据与之相连的数组维数的增加而增加。如果将一个一维数组连接到以上某个函数，该函数将以单个索引输入为单位扩展。如果将一个二维数组连接到这个函数，该函数将以两个索引输入为单位扩展，其中一个用于行索引，另一个用于列索引。

所连接的索引输入决定了要访问或修改的子数组的形状。例如，如果 Index Array 函数的输入为一个二维数组，但只连接了行索引输入，那么提取的是该数组的一个完整的一行。如果只连接了列索引输入，那么提取的是该数组的一个完整的一列。如果同时连接了行索引输入和列索引输入，那么提取的是数组的单个元素。每项输入是独立的，因此可以访问数组中任何维数的任何部分。

下图所示的程序框图为使用 Index Array 函数来提取二维数组中的一行和一个元素。



要访问一个数组中的多个连续值，可以扩展 Index Array 函数，而无需为所扩展的索引输入端连线赋值。例如，要提取某个二维数组的第一、二、三行，可将该 Index Array 函数扩展三个单位，然后将一维数组显示控件连接到每个子数组 (sub-array) 输出端。

数组中的默认数据

索引超出数组范围时，数组元素参数将会生成默认值。可以使用 Array Size 函数决定数组的大小。

使用 While 循环 (While Loop) 索引超过数组最后元素的一个元素或给 Index Array 函数的 **index** 输入端赋了一个太大的数值，或将空数组赋给 Index Array 函数都会非预期地超过数组索引的最大值。

关于索引的更多信息见第 8 章*循环和结构*中的*自动索引循环*一节。关于数据类型默认值的更多信息参见 *《LabVIEW 快速参考指南》(LabVIEW Quick Reference Card)*。

簇

簇将不同类型的数据元素归为一组。 LabVIEW 错误簇是簇的一个例子，它包含一个布尔值、一个数值和一个字符串。簇类似于文本编程语言中的记录或结构体。

关于使用错误簇的更多信息参见第 6 章*运行和调试 VI* 中的*错误簇*一节。

将几个数据元素捆绑成簇可以消除程序框图上混乱的连线，减少子 VI 所需的接线器接线端 (connector pane terminals) 的数目。接线器最多可以有 28 个接线端。如果前面板上要传送给另一个 VI 的输入控件和显示控件多于 28 个，应将其中的一些对象组成一个簇，然后为该簇分配一个接线器接线端。

程序框图上的绝大多数簇都含有一个粉红色的接线模型和数据类型接线端。由数值控件组成的簇，有时也称为点，含有一个褐色的连线式样和数据类型接线端。可以将褐色的数值簇连接到数值函数，比如加法或平方根函数来对簇中的所有元素同时进行同样的运算。

簇元素顺序

尽管簇和数组元素都是有序的，但如需访问特定簇元素，则必须一次拆分 (unbundle) 所有簇元素或使用 Unbundle By Name 函数。簇不同于数组的地方还在于簇的大小是固定的。与数组一样，一个簇里面要么全是输入控件要么全是显示控件。簇不能同时含有输入控件和显示控件。

簇元素有自己的逻辑顺序，与它们在壳中的位置无关。放入簇中的第一个对象是元素 0，第二个为元素 1，依此类推。如果删除某个元素，该顺序会自动调整。簇顺序决定了簇元素在程序框图上的 Bundle 和 Unbundle 函数上作为接线端出现的顺序。右键单击簇边框并从快捷菜单中选择 **Reorder Controls In Cluster** 可以查看和修改簇顺序。

要对两个簇进行连线，它们必须要有相同数目的元素。与簇顺序相对应的元素也必须具有相兼容的数据类型。例如，一个簇中的双精度浮点数值与另一个簇中的一个字符串有相同的簇顺序，如果将这两个簇相连，那么连线将是断开的并且 VI 不能运行。如果数值的表示法不同， LabVIEW 会将它们强制转换成同一表示法。

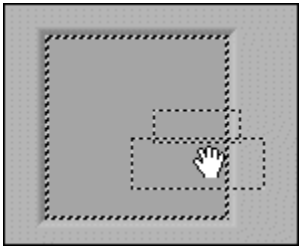
簇函数

使用簇函数 (Cluster Functions) 可以创建簇并对其操作。比如，可以执行以下操作：

- 从簇中提取单个数据元素。
- 向簇添加单个数据元素。
- 将簇拆分成单个数据元素。

创建簇输入控件、显示控件和常量

如下图所示，可以通过以下方式在前面板上创建一个簇输入控件或显示控件：先在前面板上放置一个簇壳，然后将一个数据对象或元素拖放到该簇壳中，该数据对象或元素可以是数值、布尔值、字符串、路径、引用句柄、簇输入控件或显示控件。



要在程序框图中创建一个簇常量 (cluster constant)，先从**函数**选板中选择一个簇常量，并将簇壳放置于程序框图中，然后将字符串常量、数值常量或簇常量放置到该簇壳中。簇常量用于存储常量数据或用于同另一个簇进行比较。

10

图形和图表

采集或生成数据后，可以使用图形 (Graph) 或图表 (Chart) 以图形化的形式显示数据。

图形和图表的区别在于它们显示和更新数据的方式不同。带有图形的 VI 通常先将数据收集到数组中，然后绘制到图形上。它类似于电子表格，先存储数据然后生成数据图形。数据绘制到图形上时，图形不会显示旧数据而只显示当前的新数据。通常在快速采集连续数据的过程中使用图形。

而图表将新的数据点追加到已显示的数据点上以形成趋势图。在图表上，可以结合已采集的数据查看当前读数或测量值。向图表中添加更多数据点时，图表将会滚动显示，即在图表右侧显示所添加的新数据点，并且旧数据点会在左侧消失。通常在每秒只增加少量数据点的慢速过程中使用图表。

图形和图表的类型

LabVIEW 包括以下图形和图表类型：

- **波形图和图表 (Waveform Graphs and Charts)** －显示具有恒定采样率的数据。
- **XY 图 (XY Graphs)** －显示非均匀采样的数据，以及显示多变量函数的数据。
- **强度图和图表 (Intensity Graphs and Charts)** －通过使用颜色显示第三维数据值的方式在二维标绘图上显示三维数据。
- **数字波形图 (Digital Waveform Graphs)** －以脉冲或数字线的形式显示数据。
- **(Windows) 三维图形 (3D Graphs)** －在前面板用 ActiveX 三维标绘图对象显示三维数据。

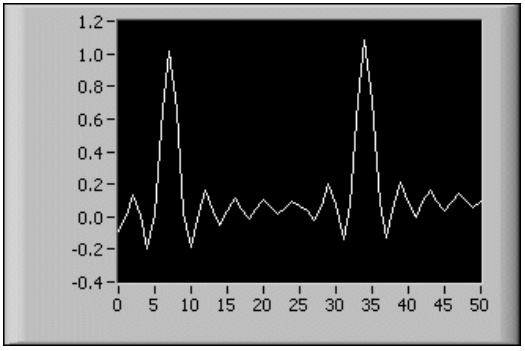
关于各种图形和图表的范例见 `labview\examples\general\graphs`。

波形图和图表

LabVIEW 内含波形图和图表 , 它们可以显示具有恒定速率的数据。

波形图

波形图 (Waveform Graph) 可显示均匀采样的单条或多条曲线 (Plot)。波形图仅绘制单变量函数，比如 *y* = f(*x*)，并且点沿 x 轴均匀分布，比如随时间变化采集到的波形。下图为一个波形图的范例。



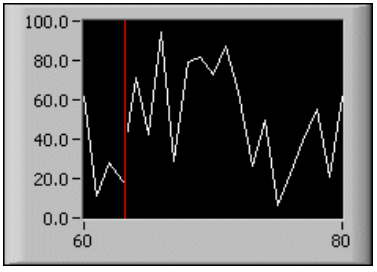
波形图可显示包含任意个数据点的图形。波形图也支持多种数据类型，从而减少对数据类型进行转换的工作量，以方便使用。

注　请使用数字波形图显示数字数据。关于数字波形图及其支持的数据类型的详细信息见本章的*数字波形图*一节。

关于波形图可支持的数据类型的范例见 `labview\examples\general\graphs\gengraph.llb` 中的 Waveform Graph VI。

波形图表

波形图表 (Waveform Chart) 是显示一条或多条曲线的一类特殊的数值显示控件，用来显示通常以恒定速率采集到的数据信号曲线。下图为一个波形图表的范例。



波形图表会保留来源于前几次更新的数据的历史记录，这样的历史记录也称为数据缓冲区。右键单击图表，并从快捷菜单中选择 **Chart History**

Length 可以配置缓冲区大小。波形图表的图表历史数据长度 (Chart History Length) 的默认大小为 1024 个数据点。向图表传送数据的频率决定了图表重绘的频率。

关于波形图表的范例见 `labview\examples\general\graphs\charts.llb`。

波形数据类型

波形数据类型包括波形的数据、起始时间和时间间隔 (△ *t*)。 Build Waveform 函数可用于创建波形。默认情况下，很多用于采集或分析波形的 VI 和函数都可以接收和返回波形数据类型。当将波形数据连接到一个波形图或图表上，该图或图表会根据波形数据、起始时间和△ *x* 自动绘制波形。将一个波形数据数组连接到波形图或图表上时，该图形或图表会自动绘制所有波形。

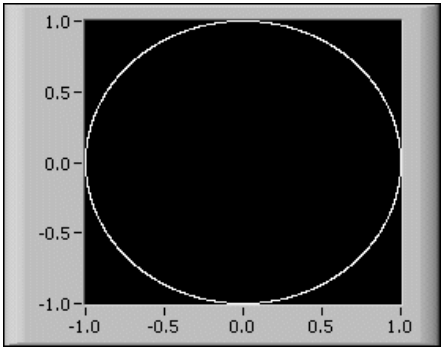
关于数字波形数据类型的详细信息见本章的*数字波形数据类型*一节。

XY 图

XY 图是通用的笛卡尔绘图对象，可以用来绘制多变量函数，比如圆形或具有可变时基的波形。 XY 图可以显示任何均匀采样或非均匀采样的点的集合。

在 XY 图形中可以显示 Nyquist 平面 (Plane)、 Nichols 平面、 S 平面和 Z 平面。上述平面的行和标签的颜色与笛卡尔线相同，并且无法修改平面标签的字体。

下图为一个 XY 图的范例。

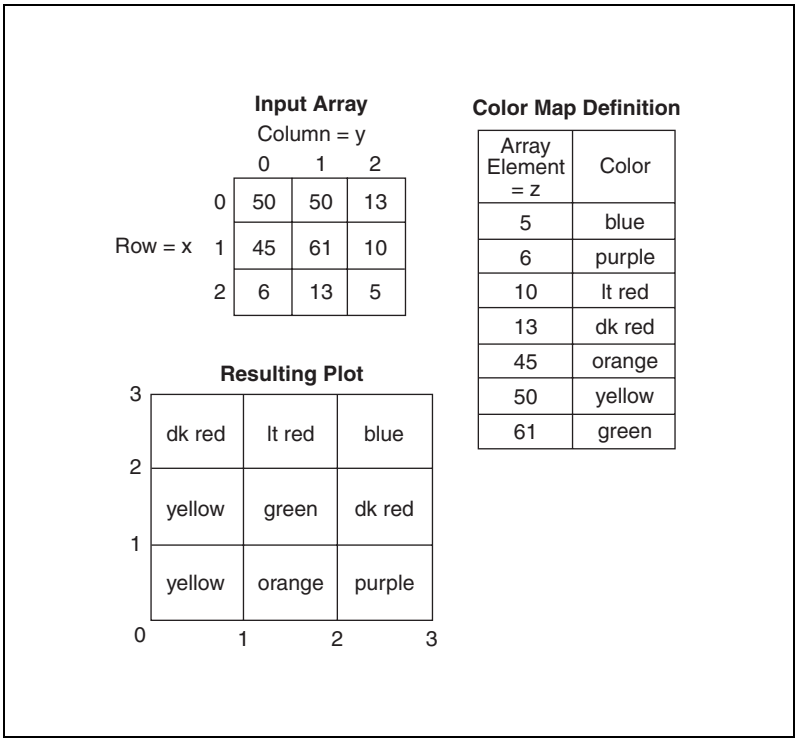


XY 图可显示包含任意个数据点的曲线。它支持多种数据类型，从而减少对数据类型进行转换的工作量，以方便使用。

关于 XY 图形的范例见 labview\examples\general\graphs\gengraph.llb 中的 XY Graph VI。

强度图和图表

强度图和图表 (Intensity Graphs and Charts) 通过在笛卡尔平面上放置颜色块，从而在二维标绘图上显示三维数据。例如，强度图和图表可以显示温度分布和地形（其中量值代表高度）。强度图和图表接收三维数字数组。数组中的每一个数字代表一个特定的颜色。在二维数组中，元素的索引可用于设置颜色在图形中的位置。下图为强度图表操作的有关概念。



数据行在图形或图表上将以新列显示。如需仍以行的方式显示该行，可将一个二维数组数据类型连接到图形或图表，右键单击图形或图表并从快捷菜单中选择 **Transpose Array**。

数组索引对应于颜色块的底部左顶点。颜色块有一个单位面积，即由数组索引所定义的两点间的面积。强度图或图表最多可以显示 256 种不同颜色。

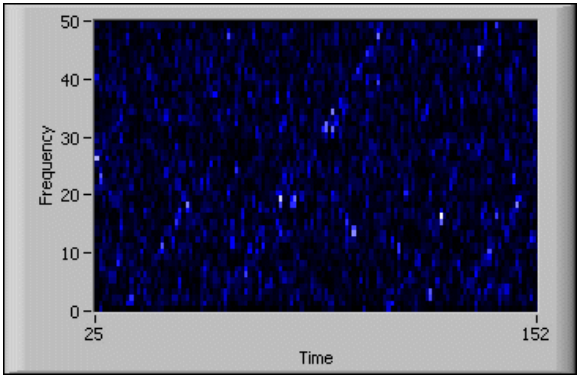
关于各种强度图和图表的范例见 labview\examples\general\graphs\intgraph.llb。

强度图表

在强度图表上绘制一个数据块以后，笛卡尔平面的原点将移动到最后一个数据块的右边。在图表处理新数据时，新数据出现在旧数据的右边。如图表显示已满，旧数据将移出图表的左边界。这一点类似于带状图表 (Strip Chart)。

关于带状图表的详细信息见本章的*配置图表刷新模式*一节。

下图为一个强度图表的范例。



强度图表和波形图表共享部分可选项，如刻度图例 (Scale Legend) 和图形工具选板 (Graph Palette)，右键单击图表并从快捷菜单中选择 **Visible Items** 可以显示或隐藏上述选项。此外，由于强度图表包含颜色作为第三维，因此一个类似于颜色梯度控件的刻度定义了数值到颜色的范围及映射。

关于颜色映射的详细信息见本章的*强度图和图表的颜色映射*。

跟波形图表一样，强度图表也有一个保存了前几次更新数据的历史数据缓冲区。右键单击图表，并从快捷菜单中选择 **Chart History Length** 可以配置缓冲区大小。强度图表缓冲区的默认大小为 128 个数据点。强度图表的显示可能需要占用大量的内存。

强度图

强度图类似于强度图表，只是它并不保存先前的数据，也不支持更新模式。每次将新数据传送至强度图时，新数据将替换旧数据。和其它图形一样，强度图也包括游标 (Cursor)。每个游标可以显示图形上指定点的 *x*、*y* 和 *z* 值。

关于使用游标的详细信息见本章的*使用图形游标*一节。

强度图和图表的颜色映射

强度图或图表使用颜色在二维标绘图上显示三维数据。为强度图或图表设置好颜色映射后，可以配置图形或图表的颜色刻度。颜色刻度包括至少两个任意坐标，每个坐标均含有数值和对应的显示颜色。强度图或图表所显示的颜色与指定颜色的数值一一对应。颜色映射对于通过视觉表示数据范围来说非常有用（比如当绘图数据超过极限值时）。

通过用与设定颜色梯度数值控件 (Color Ramp Numeric Control) 颜色相同的方式可以交互地设置强度图和图表的颜色映射。

注　强度图或图表显示的颜色会受到显卡所能显示的颜色和颜色数的限制，同时还受分配给显示所用的颜色数的限制。

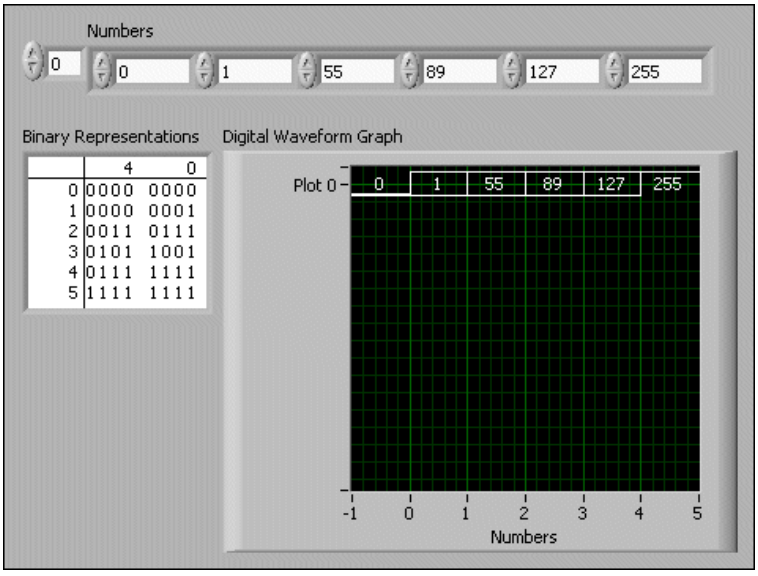
关于颜色映射的范例信息见 `labview\examples\general\graphs\intgraph.llb` 中的 Create IntGraph Color Table VI。

数字波形图

数字波形图可用于显示数字数据，尤其在使用定时框图或逻辑分析器时。

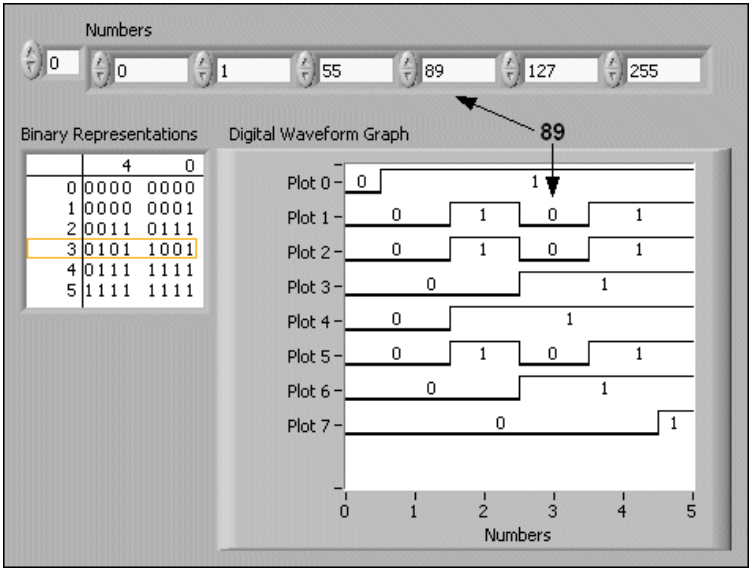
数字波形图可以接收数字波形数据类型、数字数据类型和包含这些数据类型的数组作为输入。默认情况下，由于数字波形图压缩显示数字总线，因此该图形会在单条曲线上绘制数字数据。如果连接了一个数字数据数组，数字波形图将按照数组的顺序为每个数组元素绘制不同的曲线。

以下前面板中的数字波形图显示了在单条曲线上绘制数字数据。 VI 将 **Numbers** 数组中的数字转换成数字数据，然后在 **Binary Representations** 数字数据显示控件中显示这些数字的二进制表示。在该数字图形中，数字 0 以无顶部直线的形式表示所有数字位的值为零。而数字 255 则以无底部直线的形式来表示所有数字位的值为 1。



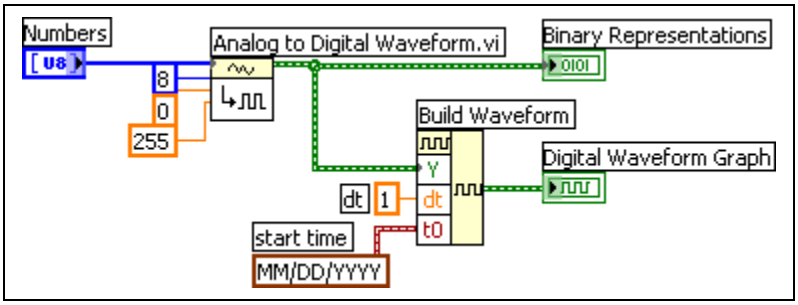
右键单击 y 轴并从快捷菜单中选择 **Expand Digital Buses**，可以绘制数字数据的每一个采样值。每一条曲线表示数字波形中的各个不同数位。

以下前面板中的数字波形图显示了 **Numbers** 数组中的六个数字。



Binary Representations 数字显示控件显示了这些数字的二进制表示。表中的每一列代表一个位。例如，数字 89 需要 7 位内存 （第 7 列的 0 表示未使用的位）。数字波形图上的点 3 绘制了表示数字 89 所必需的 7 位，数值 0 表示曲线 7 上未使用的第 8 个数字位。

以下 VI 显示了将数字数组转换成数字数据，并使用 Build Waveform 函数收集在数字数据控件中输入的起始时间、时间间隔 (△ *t*) 以及数字以便显示数字数据。



关于数字数据控件的详细信息见第 4 章 *创建前面板*的 *数字数据控件*一节。

关于数字波形图的范例见 `labview\examples\general\graphs\DWDT Graphs.llb`。

数字波形数据类型

数字波形数据类型包括数字波形的起始时间、△ *x*、数据和属性。 Build Waveform 函数可用于创建数字波形。当将数字波形数据连接到一个数字波形图上时，该图形会根据时间信息和数字波形数据自动绘制波形。将数字波形数据连接至数字数据显示控件，可以查看数字波形的采样和信号。

关于波形数据类型的详细信息见本章的 *波形数据类型*一节。

三维图形

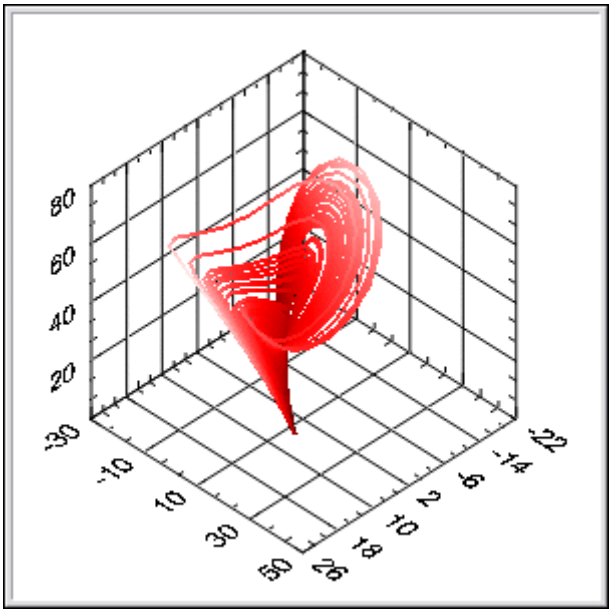
对于许多现实世界中的数据集合，比如某个表面的温度分布、联合时频分析、飞机的运动等，都需要在三维空间中显示数据。使用三维图形，可以使三维数据可视化并可以通过修改三维图形属性来改变数据的显示方式。

注　只有 Windows 中的 LabVIEW 完整版和专业版开发系统中才有三维图形控件。

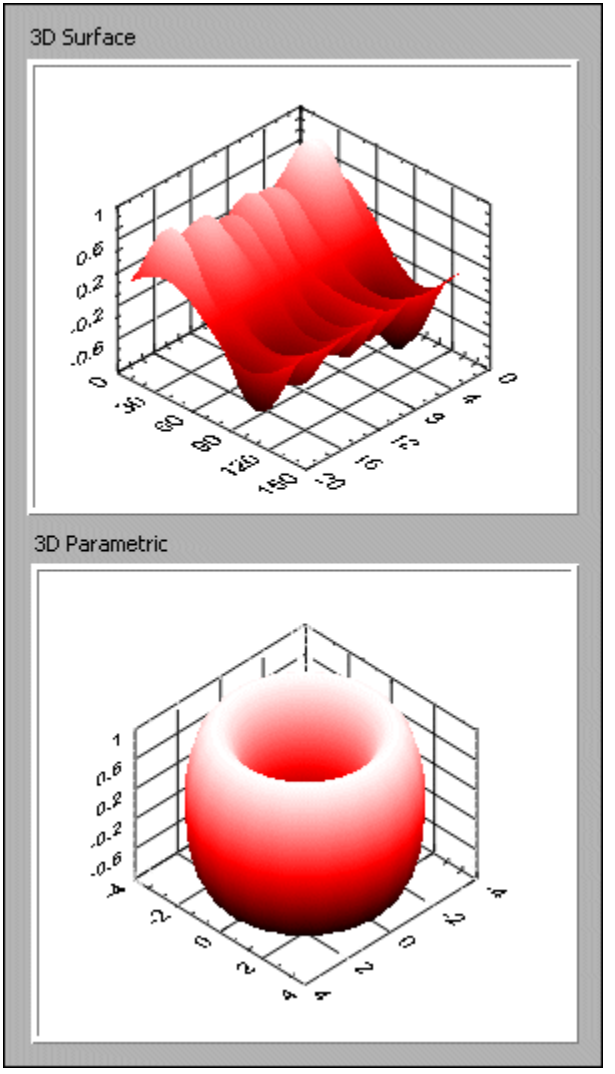
LabVIEW 包括下面几种三维图形：

- **三维曲面图 (3D Surface Graph)** －在三维空间绘制一个曲面。
- **三维参数曲面图 (3D Parametric Surface Graph)** －在三维空间绘制一个参数曲面。
- **三维曲线图 (3D Curve Graph)** －在三维空间绘制一条曲线。

将三维图形跟三维图形 VI 结合起来可以绘制曲线和表面。曲线包含图形上的单个点，每个点都包括 *x*、*y* 和 *z* 坐标，VI 用线连接这些点。曲线可以比较理想地显示运动对象的路径，比如飞机的飞行轨迹。下图显示了一个三维曲线图范例。



曲面图使用 *x*、*y* 和 *z* 数据绘制图形上的各点，然后连接这些点以形成数据的三维曲面。例如，可以使用曲面图实现地形映射。下图为三维曲面图和三维参数曲面图的范例。



三维图形使用了 ActiveX 技术和用于处理三维表示的 VI。选择三维图形时， LabVIEW 在前面板上放置一个包含该三维图形控件的 ActiveX Container。同时也在程序框图上放置一个指向该三维图形控件的引用。LabVIEW 会将该引用与 3 个三维图形 VI 中的某一个进行连接。

自定义图形和图表

每个图形和图表都包括各种选项，用户可以使用这些选项自定义图形和图表的外观，从而表达更多信息或突出显示数据等。尽管图形和图表绘制数据的方式不同，但它们也包括部分可以从快捷菜单访问的共同选项。不过有些选项仅对特殊类型的图形或图表有效。

关于图形或图表特有选项的详细信息见本章的*自定义图形*和*自定义图表*两节。

使用多个 X 和 Y 刻度

所有图形都支持多个 x 刻度和 y 刻度，而所有图表均只支持多个 y 刻度。可以在图形或图表上使用多个刻度显示多个不共享 x 刻度或 y 刻度的图形。如需为图形或图表添加多个刻度，请使用右键单击图形或图表的刻度并从快捷菜单中选择 **Duplicate Scale**。

自动刻度调整

所有图形和图表均可以自动调整水平和垂直刻度，以便与连接到图形或图表上的数据相匹配，这种行为称作自动刻度调整。打开或关闭自动刻度调整，请使用右键单击图形或图表，并从快捷菜单中选择 **X Scale»AutoScale X** 或 **Y Scale»AutoScale Y**。默认情况下，图形和图表已启用自动刻度调整功能，不过自动刻度调整会降低系统的性能。

操作工具或标注工具可以直接改变图形和图表的水平或垂直刻度。

格式化 X 和 Y 刻度

通过 **Properties** 对话框中的 **Format and Precision** 选项卡可以指定 x 轴和 y 轴的刻度在图形或图表上的显示方式。

默认情况下， x 刻度使用浮点表示法并带有 `Time` 标签， y 刻度使用自动格式化并带有 `Amplitude` 标签。要配置图形和图表的刻度，请使用右键单击该图形或图表并从快捷菜单中选择 **Properties**，然后在出现的 **Graph Properties** 对话框或 **Chart Properties** 对话框中加以配置。

通过 **Properties** 对话框中的 **Format and Precision** 选项卡可以指定图形或图表刻度 （坐标轴）的数字格式。在 **Scales** 选项卡可以重命名坐标轴刻度并对其格式化。默认情况下，图形或图表刻度在自动切换成科学表示法之前最多可显示六位数。

在 **Format and Precision** 选项卡中，选择 **Advanced editing mode** 可以显示文本选项，直接输入格式化字符串。要自定义刻度的外观和数值精度，请输入格式化字符串 (Format Strings)。

使用图形工具选板

如下所示，在运行 VI 时使用图形工具选板 (Graph Palette) 可以实现与图形或图表的交互式操作。

通过图形工具选板，可以移动游标、缩放以及平移所显示的图像。右键单击图形或图表并从快捷菜单选择 **Visible Items»Graph Palette** 可以显示或隐藏图形工具选板。图形工具选板从左至右显示了下列按钮：

- **游标移动工具 (Cursor Movement Tool)** (仅对图形有效) －移动所显示图形的游标。
- **缩放 (Zoom)** －放大或缩小显示图形。
- **平移工具 (Panning Tool)** －在显示区域内移动曲线或标绘图。

单击图形工具选板中的按钮后，即可移动游标、缩放或平移显示图像。被启用的按钮会显示绿色指示灯。

自定义图形和图表的外观

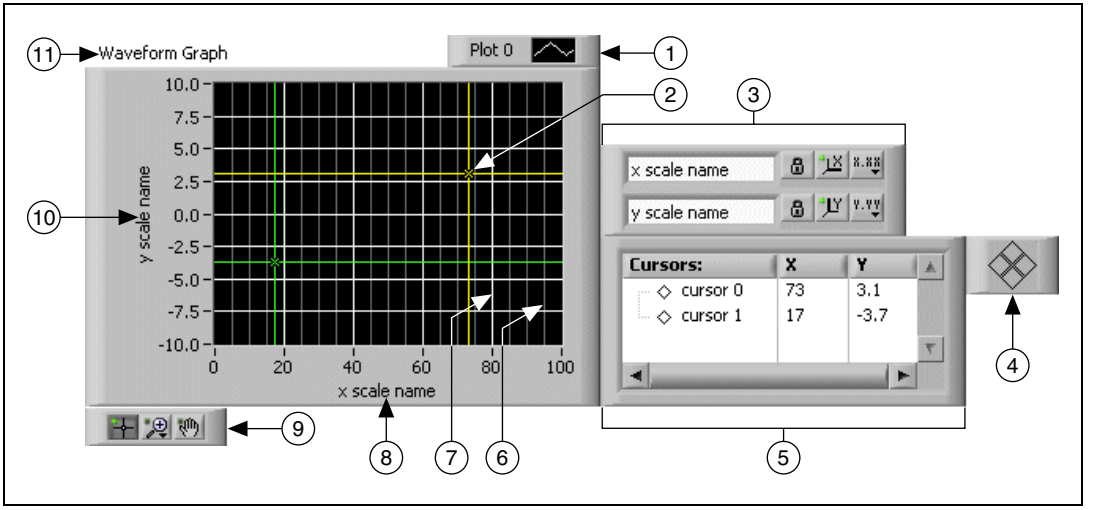
通过显示或隐藏选项可以自定义图形和图表的外观。右键单击图形或图表并从快捷菜单选择 **Visible Items** 可以显示或隐藏以下选项：

- **图例 (Plot Legend)** －定义曲线的颜色和式样。改变图例的大小可以显示多条曲线。
- **刻度图例 (Scale Legend)** －定义刻度标签、配置刻度属性。
- **图形工具选板 (Graph Palette)** －在 VI 运行时移动游标、缩放以及平移图形或图表。
- **X 刻度和 Y 刻度 (X Scale and Y Scale)** －格式化 x 刻度和 y 刻度。

  关于刻度格式化的详细信息见本章的*格式化 X 和 Y 刻度*一节。
- **游标图例 (Cursor Legend)**(仅对图形有效) －在指定点的坐标位置显示标记。可以在图形上显示多个游标。
- **X 滚动条 (X Scrollbar)** －滚动显示图形或图表中的数据。使用滚动条可以查看图形或图表当前没有显示的数据。
- **数字显示 (Digital Display)** (仅对波形图表有效) －显示图表的数值。

自定义图形

每个图形均包括各种选项，用户可以自定义图形用以满足数据显示的要求。例如，可以修改图形游标的行为和外观或配置图形刻度。下图显示了一个图形所具有的元素。



1 图例	4 游标移动器	7 栅格标记	10 Y 刻度
2 游标	5 游标图例	8 X 刻度	11 标签
3 刻度图例	6 微型栅格标记	9 图形工具选板	

使用右键单击图形，从快捷菜单中选择 **Visible Items**，并选择相应的元素可以添加上图所列出的绝大多数元素。右键单击图形，并从快捷菜单中选择相应选项可以配置图形。

使用图形游标

在图形上使用游标可以读取曲线上或绘图区域中某个点的正确值。游标值显示在游标图例中。

右键单击该图形并从快捷菜单选择 **Visible Items»Cursor Legend** 可以查看游标图例。在图形中添加游标，请使用右键单击游标图例中任意区域，并选择 **Create Cursor**，然后从快捷菜单中选择游标模式。

游标模式定义了游标位置。游标包括下列模式：

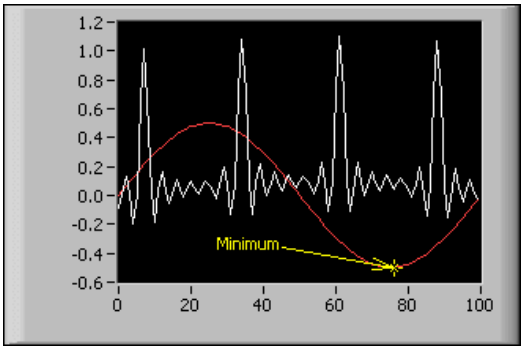
- **自由 (Free)** －在整条曲线或整个绘图区域内自由移动游标。
- **单曲线 (Single-Plot)** －仅位于与游标关联的相应曲线上。并可以在相关曲线中移动。右键单击游标图例并从快捷菜单中选择 **Snap To**，游标可与一条或所有曲线进行关联。
- **多曲线 (Multi-Plot)** －位于整个绘图区域内某个特定数据点的游标。多曲线游标可报告所有曲线的特定 x 坐标的值。可以将游标放在绘图范围内的任意曲线上。右键单击游标图例并从快捷菜单中选择 **Snap To**，游标可与一条或所有曲线进行关联。该模式只对混合信号图形有效。

注　创建游标模式后无法对其进行修改，必须删除游标并创建另一游标。

通过多种方式可以自定义游标的外观，如在曲线上标注游标、指定游标的颜色、线条、点和游标式样等。右键单击游标图例所在行，并从快捷菜单中选择相应选项可以自定义游标。

使用图形注释

使用图形注释 (Graph Annotation) 可在绘图范围内高亮显示数据点。注释包括一个标签和一个箭头来确定注释和数据点。一个图形可以有任意多个注释。下图为一个使用注释的图形范例。



右键单击图形并从快捷菜单中选择 **Data Operations»Create Annotation** 以显示 **Create Annotation** 对话框。 **Create Annotation** 对话框可用于指定注释名称、确定注释在曲线或绘图区域内的位置。

Create Annotation 对话框中的 **Lock Style** 下拉菜单可用于指定注释在曲线或绘图区域内的位置。 **Lock Style** 包括以下选项：

- **Free** －在曲线或绘图区域内自由移动注释。LabVIEW 并没有将注释与曲线或绘图区域内任一曲线进行关联。
- **Snap to All Plots** －将注释移至曲线或绘图区域内任意曲线上最近的数据点。
- **Snap to One Plot** －仅可在特定曲线上移动注释。

通过多种方式可以自定义注释的行为和外观，如可以隐藏或显示曲线或绘图区域内的注释名称或箭头、指定注释的颜色、线条、点和式样等。右键单击注释，并从快捷菜单中选择相应选项便可以自定义注释。

如需删除注释，右键单击该注释，并从快捷菜单中选择 **Delete Annotation**。要删除曲线或绘图区域内的所有注释，请使用右键单击图形并从快捷菜单中选择 **Data Operations»Delete All Annotations**。

自定义三维图形

三维图形包括各种选项用于自定义操作，如三维标绘图式样、刻度格式化、网格和保护绘图。由于三维图形使用 ActiveX 技术和用于处理三维表示的 VI，因此可以给三维图形设置与其它图形不同的选项。创建应用程序时，使用 ActiveX 属性浏览器 (ActiveX Property Browser) 可以设置三维图形的属性。右键单击三维图形，并从快捷菜单中选择 **Property Browser** 显示 ActiveX 属性浏览器。

如需允许用户在运行时改变常规属性，或者通过程序设置属性，请使用 3D Graph Properties VI。

自定义图表

与图形显示新数据并且覆盖已存储数据的方式不同，图表对数据进行周期性更新并保留先前已经存储的历史数据。

通过自定义图表以便符合数据的显示要求。图表选项包括滚动条、刻度图例、图形工具选板、数字显示和刻度的时间表示等。也可以修改图表的历史长度、更新模式和图例显示。

配置图表历史长度

LabVIEW 将已添加到图表中的数据点存储在一个图表历史缓冲区中。图表历史缓冲区的默认大小为 1024 个数据点。使用右键单击该图表，并从快捷菜单中选择 **Chart History Length** 可以配置历史缓冲区大小。图表滚动条可用于查看先前已经采集的数据。右键单击图表并从快捷菜单中选择 **Visible Items»X Scrollbar** 可以显示滚动条。

注　设置大的图表历史值可能需要占用大量的内存。

配置图表刷新模式

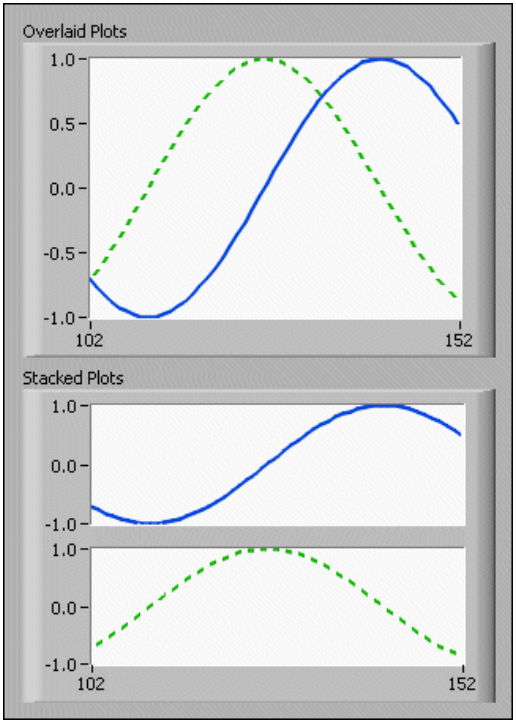
可以配置图表刷新和显示新数据的方式。右键单击该图表，并从快捷菜单中选择 **Advanced Update Mode** 可以配置图表刷新模式。图表使用以下模式显示数据：

- **带状图表 (Strip Chart)** －从左到右连续滚动地显示运行数据，并且旧数据和新数据依次位于图表的左侧和右侧。带状图表类似纸带图形记录器。图表的默认刷新模式是 **Strip Chart**。
- **示波器图表 (Scope Chart)** －显示某一项数据，比如脉冲或波形，并且从左到右部分地滚过图表。图表会将新数值绘制到前一个数据的右边。当曲线到达绘图区域的右边界时， LabVIEW 将擦除整条曲线并从左边界重新开始绘制。示波器图表的重新跟踪显示特性类似于示波器。

- **扫描图表 (Sweep Chart)** — 类似于示波器图表，两者的不同之处在于扫描图表在右边显示旧数据而在左边显示新数据，并且用一条垂直线将这两部分数据隔开。当曲线到达绘图区域的右边界时， LabVIEW 并不擦除扫描图表中的曲线。扫描图类似于心电图仪的显示。

使用层叠曲线和堆栈曲线

通过使用单个垂直刻度可以在一个图表上显示多条曲线，这称为层叠曲线 (Overlaid Plot)；或使用多个垂直刻度，这称为堆栈曲线 (Stacked Plot)。下图为层叠曲线和堆栈曲线的范例。



右键单击图表并从快捷菜单中选择 **Stack Plots** 以多个垂直刻度的方式查看图表曲线；选择 **Overlay Plots** 以单个垂直刻度的方式查看图表曲线。

关于不同类型图表及其所支持数据类型的范例见 `labview\examples\general\graphs\charts.llb` 中的 Charts VI。

11

文件输入 / 输出

通过文件 I/O(File I/O) 操作可以从文件读写数据。 **File I/O** 选板上的 File I/O VI 和函数可用于实现文件 I/O 的所有功能，其中包括：

- 打开和关闭数据文件
- 读写文件数据
- 读写电子表格格式的文件
- 移动或重命名文件和目录
- 修改文件属性
- 创建、修改和读取配置文件

可使用单个 VI 或函数进行文件打开、读写、关闭操作，也可用函数控制每个步骤。 Read From Measurement File Express VI 和 Write To Measurement File Express VI 可用于读写 `.lvm` 或 `.tdm` 文件。

关于 `.tdm` 文件的详细信息见本章的*存储 VI 的应用*一节。

文件 I/O 基础

典型的文件 I/O 操作包括以下流程。

1. 创建或打开文件。通过指定路径或在 LabVIEW 中以对话框的形式确定文件位置，从而指定现有文件的路径或所创建新文件的位置。文件打开后，就用一个引用句柄 (refnum) 来表示该文件。

   关于引用句柄的详细信息见第 4 章*创建前面板*的*对象或应用程序的引用*一节。
2. 读写文件。
3. 关闭文件。

File I/O VI 和部分 File I/O 函数 (如 Read from Text File 和 Write to Text File 函数) 可执行通用文件 I/O 的所有以上三项操作。执行多项操作的 VI 和函数可能在效率上低于专项操作函数。

许多 File I/O VI 和函数含有数据流 (flow-through) 参数，通常是引用句柄或路径，该参数返回与相应的输入参数相同的值。

关于数据流参数的详细信息见第 5 章*创建程序框图*的*数据流参数*一节。

选择文件 I/O 格式

采用何种 **File I/O** 选板上的 VI 取决于文件的格式。现可读写三种格式的文件：文本文件、二进制文件和数据记录 (Datalog) 文件。而所使用的格式取决于文件中保存的数据，以及访问这些数据的应用程序。

通过以下条件可决定所需的格式：

- 如需由其它应用程序访问这些数据，如 Microsoft Excel，可以使用最常用和方便的文本文件。
- 如需随机读写文件或者读取速度和磁盘空间有限，可以使用二进制文件，因为在磁盘空间利用和读取速度方面二进制文件比文本文件更有效。
- 如需在 LabVIEW 中处理复杂的数据记录或不同的数据类型，可以使用数据记录文件。如果希望处理仅来自 LabVIEW 的数据以及存储复杂的数据结构，数据记录文件是最好的存储数据方式。

如果数据本身并不是文本格式，如图形 (Graph) 或图表 (Chart) 数据，由于数据的 ASCII 表示通常要比数据本身大，因此文本文件要比二进制和数据记录文件占用更多的内存。例如，将 -123.4567 作为单精度浮点数保存时只需要 4 个字节，而其 ASCII 表示需要 9 个字节，每个字符占用一个字节。

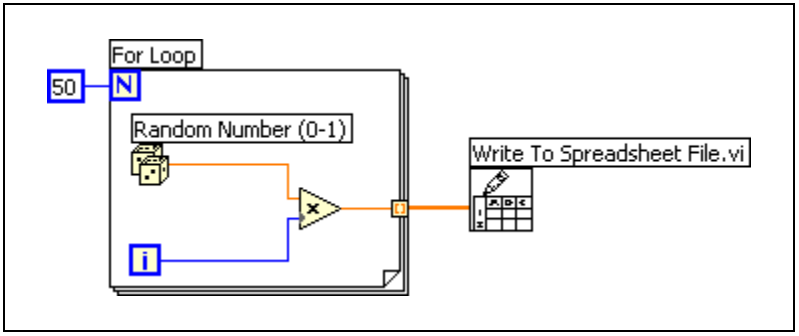
此外，在文本文件中很难进行数字数据的随机访问。尽管字符串中的每个字符占用一个字节的空间，但是将一个数字表示为字符串所需要的空间通常是不固定的。要查找文本文件中的第 9 个数字， LabVIEW 必须先读取和转换前面 8 个数字。

VI 和函数在通用文件 I/O 中的应用

File I/O 选板上的 VI 和函数可用于实现通用文件 I/O 操作，如读写以下类型的数据：

- 在电子表格文本文件中读写数值
- 在文本文件中读写字符
- 从文本文件读取行
- 在二进制文件中读写数据

以下程序框图说明如何使用 Write To Spreadsheet File VI 向电子表格文件传输数字。运行 VI 时， LabVIEW 会提示是否将数据写至现有文件，或者创建新文件。



打开、读取、写入函数需要输入文件路径。如果没有连接文件路径，会出现对话框提示您指定所要读写的文件。

File I/O 选板上的函数可用于控制单项文件 I/O 操作，如创建或打开文件、向文件读写数据、关闭文件等。上述 VI 可实现以下任务：

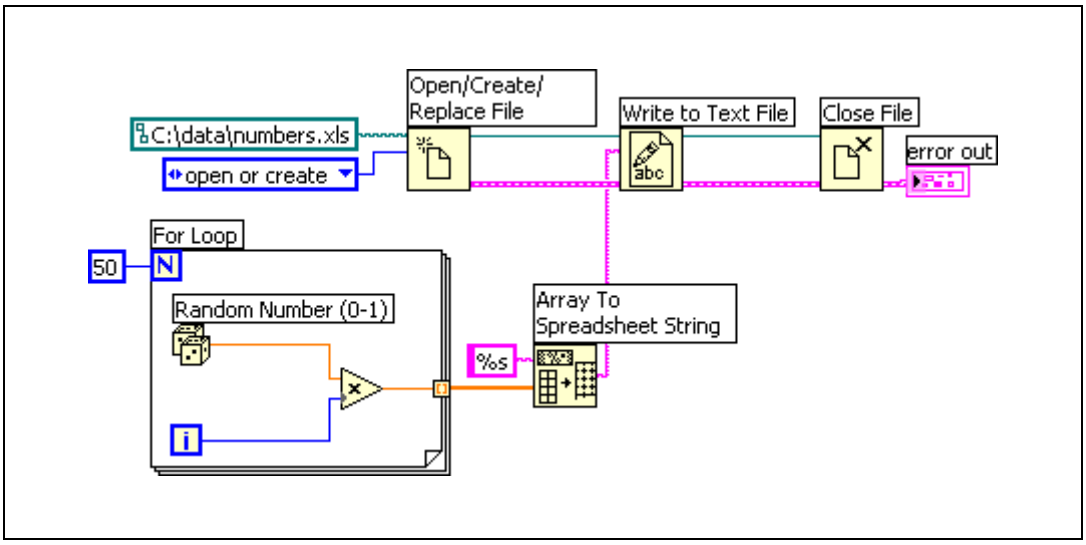
- 创建目录
- 移动、复制或删除文件
- 列出目录内容
- 修改文件属性
- 对路径进行操作

路径 (path) 是一种 LabVIEW 数据类型，用于指定磁盘上文件的位置，如下图所示。

路径包含文件所在的磁盘、系统根目录到文件之间的路径以及文件名等信息。路径和显示控件应根据特定平台的标准语法分别输入或显示路径。

关于路径输入控件和显示控件的详细信息见第 4 章 *创建前面板*的 *路径输入控件和显示控件*一节。

以下程序框图说明如何使用 File I/O 函数向电子表格文件传输数值数据。运行该 VI 时， Open/Create/Replace File 函数将打开 `numbers.xls` 文件，Write to Text File 函数会将数字字符串写入文件，Close File 函数关闭文件。如果不关闭文件，文件将驻留在内存中，且其它应用程序或用户无法访问该文件。



以上程序框图与 Write to Spreadsheet VI 完成的任务相同，我们可以做一下比较。以上程序框图使用单个函数执行各个文件操作，如采用 Array To Spreadsheet String 函数将数字数组转换成字符串。而 Write to Speadsheet File VI 执行多项文件操作，如打开文件、将数字数组转换成字符串并关闭文件。

关于使用高级 File I/O VI 和函数的范例见 `labview\examples\file\datalog.llb` 中的 Write Datalog File Example VI。

File I/O 函数还可用于磁盘流操作，它可以减少函数与操作系统交互的次数，从而节省内存资源。磁盘流是一项在进行多次写操作时保持文件打开状态的技术，比如用于循环中。如果将路径控件或常数与 Write to Text File 函数、Write to Binary File 函数或 Write to Spreadsheet File VI 连线时，在每次函数或 VI 运行时都会进行打开和关闭文件操作，增加了系统开销。如果避免对同一文件进行频繁的打开和关闭操作，将提高 VI 效率。

在循环前面放置 Open/Create/Replace File 函数，在循环内部放置读写函数，在循环之后放置 Close File 函数，即可创建一个典型的磁盘流模式。这样只有写操作在循环内部进行，从而避免重复打开和关闭动作引起的系统开销。

在采集大量数据时，如果对速度要求很高就可以采用磁盘流技术。在进行数据采集的同时可以将数据连续写入文件中。为获取更好的效果，在采集结束前应避免运行其它 VI 和函数 （如分析 VI 和函数等）。

存储 VI 的应用

Storage 选板上的存储 VI(Storage VI) 可用于读写二进制测量文件 (.tdm) 中的波形和波形属性。通过 .tdm 文件可在 NI 软件 （如 LabVIEW 和 DIAdem）间进行数据交换。

注　存储 VI 仅适用于 Windows 操作系统。

存储 VI 可联合波形和波形属性以便构成通道。通道组可管理一组通道，一个文件中可包括多个通道组。如保存通道名称，在现有通道中可以快速追加或检索数据。存储 VI 可以同时支持数据值以及字符串数组和时间标识数组。在程序框图上，引用可代表文件、通道组和通道。

存储 VI 也可以查询文件以获取符合条件的通道组或通道。

如果开发过程中系统要求有所更改，或者需要在文件中添加其它数据，存储 VI 可以修改文件格式而不会导致文件不可用。

关于使用存储 VI 的范例信息见 `labview\examples\file\storage.llb`。

Read From Measurement File Express VI 和 Write To Measurement File Express VI 也可以用于读写 .tdm 测量文件。

创建文本和电子表格文件

将数据写入文本文件，必须先将数据转换成字符串。要将数据写入电子表格文件，必须将字符串格式化成电子表格字符串，电子表格字符串是一种含有分隔符的字符串，比如制表符。

关于格式化字符串的详细信息见第 9 章 *使用字符串、数组和簇将数据分组* 中的 *格式化和解析字符串*一节。

由于大多数文字处理应用程序读取文本时并不要求格式化的文本，因此将文本写入文本文件无需进行格式化。将文本字符串写入文本文件时，可以使用 Write to Text File 函数自动打开和关闭文件。

也可使用 Write to Binary File 函数创建独立于平台的文本文件。 Read from Binary File 函数可用于在独立于平台的文本文件中读取数据。

关于二进制文件的详细信息见 *创建二进制文件*一节。

使用 Write to Spreadsheet File VI 或 Array to Spreadsheet String 函数将来自图形、图表或采集的数据集转换成电子表格字符串。

由于文字处理应用程序采用了 File I/O VI 无法处理的字体、颜色、风格和大小不同的格式化文本，因此从文字处理应用程序所保存的文件中读取文本可能会导致错误。

如果要将数字和文本写入电子表格或文字处理应用程序，字符串函数和数组函数可以格式化数据并将这些字符串组合起来，然后将数据写入文件。

关于使用上述函数格式化字符串的详细信息见第 9 章*使用字符串*、*数组和簇将数据分组*中的*编辑*、*格式化*、*解析字符串*和*数组函数*两节。

格式化文件以及将数据写入文件

Format Into File 函数可以将字符串、数字、路径和布尔数据格式化成文本，并将格式化以后的文本写入文件。该函数可一次实现多项功能，而无须先使用 Format Into String 函数来格式化字符串，然后使用 Write to Text File 函数将结果字符串写入文件中。

关于格式化字符串的详细信息见第 9 章*使用字符串*、*数组和簇将数据分组*中的*格式化和解析字符串*一节。

从文件中扫描数据

Scan From File 函数通过扫描文件中的文本，可以获取字符串、数值、路径和布尔值并将文本转换成某种数据类型。该函数可一次实现多项功能，而无须先使用 Read from Binary File 或 Read from Text File 函数来读取数据，然后使用 Scan From String 将结果扫描至文件中。

创建二进制文件

Write to Binary File 函数用于创建二进制文件。 Write to Binary File 函数的 **data** 输入端可与任何类型的数据连线。 Read from Binary File 函数用于指定读取文件中某种数据类型，只需将该数据类型的输入控件或常数与 Read from Binary File 函数的 **data type** 输入端相连。 Write to Binary File 和 Read from Binary File 函数可用于读取由不同操作系统创建的二进制文件。

创建数据记录文件

通过 **Datalog** 选板的数据记录 (Datalog) 函数可创建和读取数据记录文件，用于采集数据并将数据写入文件中。

无需格式化数据记录文件中的数据，但在读写数据记录文件时必须先指定数据类型。例如，采集带有时间和日期记录的温度读数时，将这些数据写入数据记录文件需要将该数据指定为包含一个数字和两个字符串的簇。关于向数

据记录文件写数据的范例信息见 `labview\examples\file\datalog.llb` 中的 Simple Temp Datalogger VI。

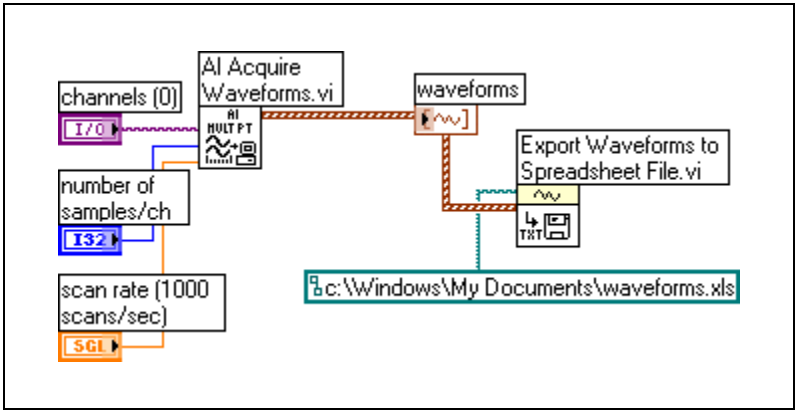
如果读取一个带有时间和日期记录的温度读数的文件，必须将所要读取的内容指定为包含一个数字和两个字符串的簇。关于读取数据记录文件的范例信息见 `labview\examples\file\datalog.llb` 中的 Simple Temp Datalog Reader VI。

将波形数据写入文件

Write Waveforms to File 和 Export Waveforms to Spreadsheet File VI 可以将波形写入文件。在电子表格、文本文件或数据记录文件中可以写入波形数据。

如果仅在 VI 中使用波形，可将该波形保存为数据记录文件 (`.log`)。

以下 VI 采集多个波形并在一个图形上进行显示，然后将这些波形数据写入电子表格文件中。

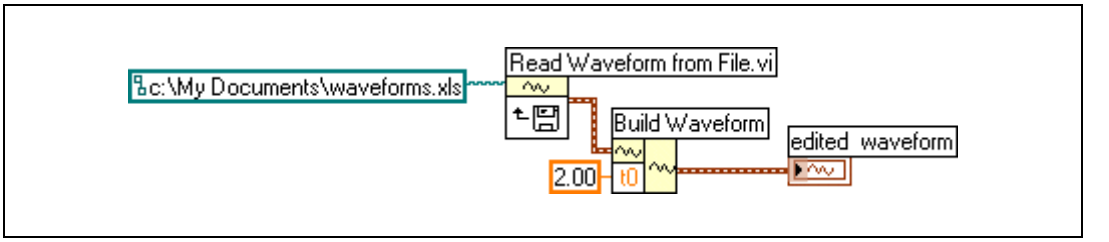


也可以使用 **Storage** 选板上的存储 VI 或 Write To Measurement File Express VI 将波形写入文件。

从文件中读取波形数据

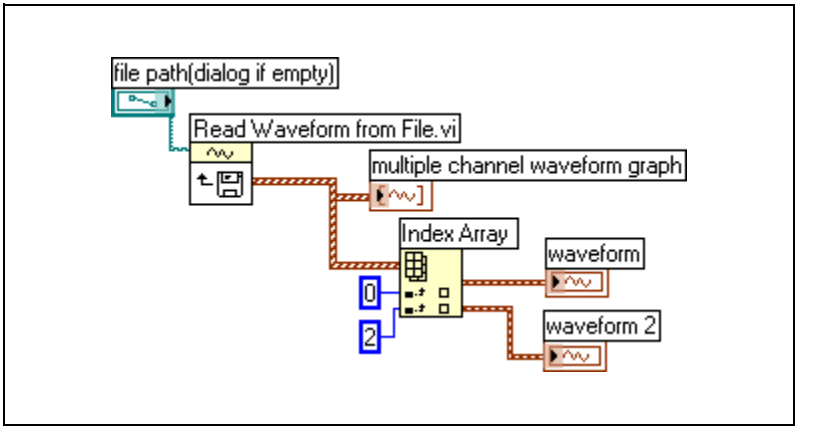
Read Waveform from File VI 可以从文件中读取波形。读取单个波形后，Build Waveform 函数可以添加或编辑波形数据元素，或使用 Get Waveform Attribute 函数提取波形属性。

下列 VI 可从文件中读取一个波形，编辑波形的 **t0** 元素，并将编辑后的波形绘制在图形上。



Read Waveform from File VI 也可以从文件中读取多个波形。该 VI 返回一个波形数据数组，可以在多曲线图形中显示。如需访问文件中的单个波形，必须为该波形数据类型数组建立索引，如以下程序框图所示。该 VI 可访问含有多个波形的文件。Index Array 函数读取文件中的第一和第三个波形，并将它们绘制在两个独立的波形图上。

关于索引数组的详细信息见第 9 章 *使用字符串、数组和簇将数据分组*中的 *数组*一节。关于波形图的详细信息见第 10 章 *图形和图表*的 *波形图*一节。



也可以使用 **Storage** 选板上的存储 VI 或 Read From Measurement File Express VI 从文件中读取波形。

12

编制 VI 文档和打印 VI

LabVIEW 可用来编制 VI 文档和打印 VI。

编制了 VI 文档，便可以在开发过程中的任何一个阶段记录程序框图和前面板的信息。

一些打印 VI 的选项可用于打印 VI 的相关信息，而另一些则用于打印 VI 生成的数据和结果报表。是要通过手动方式打印还是通过编程方式打印，报表格式所用选项的多少，是否需要在独立可执行应用程序中设置打印功能，是在何种平台上运行 VI，这些因素都会影响所使用的打印方式。

编制 VI 文档

LabVIEW 可为已完成的 VI 编制文档并且为 VI 用户创建操作说明。在 LabVIEW 中可以查看和打印该文档，并将其保存为 HTML、 RTF 或文本文件。

要创建有效的 VI 文档，应创建 VI 和对象的说明。

为 VI 及其对象 （如输入控件和显示控件）创建说明，来描述 VI 或对象的目的，向用户提供使用说明。在 LabVIEW 中可以查看说明并将其打印出来，还可以将说明保存为 HTML、 RTF 或文本文件。

选择 **File»VI Properties**，然后从 **Category** 下拉菜单中选择 **Documentation**，可以创建、编辑和查看 VI 说明。右键单击对象并从快捷菜单中选择 **Description and Tip**，也可以创建、编辑和查看对象说明。提示框 (Tip strips) 是一种简短的说明，在 VI 运行时将鼠标移到某个对象上，提示框就会出现。如果在 **Description and Tip** 对话框中没有输入提示，则不会出现提示框。将鼠标分别移到 VI 图标或对象上时，该 VI 或对象的说明会在 **Context Help** 窗口中出现。

注 您不能为**函数**选板中的 VI 或函数输入说明。

打印 VI

用户主要可以使用以下方法来打印 VI：

- 选择 **File»Print Window** 来打印活动窗口的内容。
- 选择 **File»Print** 打印有关 VI 的更多全面的信息，这些信息包括前面板、程序框图、子 VI、控件、 VI 历史等。
- 通过编程方式打印一个 VI 窗口，或者通过编程方式打印或保存一个包含 VI 文档或 VI 所返回数据的报表。

选择 **File»Print** 可以打印 VI 文档或将其保存为 HTML、RTF 或文本文件。可以选择内置的文档格式，也可以创建自定义文档格式。所创建的文档可以包含以下这些条目：

- 图标和接线器 (Icon and connector pane)
- 前面板和程序框图 (Front panel and block diagram)
- 输入控件、显示控件和数据类型接线端 (Controls, indicators, and data type terminals)
- 输入控件和显示控件的标签和标题 (Labels and captions)
- VI 和对象说明 (VI and object descriptions)
- VI 层次结构 (VI hierarchy)
- 子 VI 列表 (List of subVIs)
- 修订历史 (Revision history)

注　为某些类型的 VI 所创建的文档可能不会包括上述所有条目。例如，由于多态 VI (polymorphic VI) 没有前面板或程序框图，因此为多态 VI 所创建的文档中不能包括这两个条目。

LabVIEW 生成的 HTML 或 RTF 文件可用来创建用户编译好的帮助文件。**(Windows)** 可以将 LabVIEW 生成的单个 HTML 文件编译成 HTML 帮助文件。 **(Mac OS)** 可以将 LabVIEW 生成的单个 HTML 文件用于 Apple 帮助文件。

可以将 LabVIEW 生成的 RTF 文件编译成 **(Windows)** WinHelp 或者 **(Linux)** HyperHelp 文件。

A

技术支持及专业服务

如需更多关于技术支持及专业服务的内容，请访问 National Instruments 网站 ni.com。

- **技术支持**—关于在线技术支持请登录网站ni.com/support，网站包括以下内容：
  - **自助资源库** — 请访问 National Instruments 网站，查阅有关软件驱动及更新、在线知识库、产品使用手册、故障解答向导、数千个实例程序、产品教程、应用指南、仪器驱动等相关信息。
  - **免费技术支持** — 所有注册用户将免费获得基础服务，其中包括在 ni.com/exchange 的 NI 开发交流平台上与上百位应用工程师进行技术交流。National Instruments 的应用工程师将确保您所有的问题得以解答。

    如需了解更多当地的技术支持服务，请访问 ni.com/services 或 ni.com/contact 与当地办事处联系。
- **培训及认证** — 请访问 ni.com/training 查阅有关定制培训、电子教学虚拟课堂、互动 CD 及认证项目信息。同时可在全球各地报名参加面授课程。
- **系统集成** —National Instruments Alliance Partner 可为您解决时间限制、内部技术资源短缺或其它项目问题。详情请致电当地 NI 办事处或登录网站 ni.com/alliance。
- **合规声明 (DoC)**— 通过生产商合规声明，本合规声明确保产品符合欧盟理事会相关产品规范。本系统保证电磁兼容性 (EMC) 和产品安全性。请访问 ni.com/certification 获取产品的合规声明。
- **校准证书** — 如您所购产品支持校准服务，可访问 ni.com/calibration 以获取校准证书。

如您在公司网站 ni.com 未能查到所需信息，请与 NI 总部或当地办事处联系。请在本手册首页查询全球办事处电话号码。您也可以访问全球办事处网页 ni.com/niglobal 查找最新的办事处联系方式、技术支持电话、电子邮件地址及最新消息。

词汇表

符号	前缀	值
y	yocto	10^{-24}
z	zepto	10^{-21}
a	atto	10^{-18}
f	femto	10^{-15}
p	pico	10^{-12}
n	纳 (nano)	10^{-9}
μ	微 (micro)	10^{-6}
m	毫 (milli)	10^{-3}
c	centi	10^{-2}
d	deci	10^{-1}
da	deka	10^{1}
h	hecto	10^{2}
k	千 (kilo)	10^{3}
M	兆 (mega)	10^{6}
G	千兆 (giga)	10^{9}
T	太 (tera)	10^{12}
P	peta	10^{15}
E	exa	10^{18}
Z	zetta	10^{21}
Y	yotta	10^{24}

数字 / 符号

∞	无穷。
Δ	Delta ；差分。 Δ*x* 表示 *x* 从一个指标变为另一个指标的改变量。
π	Pi。

1D	一维。
2D	二维。
3D	三维。

A

A	安培。
ASCII	美国标准信息交互码。

B

编辑模式 (edit mode)	可以对 VI 进行修改的状态。
编译 (compile)	将高级代码转换为机器可执行代码的过程。在用户创建或编辑修改 VI 后，LabVIEW 在首次运行这些 VI 前将自动进行编译。
标绘图 (plot)	数据阵列在图形或图表上的图形表示。
标量 (scalar)	由刻度上的某一点所表示的数值，标量是相对于数组的单个值。标量布尔值和簇均为其各自对应数据类型的单个实例。
标签 (label)	用于命名、描述前面板或程序框图中的对象或区域的文字对象。
标签工具 (Labeling tool)	创建标签并向文本窗口输入文本的工具。
表示 (representation)	数值数据类型的子类型，其中包括 8 位、 16 位和 32 位有符号和无符号整数，以及单精度、双精度、扩展精度和浮点数。
波形 (waveform)	以特定采样率采集的多个电压读数的集合。
波形图表 (waveform chart)	以特定速率绘制数据点的显示控件。
布尔输入控件和显示控件 (Boolean controls and indicators)	前面板对象，用于操作和显示布尔数据 (TRUE 或 FALSE)。

C

采样 (sample)	某一次模拟或数字输入或输出的数据点。

菜单栏 (menu bar)　列出了程序中的主菜单名。菜单栏在标题栏下方。有些菜单和命令对所有的程序都适用，但一般每个程序都有其特定的菜单栏。

操作工具 (Operating tool)　在控件中输入数据或对控件进行操作的工具。

操作员 (operator)　执行程序操作和监控的人。

测量设备 (measurement device)　指某个 DAQ 设备如 E 系列多功能 I/O (MIO) 设备、 SCXI 信号调理模块、开关模块等。

层次结构窗口 (Hierarchy window)　*见* VI 层次结构窗口 (VI Hierarchy window)。

层叠式顺序结构 (Stacked Sequence structure)　以数字顺序执行子程序框图的程序控制结构。如果无法使用数据流参数 (flow-through parameter) 时，可以通过顺序结构控制不存在数据依赖性的节点的执行顺序。层叠式顺序结构每次仅显示其中一帧，并按照顺序执行所有帧。

常量 (constant)　常量 (constants) 是程序框图上向程序框图提供固定数据值的接线端。*见* 通用常量 （universal constant）和用户自定义常量 （user-defined constant)。

程序框图 (block diagram)　程序或算法的图形化表示。程序框图由可执行图标 （即节点）和在节点间传送数据的连线组成。程序框图即为 VI 的源代码，位于 VI 的程序框图窗口。

程序实例 (application instance)　LabVIEW 创建的程序实例，用于 LabVIEW 项目中的各个终端。在 **Project Explorer** 窗口打开 VI 时，该 VI 将在用于此终端的程序实例中打开。 LabVIEW 也会创建一个主要应用实例，其中包括不属于项目的已打开 VI 或未从项目中打开的 VI。*见* 终端 (target)。

出错信息 (error message)　说明软件或硬件存在故障，或者是输入了无法接收的数据时的说明信息。

触发 (trigger)　引发或启动任何形式数据采集的事件。

簇 (cluster)　一组有序的、未索引的任何数据类型的数据元素，这些数据类型包括数值、布尔、字符串、数组或是簇等。类似于 C 语言中的结构。这些元素必须同为输入控件或同为显示控件。

簇空壳 (cluster shell)　包含簇元素的前面板对象。

错误簇 (error cluster)　包含一个布尔状态显示控件、一个数字代码显示控件和一个字符串的显示控件。

错误输出 (error out)	输出 VI 的错误结构。
错误输入 (error in)	进入 VI 的错误簇。

D

DAQ	*见* 数据采集 (data acquisition， DAQ) 或 NI-DAQ。
DAQ Assistant	配置测量任务、通道和刻度的图形化界面。
DAQ 设备 (DAQ device)	用于采集或生成数据的设备，可以包含多个通道和转换设备。 DAQ 设备包括插入式驱动器、 PCMCIA 卡和 DAQPad 设备，均连接到计算机的 USB 或 1394 （Firewire®）端口。 SCXI 模块也属于 DAQ 设备。
带状图表 (strip chart)	模仿纸带图表记录器的数值绘图显示控件，并随着绘制数据而滚动显示。
当前 VI (current VI)	前面板、程序框图或图标编辑器处于活动状态的 VI。
点 (point)	以两个 16 位的整数分别代表横坐标和纵坐标的簇。
调整圆圈或手柄大小 (resizing circles or handles)	位于对象边框上的圆圈或手柄用于表示可调整该对象大小的点。
顶层 VI (top-level VI)	位于 VI 层次结构中最顶层的 VI，该术语用于区分 VI 和子 VI。
定位工具 (Positioning tool)	移动对象、改变对象大小的工具。
断点 (breakpoint)	用于调试时暂停执行。
断点工具 (Breakpoint tool)	在 VI、节点或连线上设置断点的工具。
断开的 VI(broken VI)	发生错误导致无法运行的 VI ；在断开的**运行 (Run)** 按钮中用断开的箭头表示。
断开的**运行**按钮 (broken **Run** button)	发生错误导致 VI 无法运行时，该按钮会代替**运行 (Run)** 按钮。
对话框 (dialog box)	当一个应用程序需要更多信息来执行命令时出现的窗口。
对象 (object)	前面板和程序框图上条目的统称，包括输入控件、显示控件、节点、连线以及导入的图片。
多态 (polymorphism)	节点可根据不同的数据形式、类型、结构进行自动调节的功能。

E

Express VI	用于进行常规测量任务的子 VI。 Express VI 是在配置对话框中配置的。

F

For 循环 (For Loop)	按规定次数执行子程序的交互式循环结构。相当于文本代码：`For i = 0 to n − 1, do...`
范围 (range)	测量、接收、传输某一量值的上下限度范围。用上限值和下限值表示。
方法 (method)	对象接收信息时的执行过程。方法通常与类相关。
符号 (glyph)	小的图片或图标。
复选框 (checkbox)	在对话框中的小的方形框，可以选中或不选中。复选框一般用于在多个选项中进行选择。可以选中多个复选框。

G

G	LabVIEW 使用的图形化编程语言。
GPIB	*见* 通用接口总线 （General Purpose Interface Bus）。
工具 (tool)	特定的光标工具，可用于实现特定操作。
工具栏 (toolbar)	工具栏包含各种命令按钮，可用于运行和调试 VI。
工具选板 (**Tools** palette)	程序中包含工具集合的一个选板，该选板中的工具可用于编辑和调试前面板和程序框图中的对象。

H

hex	十六进制。十六进制计数系统。
函数 (function)	内置的执行元素，相当于文本编程语言中的操作符、函数或语句。
函数选板 (**Functions** palette)	包含 VI、函数、程序框图结构和常量的选板。
滑块 (slider)	输入控件或显示控件上的可移动部分，滑块的移动反映数值的改变。
缓冲器 (buffer)	存储采集或生成的数据的临时存储设备。

活动窗口 (active window)　表示当前设为接收用户输入的窗口，通常为最前端的窗口。活动窗口的标题栏是高亮显示的。单击一个窗口或从**窗口**（**Windows**）菜单中选择一个窗口，可以使该窗口成为活动窗口。

I

I/O　输入 / 输出。数据在计算机系统的输入输出，包括通讯通道、操作输入装置、数据采集和控制接口等。

`Inf`　以浮点表示无穷的数字化表示。

IVI　可互换虚拟仪器。为常规测试和测量仪器创建共同接口 (API) 的软件标准。

J

即时帮助窗口 (**Context Help** window)　当光标移动到每一个 LabVIEW 对象上时，显示该对象基本信息的窗口。VI、函数、常数、结构、选板、属性、方式、事件、对话框和 **Project Explorer** 中的项目均包含即时帮助信息。

计数接线端 (iteration terminal)　For 循环或 While 循环的接线端，包含当前已完成的循环的次数。

建模 (prototype)　以简单快速的方式实施一项任务以确定是否可行。建模通常功能不全面或者设计有漏洞。一般而言模型最终应当摒弃，应当在正式版本中重新创建功能。

接线端 (terminal)　用于传递数据的节点端口。

接线器 (connector pane)　前面板或程序框图窗口右上角区域，显示 VI 接线端模式。它定义了可以连接到 VI 的输入和输出端。

节点 (node)　程序执行元素。相当于文本编程语言中的语句、运算、函数和子程序。在程序框图中，节点包括函数、结构和子 VI。

结构 (structure)　程序控制元素，如平铺式和层叠式顺序结构 (Flat Sequence structure, Stacked Sequence structure)、条件结构 (Case structure)、 For 循环 (For Loop) 或 While 循环 (While Loop) 等。

矩形 (rectangle)　包含 4 个 16 位整数的簇。前两个值确定左上角的横纵坐标位置。后两个值确定右下角的横纵坐标位置。

矩阵 (matrix)　由可表示线性方程中各项系数的数字或其它数学元素组成的矩形阵列。

句柄 (handle)　指向一块内存的指针的指示器，表示引用数组和字符串。一个字符串数组是一块包含字符串句柄的内存的句柄。

K

刻度 (scale)　图形、图表和数值输入控件和显示控件的组成部分，包含一连串固定间隔的标识，用于表示测量单位。

空数组 (empty array)　具有零个元素但是已经定义好数据类型的数组。例如，一个在数据显示窗口具有数值控件但是没有为任何元素定义值的数组为空数组。

控件选板 (**Controls** palette)　包含前面板中输入控件、显示控件和修饰型对象的选板。

控制流 (control flow)　指令先后顺序决定执行顺序的编程系统。大多数文本编程语言为控制流语言。

库 (library)　*见* LLB 或项目库。

快捷菜单 (shortcut menu)　右键单击某个对象时出现的菜单。快捷菜单只作用于其所属的对象。

捆绑函数 (bundle function)　捆绑各种元素来创建簇 (clusters) 的函数。

L

LabVIEW　Laboratory Virtual Instrument Engineering Workbench。LabVIEW 是一种图形化编程语言，用图标代替文本行来创建程序。

LED　发光二极管，指示灯。

LLB　包含用于特定用途的相关 VI 集合的 LabVIEW 文件。

类 (class)　包含属性、方法和事件的一个类别。类被排列成一个层次结构，每个类继承上一级别类的属性和方法。

离散 (discrete)　具有独立变量的不连续值，通常为采样时间。

连接器 (connector)　VI 或函数节点的一部分，包含输入和输出接线端。数据通过连接器在节点中进行传输。

连线 (wire)　节点间的数据路径。

连线段 (wire segment)　单个水平或垂直连线段。

连线分支 (wire branch)　连线分支包含所有的从交叉点到交叉点、接线端到交叉点或中间没有交叉点的接线端到接线端的连线段。

连线工具 (Wiring tool)　用于连接接线端之间数据路径的工具。

连线接线头 (wire stubs)　当连线工具移至 VI 或函数节点上时，在未连接的接线端旁所出现的线头。

列表框 (listbox)　对话框内的一个框，列出一个命令的所有可选项。例如，一个磁盘上的一列文件名。

M

MAX　*见* Measurement & Automation Explorer。

Measurement & Automation Explorer　Windows 平台下的标准 NI 硬件配置和分析环境。

脉冲 (pulse)　振幅瞬时从无到有，该信号称为脉冲。

默认 (default)　预设值。许多 VI 输入在没有用户指定值的情况下将使用默认值。

目录 (directory)　将文件方便地分组的一种结构模式。目录以地址的形式显示文件的位置。目录可以包含文件或文件子目录。

N

NaN　<Not A Number>，指在不确定的浮点运算中产生的结果。一般用于不确定的操作结果，如 log(−1)。

NI-DAQ　所有 NI-DAQ 设备和信号调理组件的驱动程序。 NI-DAQ 是一个可在应用程序开发环境 (ADE) 如 LabVIEW 中调用的 VI 和 ANSI C 函数扩展程序库，用于配置所有 NI 测量设备，如 M 系列多功能输入输出 (MIO) DAQ 设备、信号调理模块和开关模块。

NI-DAQmx　最新的 NI-DAQ 驱动程序，带有控制测量设备所需的最新 VI、函数和开发工具。 NI-DAQmx 在以下方面优于前期版本的 NI-DAQ：在 LabVIEW、 LabWindows/CVI 和 Measurement Studio 中 DAQ Assistant 对设备通道设置和测量任务的支持；更优异的性能如单点模拟 I/O 更加快速、创建 DAQ 程序的 API 更为简洁并可使用更少的函数和 VI。

P

PXI	PCI eXtensions for Instrumentation (PCI 在仪器领域的扩展)。基于计算机的模块化仪器平台。
配置程序工具 (configuration utility)	指 Windows 中的 Measurement & Automation Explorer 和 Mac OS 或 Linux 中的仪器配置工具。
频率 (frequency)	f，速率的基本单位，使用频率计数器或频谱分析仪测量每秒所发生的事件或振动。频率是信号周期的倒数。
平铺式顺序结构 (Flat Sequence structure)	以数字顺序执行子程序框图的程序控制结构。如果无法使用数据流参数 (flow-through parameter) 时，可以通过顺序结构控制不存在数据依赖性的节点的执行顺序。平铺式顺序结构一次显示所有帧，并按照从左到右的顺序执行所有帧，直到执行完最后一帧。

Q

前面板 (front panel)	VI 的交互式用户界面。前面板外观类似于物理仪器 （如示波器和万用表）。
强度图 (intensity map/plot)	使用颜色在一个二维图中显示数据三维的方法。
强制转换 (coercion)	LabVIEW 执行的一种自动转换，用于改变一个数据元素的数值表示法。
强制转换点 (coercion dot)	出现在程序框图节点上，以警示两个不同数字数据类型的数据被连线连在了一起。任何数据类型被连线连接到一个变体数据类型时也会出现强制转换点。
驱动程序 (driver)	用于控制硬件设备 （如 DAQ 设备）的软件。
驱动器 (drive)	`a-z` 范围内的一个字母再加一个冒号 (`:`)，用于表示逻辑磁盘驱动器。

R

二维 (two-dimensional)	包括 2 个维数，类似于包含多个行或列的数组。
人工数据依赖性 (artificial data dependency)	在数据流编程语言中的条件，即根据数据到达传输终点 （而不是数据值）触发一个节点的执行。

S

扫描图表 (sweep chart)　以示波器操作为模型的数值显示控件。与示波器图类似，但扫描图中包括一条垂直线用于分隔新旧数据。

上色工具 (Coloring tool)　设置前景色和背景色的工具。

设备 (device)　可以作为一个实体访问的仪器或控制器，用于控制或监控现实世界的输入输出端口。设备通常通过一些通讯网络连接到主机。*见* DAQ 设备 (DAQ device) 和测量设备 (measurement device)。

示波器图表 (scope chart)　以示波器操作为模型的数值显示控件。

事件 (event)　模拟或数字信号的条件或状态。

输入控件 (control)　以交互方式向 VI 输入数据或以编程方式向子 VI 输入数据的前面板对象，如旋纽、按钮或转盘。

数据采集 (data acquisition, DAQ)　1. 通过传感器、采集传感器和测试探针或测试装置采集并测量模拟或数字电子信号。

2. 生成模拟或数字电子信号。

数据记录 (datalog)　采集数据并同时将数据存储于磁盘文件中。 LabVIEW 文件输入 / 输出 (File I/O)VI 和函数可用于记录数据。

数据记录文件 (datalog file)　将数据存为一系列单一的任意数据类型记录 （在创建文件时指定）的文件。一个数据记录文件中的所有记录都必须为同一数据类型，但该类型也可以是复数类型。例如，可以指定每条记录为包含一个字符串、一个数值和一个数组的簇。

数据类型 (data type)　信息格式。 LabVIEW 中绝大多数 VI 和函数接收的数据类型包括数值 (numeric)、数组 (array)、字符串 (string)、布尔 (Boolean)、路径 (path)、引用 (refnum)、枚举 (enumeration)、波形 (waveform) 和簇 (cluster) 等。

数据流 (data flow)　由可执行节点组成的编程系统，这些可执行节点只有在接收到所有必需的输入数据后才会开始运行。节点在执行时将自动生成输出数据。LabVIEW 就是一个数据流系统。数据流经节点的动作决定了程序框图上 VI 和函数的执行顺序。

数据依赖性 (data dependency)　数据流编程语言的条件，一个节点只有在从其它节点接收数据后才能执行。*见* 人工数据依赖性 (artificial data dependency)。

数值输入控件和显示控件 (numeric controls and indicators)　用于管理和显示数值数据的前面板对象。

数组 (array)　按一定顺序带索引的同类数据元素列表。

数组壳 (array shell)　包含数组的前面板对象。一个数组空壳包括一个索引显示、一个数据对象窗口和一个可选标签。它接收多种数据类型。

顺序结构 (sequence structure)　*见* 平铺式顺序结构 (Flat Sequence structure) 或层叠式顺序结构 (Stacked Sequence structure)。

隧道 (tunnel)　在结构中的数据输入或输出接线端。

T

探针 (probe)　用于检查 VI 中数值的调试功能。

探针工具 (Probe tool)　在连线上设置探针的工具。

条件分支 (case)　条件结构的一个子程序框图。

条件接线端 (conditional terminal)　While 循环 （While Loop）的接线端，包含一个决定 VI 是否再执行下一次循环的布尔值。

条件结构 (Case structure)　条件控制结构，根据输入的条件选择执行几个子程序框图中的一个。它综合了控制流语言中的 IF、 THEN、 ELSE 和 CASE 语句。

通道 (channel)　1. 物理通道 (Physical) －用于测量和发生模拟信号或数字信号的接线端或管脚。一个单一的物理通道可以包括多个端口，如一个差分模拟输入通道或一个八条数字线的端口。一个计数器也可以是一个物理通道，但计数器与其对应的接线端采用不同的命名。

2. 虚拟通道 (Virtual) －包括名称、物理通道、输入终端连接、测量或发生信号类型及刻度信息在内的一组属性设置。定义 NI-DAQmx 的虚拟通道可以在任务外 （全局）或任务内 （局部）。在传统 NI-DAQ 或更早的版本中配置虚拟通道是可选的，但对于在 NI-DAQmx 中进行的每个测量却是必不可少的。传统 NI-DAQ 是在 MAX 中配置虚拟通道的。而在 NI-DAQmx 中，虚拟通道既可以在 MAX 中也可以在用户程序中配置，可以将通道配置成任务的一部分也可以独立配置通道。

3. 开关通道 (Switch) －开关通道表示开关上的任意连接点。根据开关的拓扑结构，开关通道可能由一条或多条信号线组成 （通常为一条、两条或四条）。开关通道无法用来创建虚拟通道。开关通道只能在 NI-DAQmx 开关函数和 VI 中使用。

通用常量 (universal constant)　可传出特定 ASCII 字符或标准数值常量的程序框图对象，并且无法进行修改，如 π。

通用接口总线 (General Purpose Interface Bus)　GPIB。与 HP-IB 同义。通过计算机控制电子仪器的标准总线。也称 IEEE 488 总线，因为它是根据 ANSI/IEEE 488-1978、 488.1-1987 和 488.2-1992 标准进行定义的。

图标 (icon)　程序框图上节点的图形化表示。

图表 (chart)　一个或多个图形的二维显示，该显示保留了原来数据的历史记录，最多可以是用户定义的最高值。图表接收数据并且以逐个点或逐个数组的方式更新显示，在缓冲器中保留一定数目的数据点以便于显示。*见* 示波器图表 (scope chart)、带状图表 (strip chart) 和扫描图表 (sweep chart)。

图例 (legend)　图形或图表的示例，用于显示在该图形或图表上的图形名称和样式。

图形 (graph)　一个或多个曲线 （或标绘图）的二维显示。图形用于接收并绘制数据块。

图形控件 (graph control)　在直角坐标系平面中显示数据的前面板对象。

拖放 (drag)　用屏幕上的光标执行选择、移动、复制或删除对象的操作。

V

VI　*见* 虚拟仪器 (virtual instrument)。

VISA　*见* 虚拟仪器软件架构。

VI 层次结构窗口 (**VI Hierarchy** window)　图形化显示 VI 和子 VI 的层次结构的窗口。

W

While 循环 (While Loop)　一种循环结构，重复执行某一代码段直至条件满足时才停止。

维数 (dimension)　数组的大小和结构。

无法显示的字符 (non-displayable characters)　无法显示的 ASCII 字符，如 null、 backspace、 tab 等。

X

下拉菜单 (pull-down menus)	单击菜单栏中菜单所出现的菜单条目。下拉菜单中的项目通常为某一类菜单的集合。
下拉列表控件 (ring control)	特殊的数值控件，包括 32 位整数 （从 0 开始并按顺序递增）以及一连串文本标签或图形。
显示控件 (indicator)	显示输出的前面板对象，如图形或指示灯。
项目 (project)	包括一些 LabVIEW 文件和非 LabVIEW 文件，可用于说明如何生成程序、为终端调配项目或进行配置设置。
项目库 (project library)	LabVIEW 项目库包括 VI、自定义类型、共享变量、模板目录文件以及其它文件和项目库等。
项目浏览 (Project Explorer) 窗口	创建和编辑 LabVIEW 项目的窗口。
像素 (pixel)	数码图像的最小单位。
向导 (wizard)	包含多个连续页面的对话框，可以翻动页面，并在相关页面中填写相关信息。
虚拟仪器 (virtual instrument, VI)	LabVIEW 编制的程序，用于模仿物理仪器的外观和功能。
虚拟仪器软件架构 (Virtual Instrument Software Architecture)	VISA. 用于控制 GPIC、 VXI、 RS-2232 和其它类型仪器的单个接口库。
选板 (palette)	包含用于创建前面板和程序框图所需的对象和工具的面板。
选取框 (marquee)	环绕所选对象的活动虚线框。

Y

一维 (one-dimensional)	只有一个维数，如数组中仅有一行元素。
移位寄存器 (shift register)	循环结构中的可选机制，可将变量值从一次循环传递至下一次循环中。移位寄存器相当于编程语言中的静态变量。
仪器驱动程序 (instrument driver)	在系统中控制仪器硬件并与之通讯的一套实用函数。

引用句柄 (refnum)　查询编号 :LabVIEW 用以指代某一对象如 VI、程序、 ActiveX 或 .NET 对象的编号。引用句柄作为 VI 或函数的输入参数对对象进行操作。

应用程序 (application)　由 LabVIEW 开发系统创建、在 LabVIEW 运行时系统环境中执行的应用程序。

用户 (user)　*见* 操作员 (operator)。

用户自定义常量 (user-defined constant)　可传递用户预设值的程序框图对象。

语法 (syntax)　特定编程语言中语句必须遵从的规则集合。

源代码控制 (source control)　解决 VI 共享和访问权限控制中可能出现的问题，以防止数据意外丢失。可采用源控制方案在用户中实现文件共享，保障安全性，并对共享的项目进行修订记录。有时也称为 source code control。

运行模式 (run mode)　VI 正在运行或已被预设为运行的状态。单击前面板工具栏的 **Run** 或 **Run Continuously** 按钮、程序框图中的单步按钮或选择 **Operate»Change to Run Mode** 时，该 VI 可进入运行模式。当 VI 处于运行模式时，所有的前面板对象都有一个简化的快捷菜单项集合。 VI 在运行时无法对其进行编辑。

Z

整数 (integer)　包括任何自然数、自然数对应的负数和零。

帧 (frame)　平铺式或层叠式顺序结构的子程序框图。

执行过程突出显示 (execution highlighting)　动态显示 VI 执行过程以表明 VI 中数据流的一种程序调试技术。

指示框 (tip strip)　小型黄色文本框，显示接线端名称以便确定需连接的接线端。

制图 (picture)　制图显示控件同来创建图片的一系列作图指令。

终端 (target)　运行 VI 的设备或机器。必须通过 LabVIEW 项目在 RT、 FPGA 或 PDA 终端上进行操作。

转换 (conversion)　改变数据元素的类型。

子 VI(subVI)　程序框图中用于其它 VI 上的 VI，类似于子程序。

子程序框图 (subdiagram)　位于结构边框内的程序框图。

自动刻度调整（autoscaling）	刻度根据标绘数值的范围进行自动调整。在图形刻度中，自动刻度调整将确定刻度的最大和最小值。
自动索引 (auto-indexing)	循环结构在结构边框处分解和集合数组的能力。当带有自动索引功能的数组进入循环时，循环会自动分解数组，即从一维数组提取标量，而从二维数组处提取一维数组，依此类推。当数据离开循环时，循环以相反顺序将数据集合成数组。
自由标签 (free label)	前面板或程序框图中不隶属于任何对象的标签。
字符串 (string)	以文本形式来表示值。
字符串输入控件和显示控件 (string controls and indicators)	用于处理和显示文本的前面板对象。
总数接线端 (count terminal)	一个 For 循环 （For Loop）的接线端，其值决定了该 For 循环执行子程序框图的次数。

索引

A-D

E-G

H-K

L-O

P-S

T-W

X-Z