

LabVIEW™ 核心教程（一） 教程手册

教程软件版本 2012

2012 年 12 月

325290D-0118

版权

© 1993–2012 National Instruments. 版权所有。

根据版权法、未经 National Instruments Corporation 事先书面同意、本发行物不得以任何形式（包括电子或机械形式）进行全部或部分复制或传播、包括影印、录制、储存于任何信息检索系统中、或翻译。

National Instruments 公司尊重他方的知识产权、也恳请我们的用户能给予同样的尊重。NI 软件受版权和其他知识产权法律的保护。当 NI 软件被用来生产复制属于他方的软件或其他资料时、请确保您仅在符合任何有效许可证条款或其他法律限制的前提下、以 NI 软件生产复制该资料。

商标

LabVIEW, National Instruments, NI, ni.com, National Instruments 公司标识, 以及鹰形标识均为 National Instruments Corporation 的商标。关于其它 National Instruments 商标, 请访问 ni.com/trademarks 参考 *Trademark Information*。

此处所提及的其它产品和公司名称为其各自公司的商标或商业名称。

National Instruments Alliance Partner Program 的成员为独立于 National Instruments 的商业实体、与 National Instruments 无代理、合伙或合资关系。

最终用户许可证协议和第三方法律声明

可在下列位置找到最终用户许可证协议和第三方法律声明：

- 法律声明位于 <National Instruments>_Legal Information 和 <National Instruments> 目录。
- 最终用户许可协议位于 <National Instruments>\Shared\MDF\Legal\license 目录。
- 关于在使用 NI 产品生成的安装程序中包含法律信息的细则, 请参考 <National Instruments>_Legal Information.txt 文件。

专利权

关于 National Instruments 产品和技术的专利权、见软件中的**帮助»专利信息**、光盘上的 patents.txt 文档、或登录 ni.com/patents 查看 *National Instruments Patent Notice*。

全球技术支持及产品信息

ni.com/china

全球办事处

请通过 ni.com/niglobal 访问各个分公司的网址，获取最新的联系方式、技术支持电话、Email 地址、当前活动等信息。

National Instruments 总部

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

如需更多关于技术支持的信息，请查阅“更多信息和资源”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。

目录

学员指南

A. NI 认证	v
B. 教程概述	v
C. 学习本教程之前的准备工作	vi
D. 安装教程软件	vii
E. 教程目标	vii
F. 教程的行文规范	viii

第 1 课

LabVIEW 导航

A. LabVIEW 是什么	1-2
B. 项目浏览器	1-4
C. VI 的组成	1-7
D. 前面板	1-11
E. 程序框图	1-16
F. 搜索控件、VI 和函数	1-25
G. 选择工具	1-27
H. 数据流	1-34
I. 创建一个简单 VI	1-35

第 2 课

疑难解答和调试 VI

A. 纠正断开的 VI	2-2
B. 调试技巧	2-3
C. 未定义或未预期的数据	2-7
D. 错误处理	2-7

第 3 课

实现 VI

A. 前面板基本介绍	3-2
B. LabVIEW 数据类型	3-3
C. 注释代码	3-11
D. While 循环	3-13
E. For 循环	3-16
F. VI 定时	3-19
G. 循环中的数据反馈	3-19
H. 数据图表绘制一波形图表	3-22
I. 条件结构	3-24

第 4 课

开发模块化应用程序

A. 理解模块化概念	4-2
B. 创建图标和连线板	4-3
C. 使用子 VI	4-6

第 5 课

创建和使用数据结构

A. 数组	5-2
B. 常见数组函数	5-4
C. 多态性	5-5
D. 自动索引	5-7
E. 簇	5-11
F. 自定义类型	5-14

第 6 课

管理文件和硬件资源

A. 硬件和软件资源	6-2
B. 文件 I/O	6-3
C. 使用 DAQ 系统采集测量	6-6
D. 仪器控制	6-12

第 7 课

使用顺序和状态机算法

A. 顺序编程	7-2
B. 状态编程	7-4
C. 状态机	7-4

第 8 课

通过变量解决数据流问题

A. 并行循环间通信	8-2
B. 写入输入控件和读取显示控件	8-3
C. 变量	8-4
D. 竞争状态	8-10

词汇表

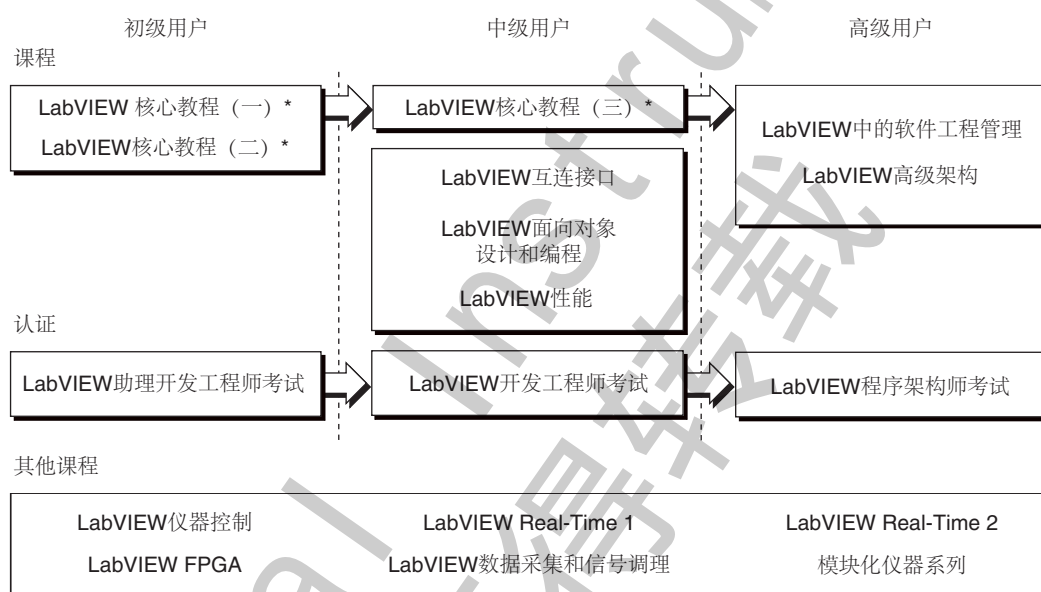
学员指南

非常感谢您购买 *LabVIEW 核心教程（一）* 课程套件。在您完成教程中的练习后，就可以开发应用程序了。在为期 3 天的面授课程 *LabVIEW 核心教程（一）* 中，将使用本教程手册和配套软件。

如在 90 天内参加培训，可将购买教材的费用抵扣相应的培训费用。关于课程时间表、教程大纲、培训中心和课程报名的详细信息请登录 ni.com/training 查询。

A. NI 认证

LabVIEW 核心教程（一） 是 NI 培训系列课程中的一门，该系列课程可帮助您熟练掌握 LabVIEW 和通过 NI 认证的 LabVIEW 助理开发工程师考试。下图为 LabVIEW 培训系列课程介绍。关于 NI 认证的更多信息见 ni.com/training。



如要获得开发 LabVIEW 的最佳效果，应优先学习星号（）核心教程。

B. 教程概述

LabVIEW 核心教程（一） 包括 LabVIEW 的编程理论、技巧、功能、VI 和函数，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。阅读本教程前请先熟悉 Windows 和使用流程图或程序框图编写算法的方法。教程和练习手册由若干节课组成。

教程手册的每节课由下列部分组成：

- 本课的学习目的和学习内容的介绍
- 本课主题的概述
- 用于测试和强化课程概念和技巧的总结测验

练习手册的每节课由下列部分组成：

- 巩固课程主题的练习
- 有些课程还包含一些有一定难度的选作练习或一套附加练习。时间允许的情况下，学员可以选作这些练习。



注 关于教程和练习手册的更新和修正信息，请登录 ni.com/info，输入信息码 Core1 查询。

某些练习需使用下列 NI 硬件产品：

- 连接 BNC-2120 的即插式多功能数据采集 (DAQ) 设备，DAQ 信号附件包含温度传感器、函数发生器和 LED 显示灯
- 连接 NI 仪器仿真器的 GPIB 接口

即使无法获取上述硬件，您仍可以完成教程中的练习。教程包含不使用硬件完成练习的相关说明。您也可以使用其他硬件替代上述硬件设备。例如，NI 仪器仿真器或连接信号源的 NI DAQ 设备（函数发生器）可使用 GPIB 仪器替代。

C. 学习本教程之前的准备工作

阅读建议

建议的阅读资料可帮助学员粗略了解 *LabVIEW 核心教程（一）* 的主要理论和概念。如要获得最佳培训效果，请完成阅读全部推荐资料。

如要获得下列推荐的阅读资料，可登录 ni.com/info，输入主题的相应信息代码：

- ☐ *LabVIEW 核心课程（一）—软件开发规范*（信息代码：SoftDev）
- ☐ *数据采集概述*（信息代码：DAQ）
- ☐ *GPIB 仪器控制指南*（信息代码：GPIB）

教材

在学习本教程手册前，请确保如下几条：

- ☐ 计算机操作系统为 Windows 7/Vista/XP
- ☐ 多功能 DAQ 设备，在 Measurement & Automation Explorer(MAX) 中配置为 Dev1
- ☐ BNC-2120 和线缆
- ☐ GPIB 接口
- ☐ NI 仪器仿真设备和电源
- ☐ LabVIEW 2012 完整版或专业版开发系统或更高版本
- ☐ DAQmx 9.5.5 或更高版本
- ☐ NI-488.2 3.0.2 或更高版本
- ☐ NI VISA 5.2 或更高版本
- ☐ GPIB 线缆
- ☐ 通过 NI 仪器模拟器软件光盘安装的 NI 仪器模拟器向导

□ *LabVIEW 核心教程（一）教程光盘*，其中包括下列文件夹：

目录	说明
Exercises	用于保存学员在学习教程中创建的 VI 和完成的练习；以及一些练习中需要调用到的子 VI 和压缩文件 (NI Instrument Simulator.zip)，此压缩文件包含用于 NI 仪器模拟器的 LabVIEW 仪器驱动程序
Solutions	包含所有教程练习的解答

D. 安装教程软件

请按以下步骤安装教程软件：

1. 将教程光盘插入电脑光驱。屏幕上会自动弹出 **LabVIEW 核心教程（一）设置** 的对话框。
2. 单击 **安装教程资料**。
3. 按照屏幕提示完成安装和配置。

练习文件位于 <Exercises>\LabVIEW Core 1\ 文件夹。



注 尖括号中的文件夹名称（例如，<Exercises>）表示文件夹位于计算机的根目录。

E. 教程目标

本教程的目标是帮助您掌握以下内容：




- 理解前面板、程序框图、图标和连线板的概念
- 学会使用 LabVIEW 的编程结构和数据类型
- 学会使用各种各样的编辑和调试技巧
- 创建和保存 VI，以便作为子 VI 调用
- 显示和记录数据
- 创建使用即插式 DAQ 设备的应用程序
- 创建使用串口和 GPIB 设备的应用程序

本教程不包括以下内容：

- 每个内置 VI、函数或者对象；教程中未提及的 LabVIEW 功能，请参考 *LabVIEW 帮助*
- 模数转换 (A/D) 的原理
- GPIB 总线操作
- 开发仪器驱动程序
- 为学员开发一个完整的应用程序；学员可单击 **帮助 » 查找范例**，通过“NI 范例查找器”查找并使用现有的范例 VI

F. 教程的行文规范

本教程的行文规范如下：

»	» 表示通过嵌套菜单和对话框选项作出最终选择。 工具»仪器»查找仪器驱动 表示打开 工具 菜单，选择 仪器 菜单项，最后选择 查找仪器驱动 选项。
	该提示符号提醒用户注意参考信息。
	该提示符号提醒用户注意重要信息。
	该警告符号提醒您采取预防措施以防受伤、数据丢失或系统崩溃。
粗体	粗体文本表示软件中的必选项（例如，菜单和对话框选项）。粗体字表示对话框或硬件的名称。
<i>斜体</i>	斜体文本表示变量、强调、交叉引用或重要概念介绍。同时也可作为占位符，表示须由用户填写的文字或数值。
等宽字体	等宽字体文本表示用户从键盘输入的文字、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、函数、运算、变量、文件名和扩展名的命名。
等宽粗体	等宽粗体文本表示在计算机屏幕上自动显示的消息和响应。该字体也用于强调与其它范例不同的代码行。
平台字体	平台字体文本表示特定的平台，随后的文本内容仅适用于该平台。

LabVIEW 导航

本节介绍了浏览 LabVIEW 环境的方法，其中包括使用菜单、工具栏、选板、工具、帮助和常见 LabVIEW 对话框。同时，您将学习运行 VI 并形成对前面板和程序框图的大致理解。本节结束时，您将创建一个获取、分析和显示数据的简单 VI。

主题

- A. LabVIEW 是什么
- B. 项目浏览器
- C. VI 的组成
- D. 前面板
- E. 程序框图
- F. 搜索控件、VI 和函数
- G. 选择工具
- H. 数据流
- I. 创建一个简单 VI

A. LabVIEW 是什么

LabVIEW 是一种图形化编程环境，通过它能够快速方便的创建具有专业用户界面的应用程序。数以百万计的工程师和科学家通过 LabVIEW 的直观图标和连线，开发复杂的测量、测试及控制系统应用程序。此外，LabVIEW 平台可在不同的终端和 OS 之间切换。事实上，LabVIEW 提供了与上千种硬件设备间的无缝集成及上百种用于高级分析和数据可视化的内置库，用户可使用内置的库创建自定义需求的虚拟仪器。

LabVIEW 程序称为虚拟仪器 (VI)，它的外观和操作类似于真实的物理仪器（例如，示波器和万用表）。VI 具有前面板和程序框图。用户界面在 LabVIEW 中被称为前面板。程序框图是指在用户界面后台的编程。前面板创建完毕后，便可使用图形化的函数添加源代码来控制前面板上的对象。程序框图上的代码为图形化代码，也称为 G 代码或程序框图代码。

与文本编程语言（C++ 和 Visual Basic）相比，LabVIEW 使用图标代替文本行创建应用程序。在文本编程中，由指令决定程序的执行次序。LabVIEW 使用图形化数据流编程。在图形化数据流编程中，由数据流经程序框图上节点的顺序决定程序的执行次序。图形化编程和数据流执行是 LabVIEW 与其他通用编程语言的两大主要区别。

在本教程中，您将学习使用 LabVIEW 有效的创建一个简单的数据采集应用程序，该程序包含采集、分析和显示数据三个步骤。尽管本教程是在 Windows 系统中教授的，但 LabVIEW 为多平台软件。用户可在 Windows、Mac OS 或 Linux 系统上开发应用程序。此外，用户还可部署 LabVIEW 应用程序至多个 Real-Time 和 FPGA 终端。

LabVIEW 特性

LabVIEW 编程具有下列特性：

- 图形化编程和编译特性
- 数据流和 / 或基于事件的编程
- 多终端和多平台特性
- 面向对象的灵活性
- 多线程可能性

在 LabVIEW 核心教程（一）中，您将了解 LabVIEW 的图形化编程和编译特性及数据流编程。下列教程将介绍其他特性：

- LabVIEW 核心教程（二）—介绍基于事件的编程。
- LabVIEW 面向对象设计和编程—介绍 LabVIEW 面向对象编程。
- LabVIEW 性能—介绍利用 LabVIEW 的多线程和内存管理，以增强程序的可执行性和内存使用率。

图形化编程和编译

程序框图上的 G 代码是一种使用图表和连线代替文本的图形化语言，它与多数传统编程语言包含相同的编程理念。例如，G 代码包含数据类型、循环、事件处理、变量、递归及面向对象的编程。LabVIEW 直接将 G 代码编译为计算机处理可执行的机器代码，无需单独的 G 代码编译步骤。

数据流和事件驱动编程

LabVIEW 编程是依据数据流编程规则执行，而不是采用多数文本编程语言（例如，C 和 C++）的执行方法。数据流执行由数据驱动，又称为数据依赖。G 代码节点间的数据流决定了执行的次序。

事件驱动编程功能扩展了 LabVIEW 的数据流环境，它允许用户直接与程序进行交互，而无需轮询。基于事件的编程同时还允许其他异步活动影响程序框图上 G 代码的执行。

多终端和多平台

通过 LabVIEW 应用程序，用户可使用多核处理器和其他并行硬件。例如，现场可编程门阵列 (FPGA)。LabVIEW 应用程序可被自动调整以适用于 2 核、4 核或更多核 CPU，通常无需进行额外的编程处理。

除少数基于平台的函数外，G 代码可以在不同操作系统下的 LabVIEW 中运行。因此，同一代码可在 Windows、Mac OS X 或 Linux 系统平台下的 LabVIEW 中运行。

面向对象

在大量编程语言中，面向对象的编程是一种热门的编程方法。大量类似但不同的项在软件中被表示为对象的类。LabVIEW 提供了工具和函数，以在 G 代码中使用面向对象编程技术。

多线程和内存管理

LabVIEW 允许代码自动进行并行处理。在其他编程语言中，用户需要手动管理线程以并行运行代码。在编译器和执行系统共同工作的 LabVIEW 中，任意可能的情况下都将自动并行运行代码。多数情况下，执行系统的详细信息对用户并不重要。因为执行系统在无需介入的情况下执行了正确的操作，但 LabVIEW 也提供了改进性能的选项。

启动 LabVIEW 环境

打开 LabVIEW 软件时，启动窗口如图 1-1 所示。

图 1-1. LabVIEW 启动窗口



通过启动窗口创建新的项目和 VI。您可选择新建项目或通过模板和范例创建项目。用户也可以打开现有的 LabVIEW 文件及访问 LabVIEW 社区资源和帮助。

新建文件或打开现有文件后，启动窗口消失。关闭全部前面板和程序框图后，启动窗口会再次显示。单击查看 » 启动窗口可随时显示该窗口。

B. 项目浏览器

VI 为 LabVIEW 程序，用户可使用多个 VI 创建 LabVIEW 应用程序。使用 LabVIEW 项目将与应用程序关联的 VI 组合在一起。通过**项目浏览器**窗口保存 LabVIEW 项目时，LabVIEW 创建一个项目文件 (.lvproj)，其中包含项目中全部 LabVIEW 文件和非 LabVIEW 文件的引用、配置信息、生成信息和部署信息。

某些 LabVIEW 应用程序（例如，简单 VI）不需要使用 LabVIEW 项目。但生成独立应用程序和共享库时必须使用 LabVIEW 项目。此外，与非开发机器终端（例如，RT、FPGA 和 PDA 终端）配合使用时必须使用 LabVIEW 项目。关于项目与 LabVIEW Real-Time、FPGA 和 PDA 模块配合使用的详细信息，见指定模块文档。

项目浏览器窗口

LabVIEW 项目包括 VI、保证 VI 正确运行的必需文件，以及其他支持文件（例如，文档或相关链接）。使用**项目浏览器**窗口管理 LabVIEW 项目。

默认情况下，**项目浏览器**窗口包括以下各项：

- **项目根目录**—包含**项目浏览器**窗口中所有其他项。项目根目录的标签包括该项目的文件名。
- **我的电脑**—表示可作为项目终端使用的本地计算机。
- **依赖关系**—包含终端下 VI 必需的项。
- **程序生成规范**—包括对源代码发布编译配置以及 LabVIEW 工具包和模块所支持的其他编译形式的配置。如已安装 LabVIEW 专业版开发系统或应用程序生成器，可使用**程序生成规范**配置独立应用程序、共享库、安装程序和 zip 文件。



提示 终端是指运行 VI 的设备。

在项目中添加其他终端时，LabVIEW 会在**项目浏览器**窗口中新建一个表示该终端的项。每个终端还包括**依赖关系**和**程序生成规范**。在每个终端下可添加文件。

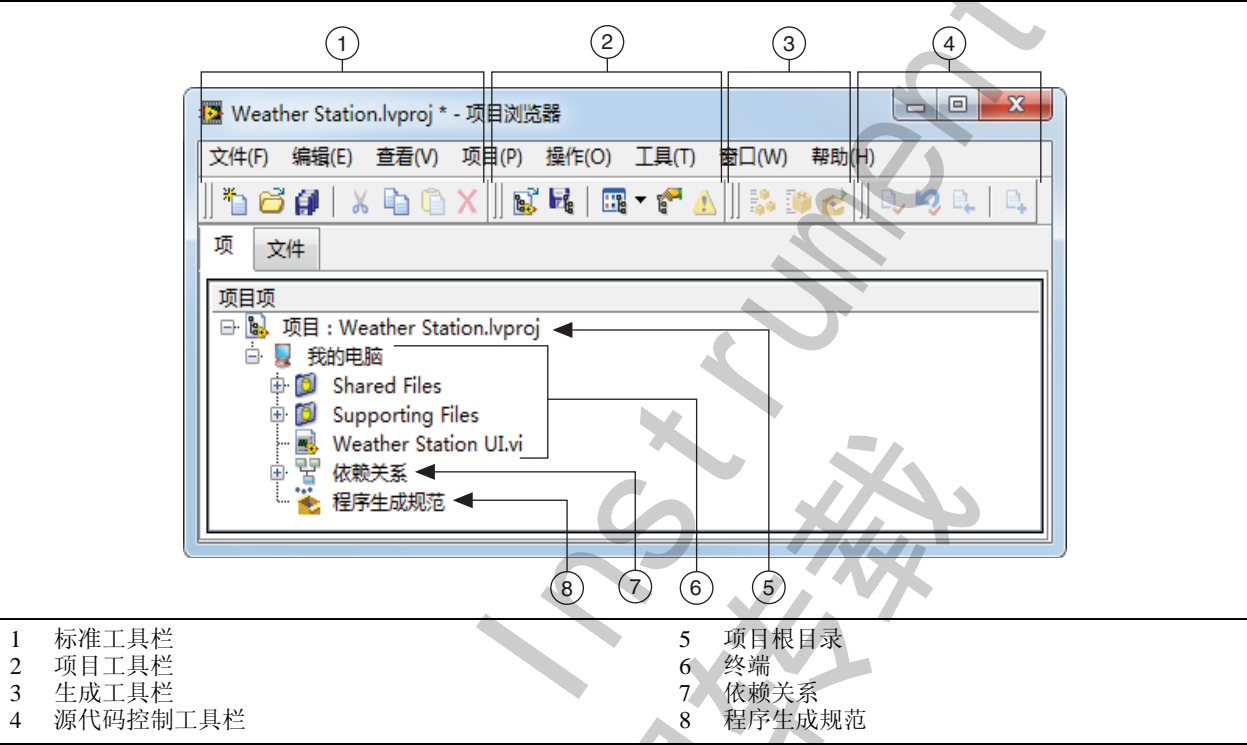
按照下列步骤使用**项目浏览器**窗口创建和编辑 LabVIEW 项目。

1. 选择**文件 » 创建项目**打开**创建项目**对话框。**创建项目**对话框包含模板和范例项目的列表，用户可使用这些模板和范例创建具有可靠设计和编程方法的项目。
2. 搜索与您的编程目标相近的模板或范例。通过下列功能搜索模板和范例：
 - **过滤**—显示某种类型的结果。例如，用于特定终端的模板或范例。
 - **附加搜索**—搜索过滤后结果的关键词、标题和说明。
 - **详细信息**—打开相关项的帮助文件。阅读帮助文件，确保模板或项目和您的编程目标相吻合。
3. 单击**下一步**或**完成**按钮配置项目的细节，包括项目的保存方式。配置项目完成后，LabVIEW 将保存项目，并打开**项目浏览器**窗口和项目的顶层 VI。
4. 使用**项目浏览器**窗口修改项目。关于如何修改项目，请参考项目中 VI 的程序框图。或者参考**项目浏览器**窗口的 **Project Documentation** 文件夹，查看修改项目的详细信息。

项目相关工具栏

通过**标准**工具栏按钮、**项目**工具栏按钮、**生成**工具栏按钮和**源代码控制**工具栏按钮执行 LabVIEW 项目的操作。工具栏位于**项目浏览器**窗口顶端，如图 1-2 所示。有时需展开**项目浏览器**窗口才能查看所有工具栏按钮。

图 1-2. 项目浏览器窗口



提示 源代码控制工具栏仅当 LabVIEW 中配置了源代码时可用。

单击**查看**»**工具栏**选择要显示或隐藏的工具栏。或者右键单击工具栏的空白区域显示或隐藏所需的工具栏。

创建 LabVIEW 项目

按照下列步骤新建一个项目。

1. 选择下列操作之一新建 LabVIEW 项目。
 - 在**启动**窗口，单击**项目**新建一个空白项目或单击**创建项目**按钮，通过模板创建项目。
 - 在打开的 LabVIEW 项目或 VI 中，选择**文件**»**创建项目**。
 - 在任意 LabVIEW 窗口，选择**文件**»**新建**。然后在**新建**对话框内选择**项目**»**项目**打开**项目浏览器**窗口。
2. 添加项至项目的终端。
3. 选择**文件**»**保存项目**保存项目。

添加文件至项目

用户可添加已有文件至项目。通过**项目浏览器**窗口中的**我的电脑**项（或其他终端）添加文件（例如，VI 或文本文件）至 LabVIEW 项目。

通过下列方式可添加项至项目：

- 右键单击**我的电脑**，从快捷菜单中选择**添加 » 文件**添加文件。或者从**项目浏览器**菜单选择**项目 » 添加至项目 » 文件**添加文件。
- 右键单击**终端**，从快捷菜单中选择**添加 » 文件夹（自动更新）**添加自动更新的文件夹。或者选择**项目 » 添加至项目 » 添加文件夹（自动更新）**添加自动更新的文件夹。LabVIEW 将根据对项目 and 磁盘进行的改动，实时监控和更新文件夹。
- 右键单击**终端**或**我的电脑**，从快捷菜单中选择**添加 » 文件夹（快照）**添加虚拟文件夹。或者选择**项目 » 添加至项目 » 添加文件夹（快照）**添加虚拟文件夹。选择磁盘上的目录，LabVIEW 将在项目中创建与磁盘目录同名的虚拟文件夹。LabVIEW 还将创建代表目录下全部内容的项目项（包括子目录的文件和内容）。选择磁盘上的文件夹将添加全部文件夹内容，包括子目录的文件和内容。



注 添加磁盘上的虚拟文件夹至项目后，除非用户更改磁盘上的文件夹，否则 LabVIEW 不会自动更新该文件夹。

- 右键单击**终端**，从快捷菜单中选择**新建 » VI**添加一个空白 VI 至终端。或者选择**文件 » 新建 VI 或项目 » 添加至项目 » 新建 VI**添加一个空白 VI。
- 在前面板或程序框图窗口选中 VI 图标，拖曳该图标至终端。
- 在计算机文件系统中选择项或文件夹并拖曳至终端。

删除项目中的项

通过下列方式可删除**项目浏览器**窗口中的项：

- 右键单击要删除的项，从快捷菜单中选择**从项目中删除**。
- 选中要删除的项并按下 <Delete> 键。
- 选择要删除的项并单击**标准**工具栏上的**从项目中删除**按钮。



注 删除项目的项不会删除相应的磁盘项。

组织项目中的项

项目浏览器窗口包括两个页面：**项**和**文件**。**项**页面用于显示项目目录树中的项。**文件**页面显示在磁盘上有相应文件的项目项。在该页上可对文件名和文件夹进行管理。**文件**页面的项目操作将同时影响和更新磁盘上项的内容。右键单击终端下的某个文件夹或项并从快捷菜单中选择**在项视图中显示**或**在文件视图中显示**，可在这两个页面间进行切换。

使用文件夹管理**项目浏览器**窗口下的项。用户可添加两种类型文件夹至 LabVIEW 项目：虚拟文件夹和自动更新文件夹。虚拟文件对项目项进行管理。右键单击**项目浏览器**窗口中的终端，从快捷菜单中选择**新建 » 虚拟文件夹**创建一个虚拟文件夹。自动更新文件夹通过实时更新来反映磁盘上文件夹的内容。在项目中添加自动更新文件夹，以磁盘上文件的形式查看项目项。

自动更新文件夹仅在**项目浏览器**的**项**页面上可见。在该页面上可查看自动生成文件夹的磁盘内容，但无法对其进行重命名、重新组织或删除项目项的磁盘操作。在**项目浏览器**窗口的**文件**页面可对自动更新文件夹中的项进行磁盘操作。**文件**页面显示了项目文件夹在磁盘上的位置。**文件**中对项目进行的操作将影响并更新磁盘上对应的文件。同样地，如修改了磁盘上 LabVIEW 以外的文件夹，LabVIEW 将对项目中的自动更新文件夹进行自动更新。

用户可管理文件夹中的项。右键单击一个文件夹并从快捷菜单选择**排列 » 名称**，按字母顺序排列项。右键单击一个文件夹并从快捷菜单选择**排列 » 类型**，按文件类型排列项。

查看项目中的文件

添加文件至 LabVIEW 项目后，LabVIEW 将把对该文件的引用保存到磁盘上。在**项目浏览器**窗口右键单击一个文件，从快捷菜单中选择**打开**。文件将使用默认的编辑器打开。

右键单击项目，从快捷菜单中选择**查看 » 完整路径**，查看项目引用文件在磁盘上的保存位置。

使用**项目文件信息**对话框查看项目引用文件在磁盘上的位置及在**项目浏览器**窗口的对应项。单击**项目 » 文件信息**显示**项目文件信息**对话框。或者右键单击项目，从快捷菜单中选择**查看 » 文件信息**显示**项目文件信息**对话框。

保存项目

通过下列方式保存 LabVIEW 项目：

- 单击**文件 » 保存项目**。
- 选择**项目 » 保存项目**。
- 右键单击项目，从快捷菜单中选择**保存**。
- 在**项目**工具栏单击**保存项目**按钮。

项目中必须有新的未保存的文件，才可保存项目。保存项目时，LabVIEW 并不会把依赖关系作为项目文件的一部分进行保存。



注 如打算对项目进行重大版本更新，请创建备份副本。

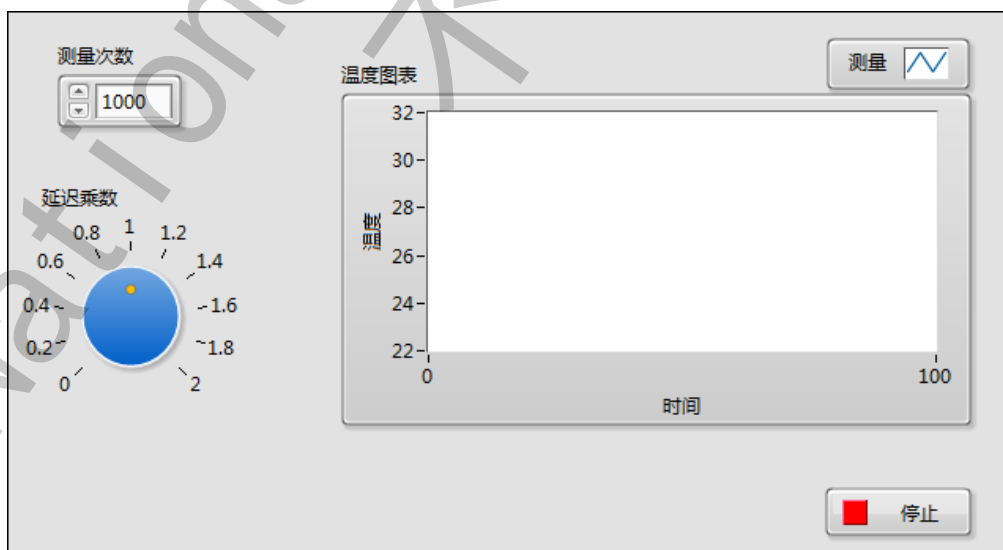
C. VI 的组成

LabVIEW VI 包含三个主要组成部分—前面板窗口、程序框图和图标 / 连线板。

前面板窗口

前面板窗口是 VI 的用户界面。图 1-3 为前面板窗口的范例。输入控件和显示控件用于创建前面板，它们分别是 VI 的交互式输入和输出接线端。

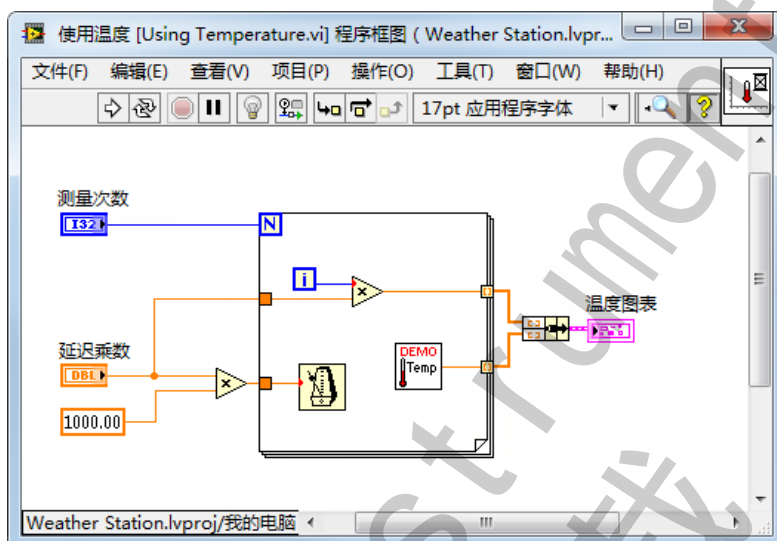
图 1-3. VI 前面板



程序框图窗口

前面板创建完成后，使用图形化表示的函数添加代码来控制前面板上的对象。图 1-4 为程序框图窗口的范例。程序框图窗口包含图形化源代码。前面板上的对象在程序框图中显示为接线端。

图 1-4. 程序框图



图标和连线板

图标和连线板允许用户在另一个 VI 中使用和查看 VI。用于其他 VI 内部的 VI 称为子 VI，其与文本编程语言中的函数类似。如要将 VI 用作子 VI，其必须具有图标和连线板。

每个 VI 在前面板和程序框图窗口的右上角显示图标。默认图标的范例如下所示。图标是 VI 的图形化表示。图标可同时包含文本和图像。如果将一个 VI 用作子 VI，程序框图上将显示代表该子 VI 的图标，默认图标中有一个数字，表明 LabVIEW 启动后打开新 VI 的个数。



如要将 VI 用作子 VI，用户需要创建连线板。如下图所示。连线板集合了 VI 各个接线端，与 VI 中的输入控件和显示控件相互呼应，类似文本编程语言中函数调用的参数列表。连线板位于前面板窗口右上角的 VI 图标旁边。在程序框图窗口不能访问连线板图标。



打开一个现有 VI

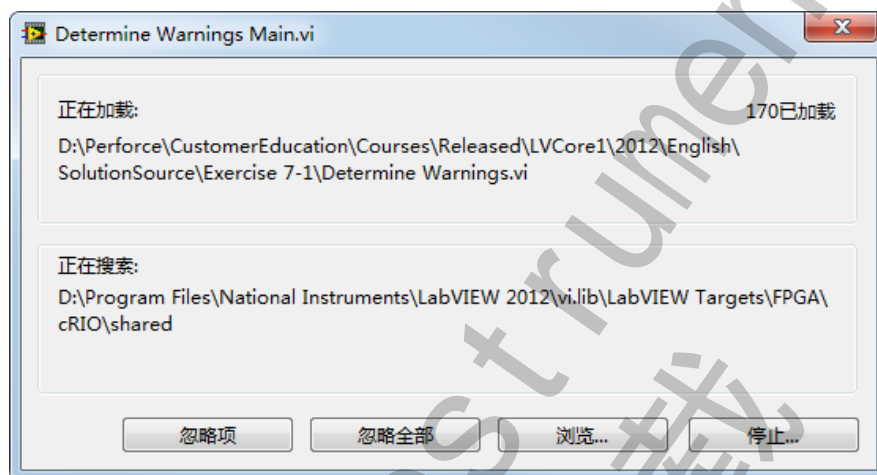
在启动窗口的**打开**列表中**浏览**，以打开现有的 VI。



提示 本教程中待编辑的 VI 位于 <Exercises>\LabVIEW Core 1 目录下。

VI 载入时将出现与下列范例类似的状态对话框。

图 1-5. 显示 VI 载入状态的对话框



载入状态将显示 VI 的子 VI 载入内存时的列表及当前载入内存的子 VI 数量。单击**停止**按钮可随时停止载入。

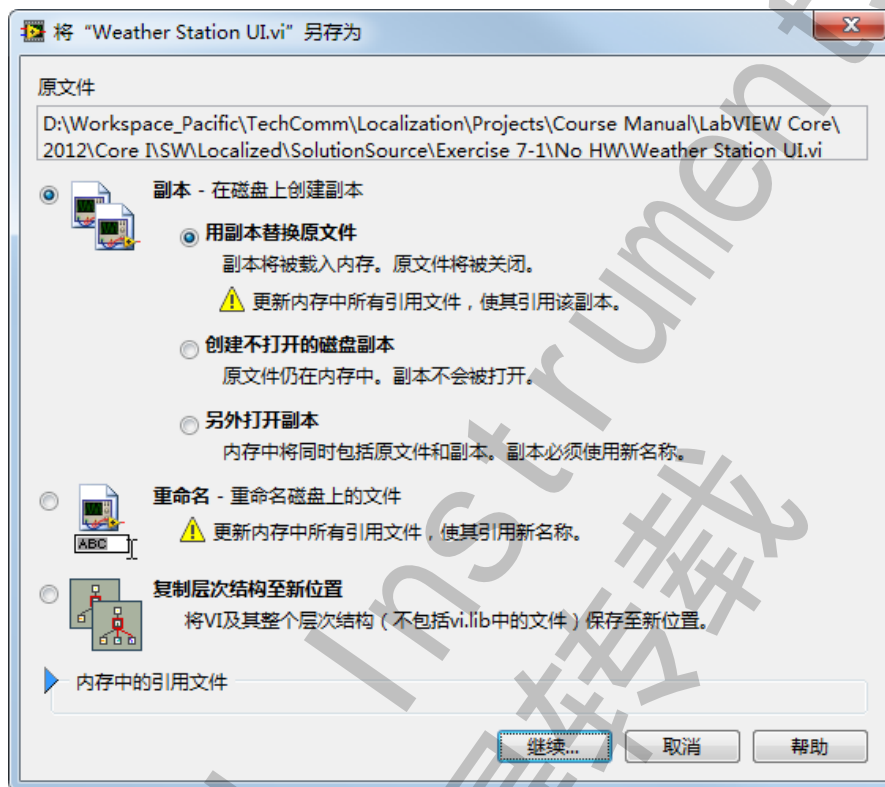
如 LabVIEW 不能立即定位子 VI，其将对 VI 搜索路径的全部目录进行扫描。单击**工具 » 选项**，从**类别**列表中选择**路径**可编辑 VI 的搜索路径。

载入时单击**忽略项**按钮，LabVIEW 将忽略该子 VI。或单击**浏览**按钮搜索丢失的子 VI。

保存 VI

如要保存新建 VI，单击**文件»保存**。如已保存 VI，单击**文件»另存为**打开**另存为**对话框。在**另存为**对话框中，创建 VI 的副本或删除原始 VI 并使用新的 VI 替换该 VI。

图 1-6. 另存为对话框

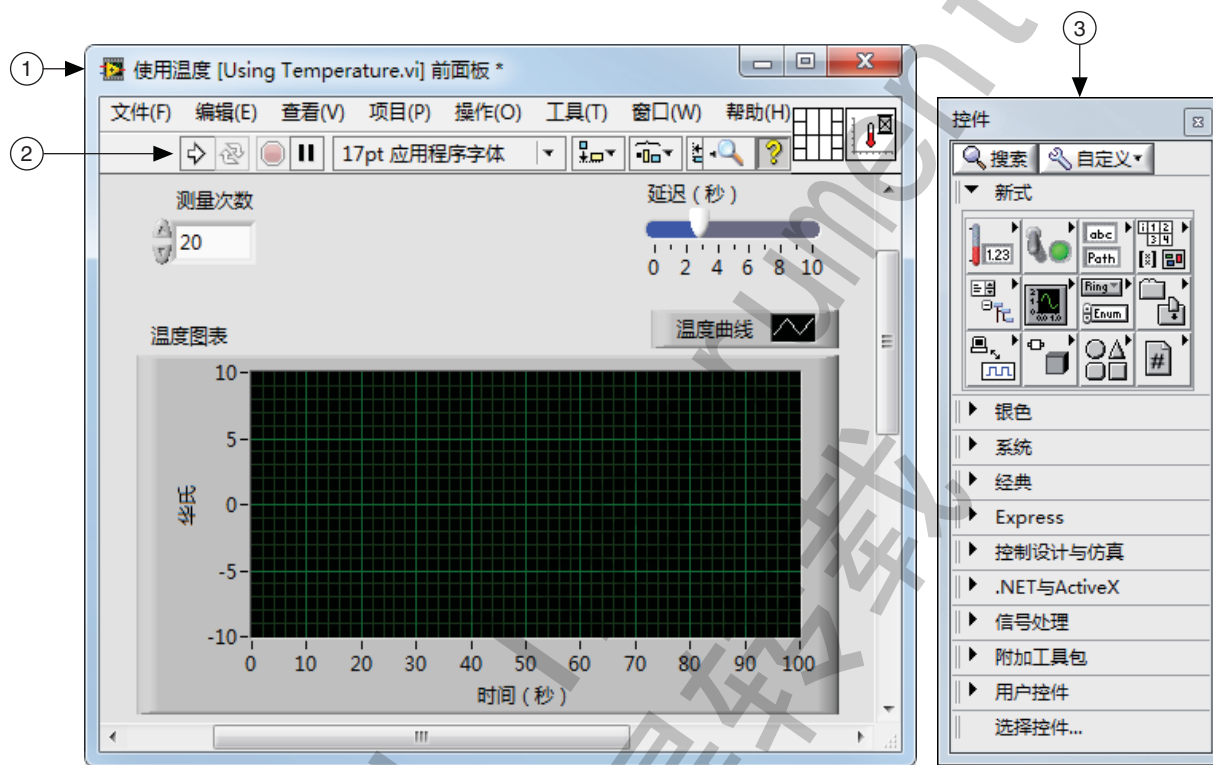


注 关于另存为对话框中选项的详细信息，见 *LabVIEW 帮助* 中的 *另存为对话框* 主题。

D. 前面板

打开新建或现有 VI 时，将出现 VI 的前面板窗口。前面板窗口是 VI 的用户界面。图 1-7 为前面板窗口的范例。

图 1-7. 前面板范例



1 前面板窗口

2 工具栏

3 控件选板

输入控件和显示控件

输入控件和显示控件用于创建前面板，它们分别是 VI 的交互式输入和输出接线端。输入控件指旋钮、按钮、转盘等输入设备。显示控件指图表、指示灯和其他显示设备。输入控件模拟输入设备并为 VI 的程序框图提供数据。显示控件模拟输出设备并显示程序框图获取或生成的数据。

图 1-7 具有下列对象：两个输入控件**测量次数**和**延迟（秒）**。一个显示控件：命名为**温度图表**的 XY 图。

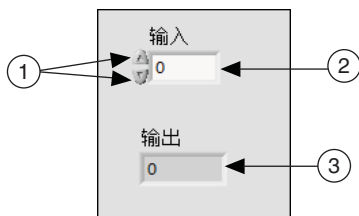
用户可更改**测量次数**和**延迟（秒）**输入控件的输入值。在**温度图表**显示控件中可查看 VI 生成的值。VI 根据程序框图创建的代码生成显示控件的值。详情见“数值控件”章节。

每个输入控件或显示控件均具有一个与其相关的数据类型。例如，**延迟（秒）**水平滑动杆为数值数据类型。常用的数据类型为数值、布尔值和字符串。详情见第 3 课—“实现 VI”。

数值控件

数值数据类型可表示不同类型的数值。例如，整数或实数。两个常用数值对象为数值输入控件和数值显示控件，如图 1-8 所示。诸如仪表和转盘之类的对象也表示数值数据。

图 1-8. 数值控件



- 1 增量 / 减量按钮
2 数值输入控件

- 3 数值显示控件

如要输入或更改数值控件的值，使用操作工具点击增量和减量按钮。或通过标签工具或操作工具双击数值，然后输入新的数值并按下 <Enter> 键。

布尔控件

布尔数据类型表示仅具有两种状态的数据。例如，真和假、开和关布尔控件用于输入及显示布尔值。布尔对象模拟开关、按钮和 LED 指示灯。垂直摇杆开关和圆形指示灯布尔对象如图 1-9 所示。

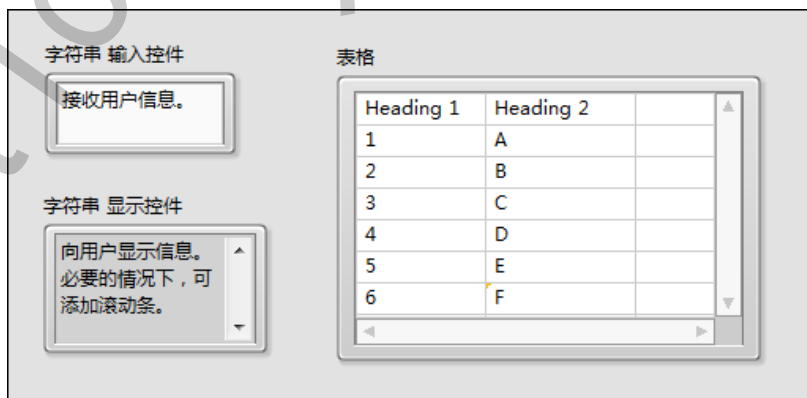
图 1-9. 布尔控件



字符串控件

字符串数据类型为 ASCII 字符序列。字符串输入控件用于从用户端接收诸如密码或用户名称等文本信息。字符串显示控件用于显示面向用户的文本信息。最常见的字符串对象为表格和文本输入框，如图 1-10 所示。

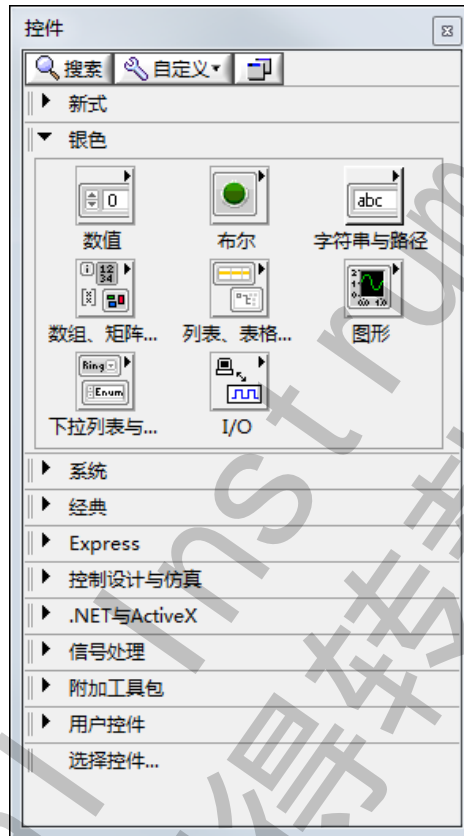
图 1-10. 字符串控件



控件选板

控件选板包括创建前面板所需的输入控件和显示控件。在前面板窗口单击**查看 » 控件选板**访问**控件**选板。**控件**选板被分为不同的类；用户可根据需要显示部分或全部类。图 1-11 为显示了全部类的**控件**选板，展开的控件类为**银色**。在本教程中，多数练习均使用**银色**选板类中的控件。

图 1-11. 控件选板



如要查看或隐藏类（子选板），可在选板上选择**自定义**按钮，选中或取消选中**更改可见选板**选项。

前面板窗口工具栏

每个窗口均带有相关的工具栏。通过前面板窗口工具栏按钮运行和编辑 VI。

前面板窗口的工具栏如下所示。



单击**运行**按钮运行 VI。如有需要，LabVIEW 将对 VI 进行编译。工具栏上的**运行**按钮显示为白色实心箭头时表示 VI 可以运行。如下图所示。如为 VI 创建连线板，白色实心箭头同时也表明可将该 VI 用作子 VI。



VI 运行时，如果是顶层 VI，**运行**按钮将如下图所示，表示不是子 VI，没有调用方。



如运行的是子 VI，**运行**按钮将如下图所示。



如正在创建或编辑的 VI 出现错误，则**运行**按钮显示为断开。如程序框图完成连线后，**运行**按钮仍显示为断开，则 VI 是断开的，无法运行。单击该按钮显示**错误列表**窗口，该窗口将列出全部错误和警告。



单击**连续运行**按钮，连续运行 VI 直至用户中止或暂停操作。再次单击该按钮可禁用连续运行。



VI 运行时，工具栏中将显示**中止执行**按钮。如无其他方式可中止 VI，单击该按钮立即停止 VI 的运行。多个运行中的顶层 VI 使用当前 VI 时，按钮显示为灰色。

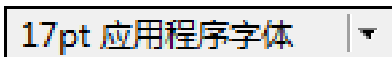


注意 中止运行按钮令 VI 在当前循环结束前立即停止运行。中止使用了外部资源（如外部硬件）的 VI 可能会导致外部资源无法恰当复位或释放并停留在一个未知状态。因此，应当为 VI 设计一个停止按钮以防这类问题的发生。

单击**暂停**按钮暂停运行中的 VI。单击**暂停**按钮时，LabVIEW 程序框图中暂停执行的位置将高亮显示，且**暂停**按钮显示为红色。再次单击**暂停**按钮继续 VI 的运行。



单击**文本设置**下拉菜单更改 VI 选中部分的字体设置（包括字体大小、样式和颜色）。



单击**对齐对象**下拉菜单对齐沿轴的对象，包括垂直中心、上边缘、左边缘等对齐方式。



单击**分布对象**下拉菜单将对象平均分布，包括间隔、压缩等。



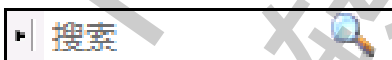
单击**调整对象大小**下拉菜单将多个前面板对象调整为相同的大小。



如有对象相互覆盖且用户想要定义对象的顶层和底层位置，可单击**重新排序**下拉菜单。使用定位工具选中其中一个对象，并选择**前移一层**、**后移一层**、**移至前面**或**移至后面**。



在 *LabVIEW 帮助* 中输入搜索项定位内容。



选择**显示即时帮助窗口**按钮切换是否显示**即时帮助**窗口。



为数值控件输入一个新的数值时，工具栏上会出现**确定输入**按钮，提醒用户只有按下 <Enter> 键，或在前面板或程序框图工作区单击鼠标，或单击**确定输入**按钮时，新数值才会替换旧数值。

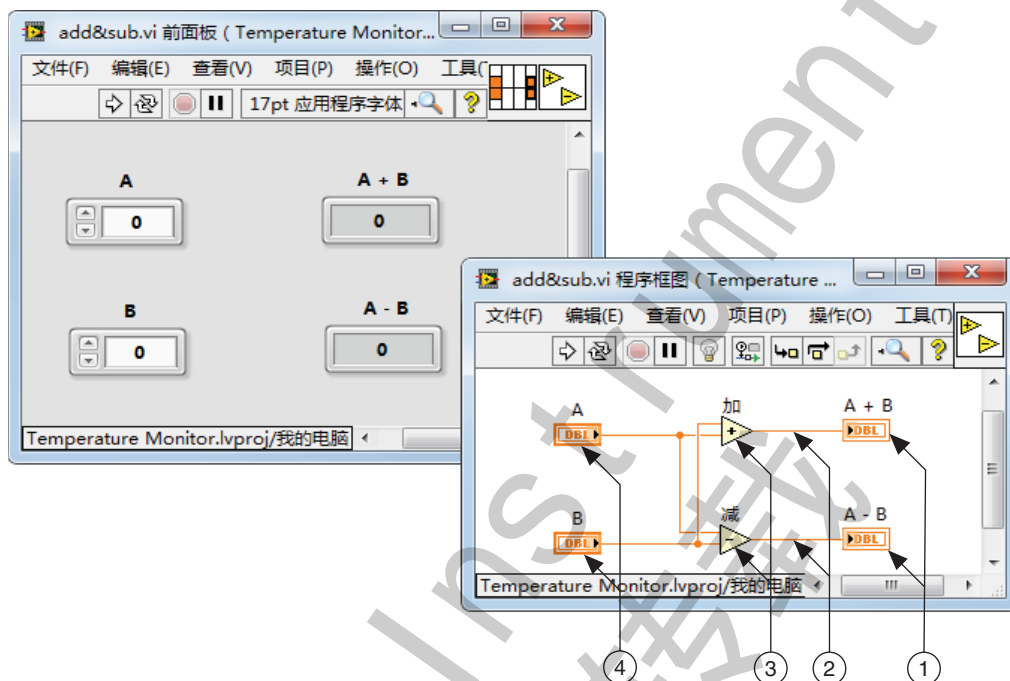


提示 数值键盘上的 <Enter> 键用于结束文本的输入，主键盘上的 <Enter> 键将增添新的一行。如要更改该操作，选择**工具»选项**，从**类别**列表中选择**环境**并勾选**使用回车键结束文本输入**选项。

E. 程序框图

程序框图对象包含接线端、子 VI、函数、常量、结构和连线，它们用于在其他程序框图对象间传输数据。

图 1-12. 程序框图和对应的前面板范例



1	2	3	4
显示控件接线端	连线	节点	控件接线端

接线端

前面板窗口的对象在程序框图上显示为接线端。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。接线端类似于文本编程语言的参数和常量。接线端类型包括输入控件或显示控件接线端和节点接线端。控件接线端属于前面板控件。用户在前面板控件输入的数据（上一个前面板的 **A** 和 **B**）通过控件接线端输入程序框图。然后，数据进入加和减函数。加减运算结束后，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（如上一前面板中的 **A+B** 和 **A-B**）。

图 1-12 中的接线端属于 4 个前面板控件。因为接线端表示 VI 的输入端和输出端，子 VI 和函数也具有如下的接线端。例如，“加”和“减”函数的连线板具有 3 个节点接线端。如要在程序框图上显示函数的接线端，右键单击函数节点并从快捷菜单中选择**显示项 » 接线端**。



输入控件、显示控件和常量

输入控件、显示控件和常量用作程序框图算法的输入端和输出端。考虑三角形面积的算法实现：

$$\text{面积} = 0.5 \times \text{底} \times \text{高}$$

在该算法中，**底**和**高**为输入，**面积**为输出。如图 1-13 所示。

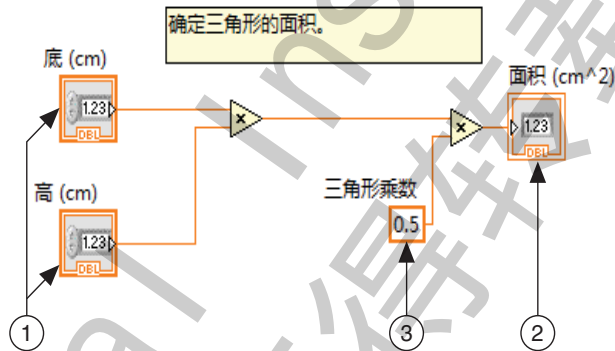
图 1-13. 三角形前面板的面积



由于用户无需更改或访问常量 0.5，因此其不会出现在前面板上。除非该常量被包含在算法的文档内。

图 1-14 为该算法在 LabVIEW 程序框图上可能的实现方式。程序框图上有 4 个由 2 个输入控件、1 个常量及 1 个显示控件创建的接线端。

图 1-14. 三角形面积程序框图，接线端显示为图标



1	输入控件	2	显示控件	3	常量
---	------	---	------	---	----

注意：**底 (cm)**和**高 (cm)**程序框图接线端与**面积 (cm²)**接线端的外观不同。程序框图上的输入控件和显示控件具有明显的区别。首个区别是指示数据流方向的接线端箭头。输入控件的箭头显示为数据离开接线端，显示控件的箭头显示为数据进入接线端。第二个区别为接线端的包围边框。输入控件的边框厚度比显示控件的边框厚度高。

接线端可采用图标显示或非图标显示。图 1-15 为采用非图标方式显示接线端的同一程序框图。但上述区别仍然存在。

图 1-15. 三角形面积程序框图，接线端显示为图标



程序框图节点

节点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、运算符、函数和子程序。节点可以是函数、子 VI 或结构。结构为过程控制元素。例如，条件结构、For 循环或 While 循环。图 1-12 中的“加”和“减”函数即为函数节点。

函数

函数是 LabVIEW 中最基本的操作元素。函数不具有前面板或程序框图，但有连线板。双击函数仅表示选中函数。函数图标为浅黄色背景。

子 VI

子 VI 为用户创建以用于其他 VI 内部或通过函数选板访问的 VI。

任何 VI 均可用作子 VI。双击程序框图中的子 VI，可查看子 VI 的前面板和程序框图。前面板包含输入控件和显示控件。程序框图包含连线、图标、函数、可能的子 VI 及其他 LabVIEW 对象。前面板和程序框图右上角均显示 VI 的图标。如 VI 用作子 VI，程序框图中显示的即为子 VI 的图标。

子 VI 也可以是 Express VI。由 Express VI 的配置是通过对话框完成的，因此是需要连线最少的节点。Express VI 用于完成常规测量任务。Express VI 的配置可被保存用作子 VI。关于通过 Express VI 配置创建子 VI 的详细信息，见 *LabVIEW 帮助* 的 *Express VI* 主题。

LabVIEW 使用彩色图标区分程序框图上的 Express VI 和其他 VI。程序框图中 Express VI 的图标为蓝色背景，子 VI 的图标为黄色背景。

可扩展节点与图标

VI 和 Express VI 可用图标或可扩展节点的形式显示。可扩展节点通常显示为被彩色背景包围的图标。子 VI 的底色为黄色，Express VI 的底色为蓝色。显示为图标可节省程序框图的空间。使用可扩展节点则便于连线并有助于用户为程序框图添加说明。默认状态下，子 VI 在程序框图上显示为图标，Express VI 显示为可扩展节点。如需将子 VI 或 Express VI 显示为可扩展节点，右键单击子 VI 或 Express VI 并从快捷菜单中单击**显示为图标**，取消勾选标记。

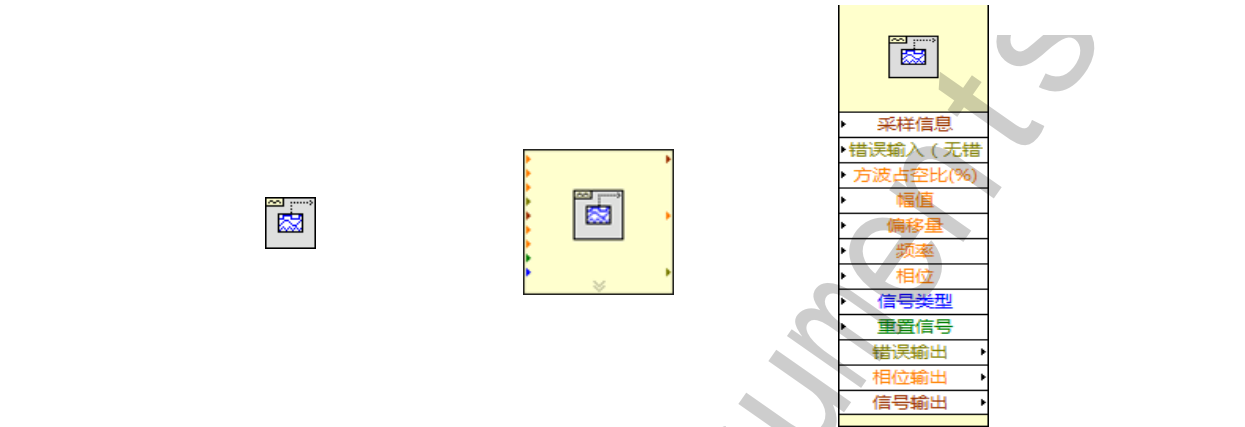
用户可调整可扩展节点的大小，以简化连线。但这将占用更多的程序框图空间。按照下列步骤，调整程序框图上节点的大小：

1. 将定位工具移到节点上。节点顶部和底部将出现调节柄。
2. 移动光标到调节柄处，使光标变成调整大小的光标。
3. 用调整大小光标向下拖动节点边界，直至出现新的接线端。
4. 松开鼠标键。

拖动节点边界超过程序框图窗口后松开鼠标键，则取消大小调整操作。

图 1-16 为“基本函数发生器”VI 进行大小调整操作后的扩展节点示意图。

图 1-16. 不同显示模式下的“基本函数发生器”VI



 **注** 以可扩展节点的形式显示子 VI 或 Express VI，则不能显示该节点的接线端，也不能启用该节点的数据库访问。

连线



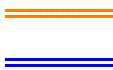






连线用于在程序框图各对象间传递数据。在图 1-12 中，输入控件和显示控件接线端通过连线连接至“加”和“减”函数。每根连线都只有一个数据源，但可以与多个读取该数据的 VI 和函数连接。不同数据类型的连线有不同的颜色、粗细和样式。

断开的连线显示为黑色的虚线，中间有个红色的 X。如下图所示。出现断线的原因有很多。试图连接数据类型不兼容的两个对象时就会产生断线。



表 1-1 为最常见的连线类型。

表 1-1. 常见连线类型

连线类型	标量	一维数组	二维数组	颜色
数值型				橙色（浮点型）， 蓝色（整型）
布尔				绿色
字符串				粉色

在 LabVIEW 中通过连线将多个接线端连接起来，使数据在 VI 间传递。输入和输出与连线传输的数据类型必须兼容，才能连线。例如，数组输出与数值输入之间不能连线。此外，连线的方向必须正确。连线仅能连接一个输入端及至少一个输出端。例如，两个显示控件间不能连线。判定连线是否兼容的因素包括输入控件的数据类型和 / 或显示控件及接线端的数据类型。

数据类型

数据类型表明可相互连接的对象、输入和输出类型。例如，开关控件的边框为绿色，可与任意带绿色标签的 Express VI 输入端相连。旋钮控件为橙色边框，可与任意带橙色标签的 Express VI 输入端相连。橙色旋钮开关无法与带绿色标签的输入端相连。连线与接线端的颜色相同。

自动连接对象

将选中的对象移到程序框图上其他对象旁时，LabVIEW 将以暂时连线来显示有效的连线方式。将对象放置在程序框图上时，放开鼠标后 LabVIEW 将自动连线。程序框图上已有对象也可自动连线。LabVIEW 将连接最匹配的接线端，对不匹配的接线端不予连线。

使用定位工具来移动对象时，按空格键则切换到自动连线模式。

默认状态下，从**函数**选板选择一个对象时，或通过按住 <Ctrl> 键的同时拖动对象来复制一个程序框图上已有的对象时，自动连线方式将启用。默认状态下，使用定位工具来移动程序框图上已有的对象时，自动连线将禁用。

选择**工具** » **选项**，从**类别**列表中选择**程序框图**可取消自动连线。

手动连接对象

连线工具移到接线端时，将出现一个提示框显示接线端的名称。此外，接线端在**即时帮助**窗口和图标上均闪烁，以提醒用户已连接至正确的接线端。连接对象时，将连线工具移至第一个接线端并单击，移动鼠标至第二个接线端再次单击。连线后，可单击连线并从快捷菜单中选择**整理连线**，通过 LabVIEW 自动选择连线路径。如有断开的连线需要移除，可按下 <Ctrl-B> 删除程序框图中的全部断开连线。

函数选板

函数选板中包含创建程序框图所需的 VI、函数及常量。在程序框图上单击**查看** » **函数选板**，访问**函数**选板。**函数**选板分为不同的类，用户可根据需要显示和隐藏类。图 1-17 为显示了全部类的**函数**选板，展开的控件类为**编程**。在本教程中，用户主要使用**编程**类，但也会使用其他类或子选板。

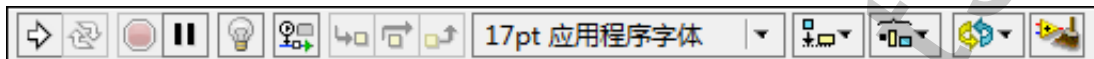
图 1-17. 函数选板



如要查看或隐藏类，可在选板上选择**自定义**按钮，选中或取消选中**更改可见选板**选项。

程序框图工具栏

运行 VI 时，用户可使用程序框图工具栏上出现的按钮调试 VI。下列为出现在程序框图上的工具栏。



单击**高亮显示执行过程**按钮，动态显示程序框图的执行过程。查看程序框图中数据的流动状态。再次单击按钮禁用高亮显示执行过程。



单击**保存连线值**按钮，将探针置于连线中时将保存执行过程中各个点的连线值，用户可以马上获取通过该连线的最新数据值。用户至少需要成功运行 VI 一次，才能获取连线值。



单击**单步步入**按钮打开一个节点并暂停。再次单击**单步步入**按钮时，将执行第一个操作，然后在子 VI 或结构的下一个操作前暂停。也可按下 <Ctrl> 和向下箭头键。VI 的单步执行是从一个节点移动至另一个节点的过程。当节点已准备就绪，可以执行时节点将闪烁。



单击**单步步过**执行节点并在下一个节点前暂停。也可按下 <Ctrl> 和向右箭头键。单步步过节点执行时，无需单步执行节点。



单击**单步步出**按钮结束当前节点的执行并暂停。VI 执行结束后，**单步步出**按钮将变为灰色。也可按下 <Ctrl> 和向上箭头键。通过单步步出节点，可单步执行节点并移动至另一节点。



单击**整理程序框图**按钮自动布局现有连线并重新排列程序框图上的对象，以获得更清晰的布局。如要配置整理选项，单击**工具 » 选项**显示选项对话框并从**类别**列表选择**程序框图**。在“程序框图整理”部分配置设置。



如在**错误列表**中勾选了**显示警告**复选框，VI 包含警告时将弹出**警告**按钮。警告表示程序框图包含潜在的问题，但不会影响 VI 的运行。



LabVIEW 帮助工具

使用**即时帮助**窗口、*LabVIEW 帮助*和 NI 范例查找器创建和编辑 VI。关于 LabVIEW 的详细信息，见 *LabVIEW 帮助*和手册。

即时帮助窗口

移动光标至 LabVIEW 对象时，**即时帮助**窗口可显示该对象的基本信息。选择**帮助**»**显示即时帮助**、按下 <Ctrl-H> 键或单击工具栏上的**显示即时帮助窗口**按钮可切换至**即时帮助**窗口的显示。

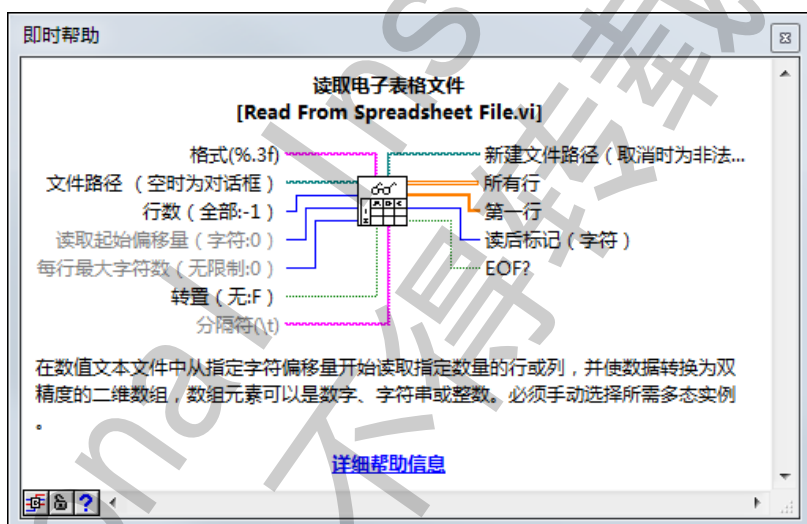


移动鼠标至前面板和程序框图对象时，**即时帮助**窗口显示子 VI、函数、常量、输入控件和显示控件的图标，以及每个接线端连线。移动鼠标至对话框选项时，**即时帮助**窗口将显示选项的说明。

在**即时帮助**窗口，必需接线端的标签显示为粗体，推荐接线端显示为纯文本，可选接线端显示为灰色。如在**即时帮助**窗口单击**隐藏可选接线端和完整路径**，则不会出现可选接线端的标签。



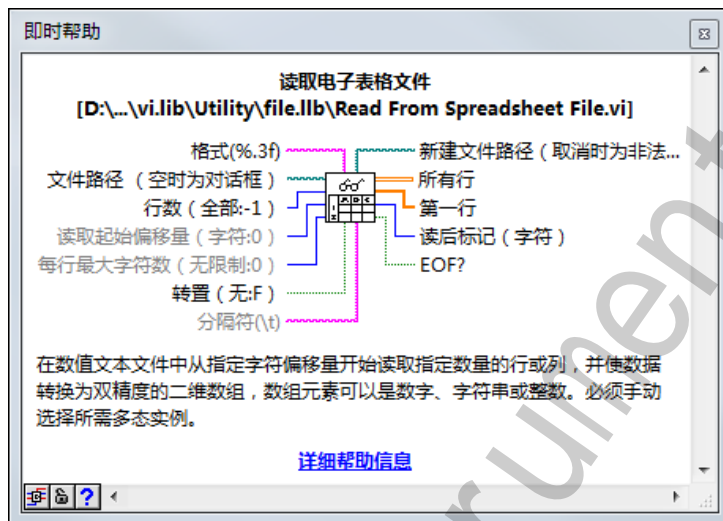
图 1-18. 即时帮助窗口



单击**即时帮助**窗口左下角的**显示可选接线端和完整路径**按钮，显示连线板的可选接线端及 VI 的完整路径。可选接线端显示为接线头，提示用户存在其他的连接。详细模式将显示全部接线端，如图 1-19 所示。



图 1-19. 详细即时帮助窗口



单击**锁定**按钮锁定**即时帮助**窗口。即时帮助内容被锁定后，移动鼠标至其他对象不会改变即时帮助窗口的内容。再次单击该按钮可解除锁定。通过**帮助**菜单也可访问该选项。



如**即时帮助**窗口中的对象在 *LabVIEW 帮助*中也有相应的介绍，**即时帮助**窗口中会出现一个蓝色的**详细帮助信息**链接。同时，**详细帮助信息**按钮已启用。单击链接或按钮显示 *LabVIEW 帮助*获取该对象的详细信息。



LabVIEW 帮助

在**即时帮助**窗口单击**详细帮助信息**按钮、单击**帮助 » LabVIEW 帮助**或单击**即时帮助**窗口的蓝色**详细帮助信息**链接访问 *LabVIEW 帮助*。用户也可以右键单击对象，从快捷菜单中选择**帮助**。

*LabVIEW 帮助*包含多数选板、菜单、工具、VI 及函数的详细介绍。*LabVIEW 帮助*也提供了使用 LabVIEW 功能的分步指导。*LabVIEW 帮助*包含下列资源的链接：

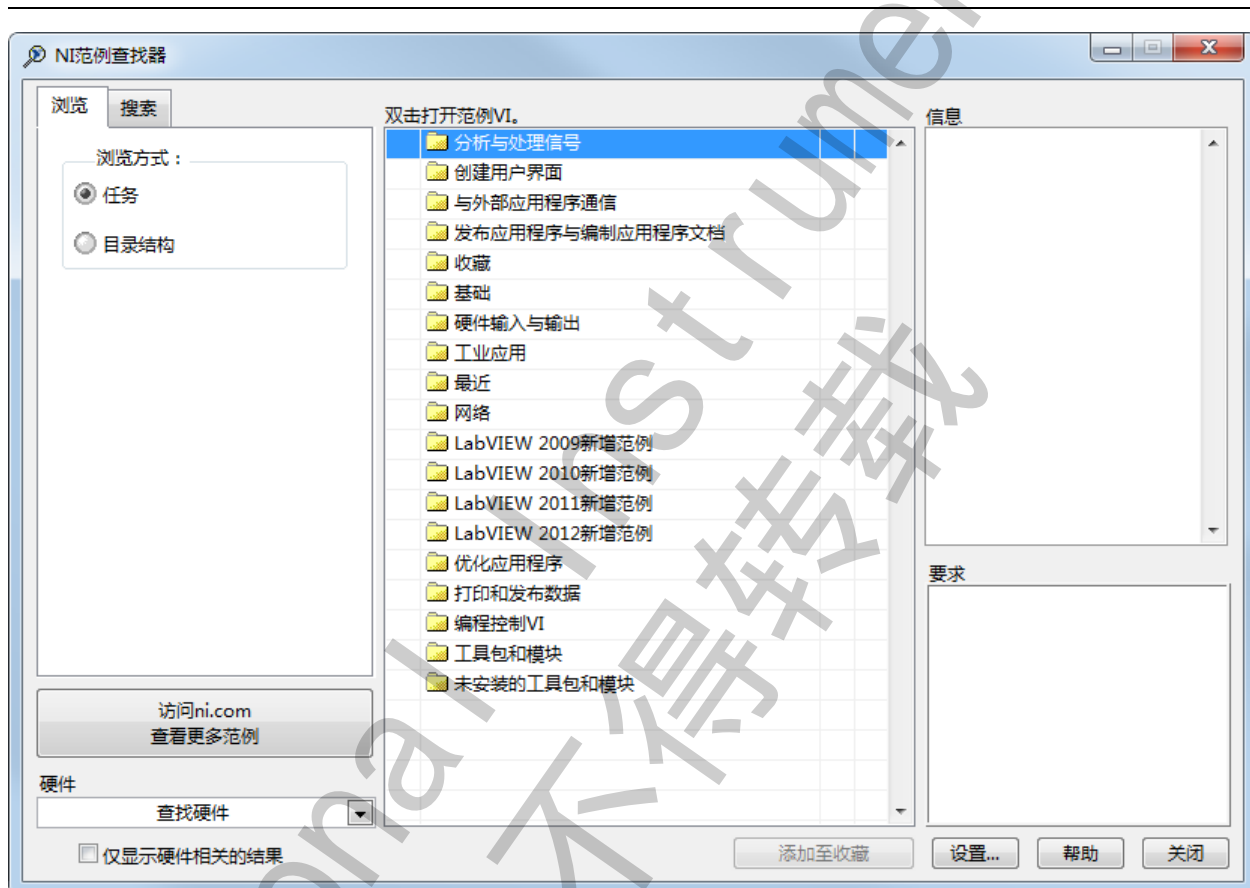
- *LabVIEW 文档资源*有助于新用户及有经验的用户的在线及打印文档，并包含了全部 LabVIEW 手册的 PDF 版本。
- NI 网站的技术支持资源。例如，NI 开发者原地、知识库和产品手册文库。

NI 范例查找器

通过 NI 范例查找器可浏览或搜索计算机上已安装的范例，或 NI 开发者园地 (ni.com/zone) 中的范例。范例可演示使用 LabVIEW 执行多种测试、测量、控制和设计任务的方法。如需启动 NI 范例查找器，可选择**帮助»查找范例**，或单击**启动**窗口的**范例**中的**查找范例**链接。

范例可演示使用特定 VI 或函数的方法。右键单击程序框图或已锁定选板上的 VI 或函数，在快捷菜单中选择**范例**，打开的帮助主题可显示该 VI 或函数的范例的链接。用户可依据应用程序修改范例，或在创建的 VI 中添加范例。

图 1-20. NI 范例查找器



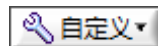
F. 搜索控件、VI 和函数

单击**查看»控件选板**或**查看»函数选板**打开**控件**和**函数**选板时，两个按钮将出现在选板的顶部。

搜索—用于将选板转换至搜索模式，通过文本搜索来查找选板上的控件、VI 或函数。选板处于搜索模式时，可单击返回按钮退出搜索模式，返回至选板。

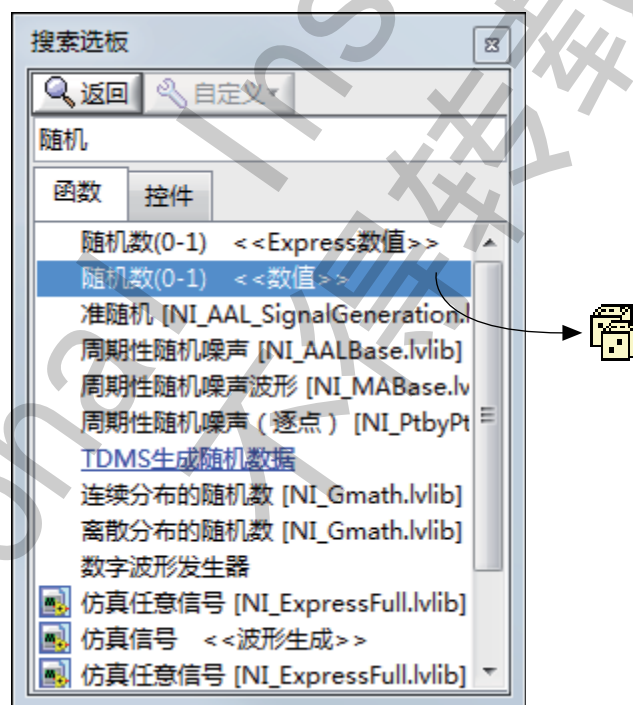


自定义—用于选择当前选板的视图模式，显示或隐藏所有选板目录，在**文本**和**树形**模式下按字母顺序对各项排序。在快捷菜单中选择**选项**，打开**选项**对话框中的**控件/函数选板**页，为所有选板选择显示模式。只有当单击选板左上方的图钉标识将选板锁定时，才会显示该按钮。



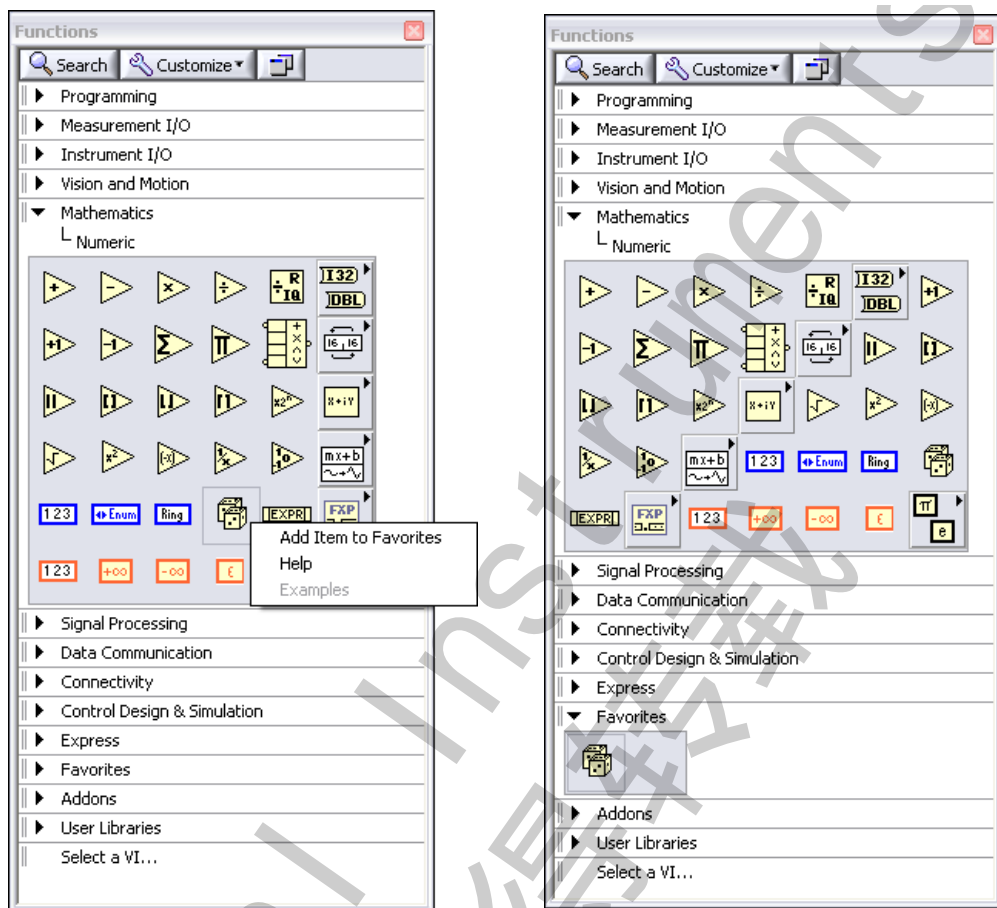
用户在熟悉 VI 和函数的位置之前，可使用**搜索**按钮搜索函数或 VI 的位置。例如，如要查找“随机数”函数，单击**函数**选板上的**搜索**按钮并在选板顶部的文本框内键入随机。LabVIEW 将列出所有匹配的项，包含以“随机”开头或其中包含“随机”文本的项。用户可单击其中一个搜索结果并将其拖拽至程序框图，如图 1-21 所示。

图 1-21. 在函数选板内搜索对象



双击搜索结果可在选板上高亮显示其位置。如对象为常用对象，可将其添加至“收藏夹”类。在选板上右键单击对象并选择**添加项至收藏夹**，如图 1-22 所示。

图 1-22. 添加项至选板的收藏夹类



与搜索按钮类型，使用**快速放置**对话框通过名称指定选板对象，并将对象放置在程序框图或前面板上。除选板对象外，在**快速放置**对话框内还可通过名称指定项目项。

按下 <Ctrl-Space> 或选择**查看 » 快速放置**显示**快速放置**对话框。输入要在程序框图或前面板上放置的对象的名称。LabVIEW 将结果显示在**名称匹配列表**窗口中。按下 <Enter> 键，双击列表中对象的名称或单击前面板或程序框图，可使对象随光标移动。单击程序框图或前面板上的某一位置，可将对象放置在该位置。

图 1-23. 在“快速放置”对话框内搜索对象



G. 选择工具

使用 LabVIEW 提供的工具创建、修改和调试 VI 工具选板上的每一个工具都对应于鼠标的的一个操作模式。鼠标动作取决于所选择的工具图标 LabVIEW 根据当前鼠标的位置选择工具。

图 1-24. 工具选板



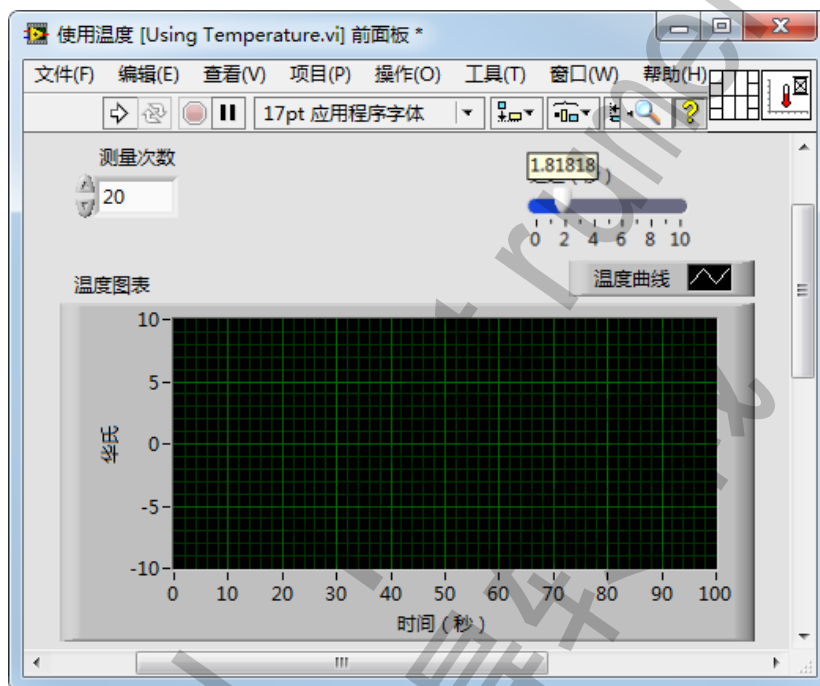
提示 在**工具选板**内手动选择所需的工具。单击**查看 » 工具选板**显示**工具选板**。

操作值工具

当鼠标指针变为下列图标时，表示操作值工具在使用中。操作值工具用于更改控件的值。例如，在 1-25 中，通过操作值工具更改水平指针滑动杆的指针位置。鼠标移动至滑动杆指针上时，鼠标将自动切换为操作值工具。



图 1-25. 使用操作值工具



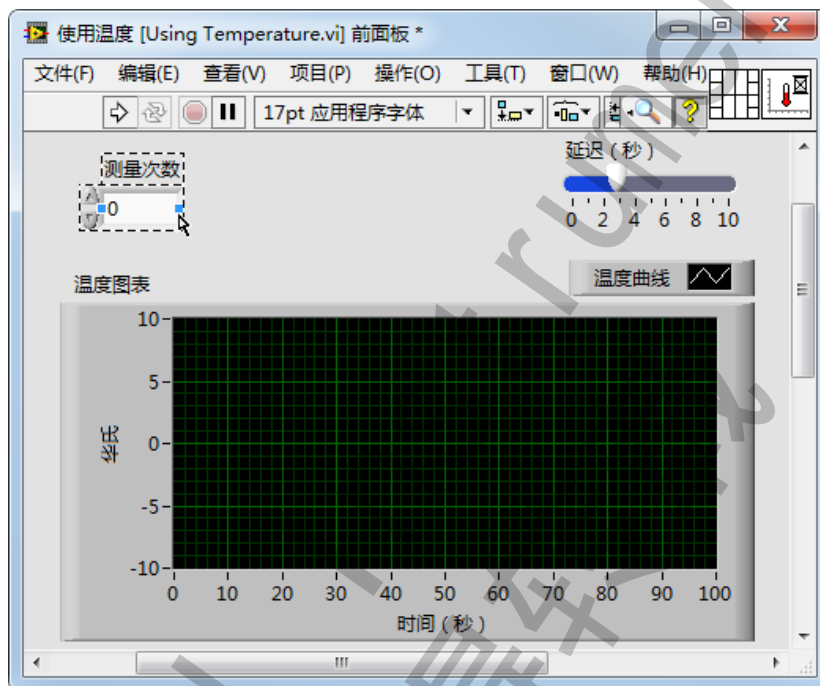
操作值工具通常用在前面板窗口，但也可在程序框图窗口使用操作值工具更改布尔常量的值。

定位工具

当鼠标指针变为下列图标时，表示定位工具在使用中。定位工具用于对象的选择、定位或调整大小。例如，在图 1-26 中，通过定位工具选择**测量次数**数值控件。选择对象后，可移动、复制或删除对象。鼠标移动至对象边沿时，鼠标将自动切换为定位工具。

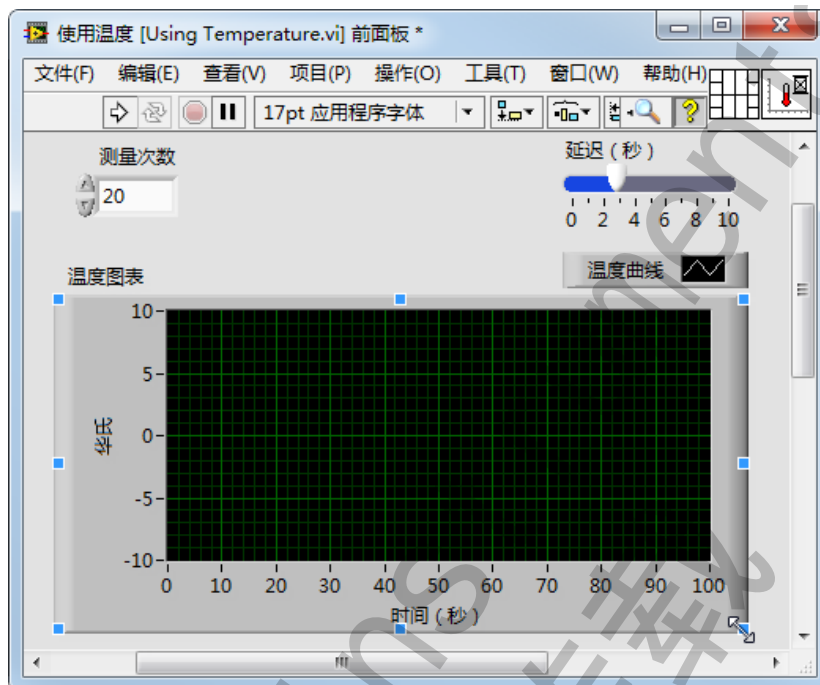


图 1-26. 使用定位工具选择对象



鼠标移动至对象的可调整大小节点时，鼠标模式切换为调整对象的大小。如图 1-27 所示。注意，调整节点大小时，鼠标悬浮在 XY 图的某一角，且鼠标模式切换为双向箭头。

图 1-27. 使用定位工具调整对象的大小



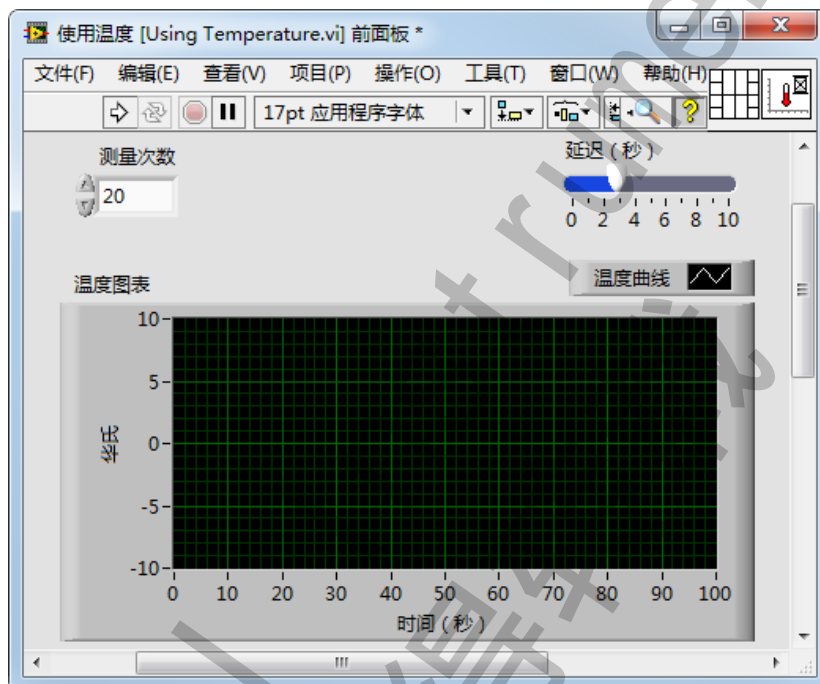
定位工具可用在前面板窗口和程序框图窗口。

标签工具

当鼠标指针变为下列图标时，表示标签工具在使用中。标签工具用户在控件中输入文本、编辑文本及创建自由标签。例如，在图 1-28 中，通过标签工具在**测量次数**数值控件中输入文本信息。鼠标移动至控件内部时，鼠标将自动切换为标签工具。单击一次鼠标，将其放置在控件内部。双击鼠标选择当前文本信息。



图 1-28. 使用标签工具



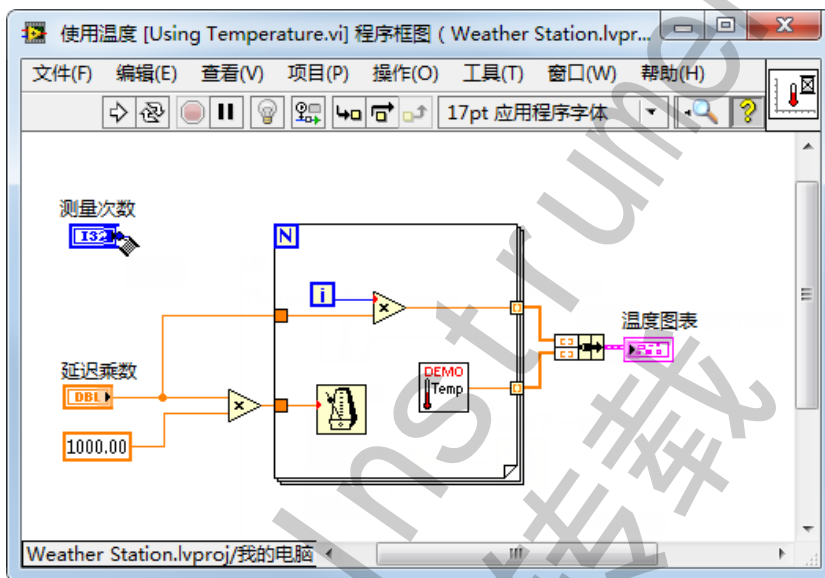
在前面板或程序框图窗口的未指定区域，即未进入到特定鼠标模式的区域内，鼠标显示为十字准线。如启用了自动选择工具，可双击任意部分以切换至标签和创建自由标签。

连线工具

当鼠标指针变为下列图标时，表示连线工具在使用中。连线工具用于连接程序框图上的对象。例如，在图 1-29 中，通过连线工具连接**测量次数**接线端至 For 循环的计数接线端。鼠标移动至接线端的输入或输出端及连线时，鼠标自动切换为连线工具。



图 1-29. 使用连线工具



连线工具主要用于程序框图窗口，及在前面板窗口创建连线板。

选板上的其他可用工具

通过**工具**选板可直接访问操作值、定位、标签和连线工具，无需使用自动选择工具模式。单击**查看»**
工具选板显示**工具**选板。

图 1-30. 工具选板



工具选板顶部的项为**自动选择工具**按钮。选择该项时，LabVIEW 将根据鼠标的位置自动选择工具。取消选择该项或选择选板中的其他项可关闭自动选择工具。下文将介绍选板上的其他工具：



对象快捷菜单用于单击鼠标左键访问对象快捷菜单。



滚动窗口工具用于不通过滚动栏实现窗口的滚动。



断点工具用于在 VI、函数、节点、连线及结构上设置断点，使执行在断点处暂停。



探针工具用于在程序框图连线上创建探针。使用探针工具可查看产生问题或意外结果的 VI 中的即时值。



获取颜色工具用于获取颜色，并复制该颜色。



设置颜色工具用于为某个对象上色。设置颜色工具也用于显示当前前面板和后面板的颜色设置。



选择设置颜色工具，右键单击对象或工作区域显示颜色提取器。

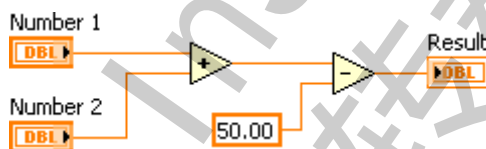
H. 数据流

LabVIEW 按照数据流模式运行 VI。当具备了所有必需的输入时，程序框图节点将运行。节点在运行时产生输出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 以及绝大多数其他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。

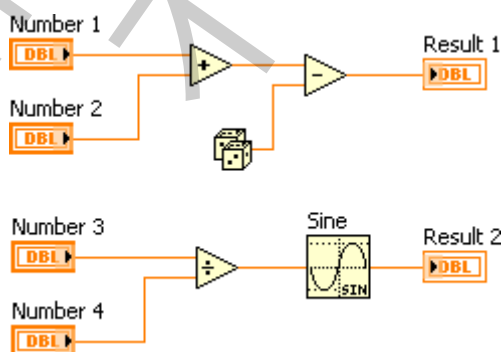
下文为一个数据流可编程范例，程序框图中包含 2 个数值进行加法运行，然后从和值中减去 50.00。如图 1-31 所示。在该范例中，程序框图从左至右执行。并非因为图中对象的摆放次序，而是因为减函数需要等待加函数执行结束并传递其结果数据至减函数，才能开始执行。请记住仅当节点的全部输入端上的数据都准备就绪，节点才能开始执行。仅当节点执行结束后，才能将数据传递至输出接线端。

图 1-31. 数据流编程范例



在图 1-32 中，请思考哪一部分代码将优先执行—加、随机数还是除函数。上述问题没有确定性答案，因为输入“加”和“除”函数的数据是同时的，且“随机数”函数不具有输入。在一部分代码必须优先于另一部分代码执行，且函数之间不存在数据依赖性的条件下，请考虑使用其他编程方式（例如，错误簇）强制执行次序。关于错误簇详细信息，见第 5 章“创建和使用数据结构”。

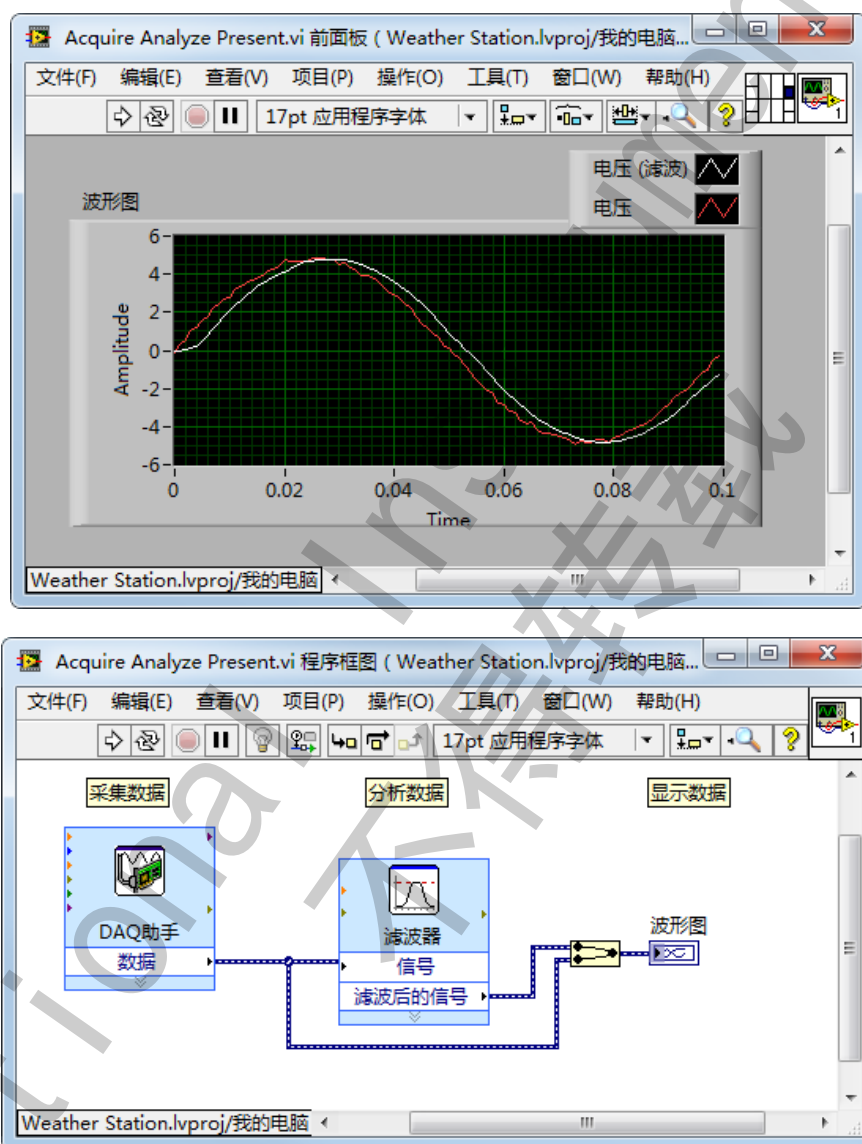
图 1-32. 多个代码片段的数据流范例



I. 创建一个简单 VI

多数 LabVIEW VI 具有三个主要任务—采集某类型的数据、分析采集到的数据并显示结果。如上述每个部分均比较简单，在程序框图上使用较少的对象即可完成整个 VI。Express VI 设计用于完成常见的且频繁使用的操作。在本章节中，您将了解一些采集、分析和显示数据的 Express VI。及使用上述 3 个任务创建一个简单 VI。如图 1-33 所示。

图 1-33. 采集、分析和显示范例的前面板窗口和程序框图



在函数选板，Express VI 被分类在 **Express** 类中。Express VI 使用动态数据类型在 Express VI 间传输数据。

采集

用于“采集数据”任务的 Express VI 包含：DAQ 助手、仪器 I/O 助手、仿真信号和读取测量文件。

DAQ 助手

DAQ 助手通过数据采集设备采集数据。在本教程中会频繁的使用“DAQ 助手”Express VI。在了解更多数据采集的详细信息前，仅使用数据采集设备的一个通道 CH0。该通道连接至 BNC-2120 的温度传感器。触摸温度传感器可改变传感器读取到的温度值。



仪器 I/O 助手

“仪器 I/O 助手”一般通过 GPIB 或串行接口采集仪器控制数据。



仿真信号

“仿真信号”Express VI 可生成仿真数据。例如，正弦波。



读取测量文件

“读取测量文件”Express VI 读取通过“写入测量文件”Express VI 创建的文件。它专门读取 LVM 或 TDM 文件格式。该 Express VI 不能读取 ASCII 文件。关于读取文件数据的详细信息，见第 6 课，“管理文件和硬件资源”。



分析

用于“分析数据”任务的 Express VI 包括一幅值和电平测量、统计和单频测量。

幅值和电平测量

“幅值和电平测量”Express VI 用于测量信号电压。电压信号包括 DC、RMS、最大峰值、最小峰值、峰峰值、周期均值和周期 RMS 测量。



统计

“统计”Express VI 用于计算波形的统计数据。统计数据包括均值、和值、标准差和极值。



频谱测量

“频谱测量” Express VI 用于波形的频谱测量。例如，幅值和功率谱密度。



单频测量

“单频测量” Express VI 搜索具有最高频率或最大幅值的单频。并检测其单频的频率和幅值。



滤波器

“滤波器” Express VI 通过滤波器和各种窗处理信号。滤波器具有以下类型：高通、低通、带通、带阻和平滑。窗包括：Butterworth、Chebyshev、反 Chebyshev、Elliptical 和 Bessel。



显示

通过实现函数功能的 Express VI（例如，“写入测量文件” Express VI）或在前面板窗口显示数据的显示控件显示结果。用于本任务的最常见的显示控件包括：波形图表、波形图和 XY 图。常见的 Express VI 包括“写入测量文件” Express VI、“创建文本” Express VI、“DAQ 助手” Express VI 和“仪器 I/O 助手” Express VI。在本范例中，“DAQ 助手”和“仪器 I/O 助手”将来自计算机的输出数据提供至 DAQ 设备或外部仪器。

写入测量文件

“写入测量文件” Express VI 采用 LVM 或 TDM 格式写入文件。关于写入测量文件的详细信息，见第 6 课，“管理文件和硬件资源”。



创建文本

“创建文本” Express VI 用于创建文本，通常用于前面板显示或导出至文件 / 仪器。关于创建字符串详细信息，见第 6 课，“管理文件和硬件资源”。



运行 VI

配置 Express VI 并完成连线后，可运行 VI。创建 VI 结束后，单击工具栏上的**运行**按钮执行 VI。



VI 运行过程中，**运行**图标切换为下图的样式。执行结束后，**运行**按钮将恢复为最初的样式，此时前面板的显示控件中包含数据。



运行按钮错误

如某个 VI 无法运行，该 VI 是断开的 VI，也就是说该 VI 无法执行。如正在创建或编辑 VI 出现错误时，**运行**按钮会显示为断开。



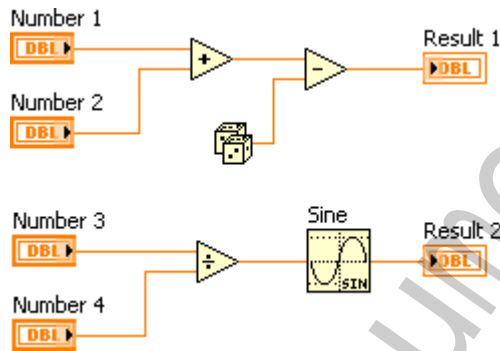
如已完成程序框图的连线而该按钮仍显示为断开，则表示 VI 是断开的且不能运行。

通常这种情况表明某个必需的输入未连接，或连线断开。单击断开的“运行”按钮，打开**错误列表**窗口。**错误列表**窗口将列出全部错误及问题描述。双击某错误可直接访问该错误。关于调试 VI 的详细信息，见第 2 课，“疑难解答和调试 VI”。

自测：练习

请参照图 1-34 回答下列测验问题。

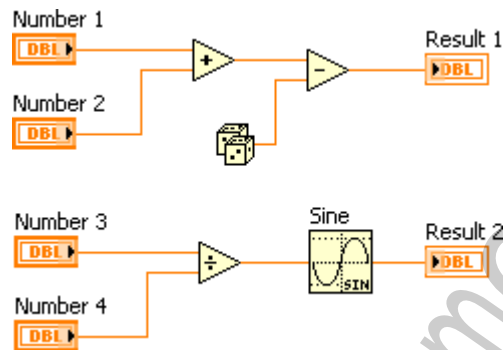
图 1-34. 数据流问题



1. 哪个函数最先执行：加还是减？
 - a. 加
 - b. 减
 - c. 未知
2. 哪个函数先执行：正弦还是除？
 - a. 正弦
 - b. 除
 - c. 未知
3. 下列哪个函数最先执行：随机数、除还是加？
 - a. 随机数
 - b. 除
 - c. 加
 - d. 未知
4. 下列哪个函数最后执行：随机数、减还是加？
 - a. 随机数
 - b. 减
 - c. 加
 - d. 未知
5. VI 的三个组成部分是什么？
 - a. 前面板窗口
 - b. 程序框图窗口
 - c. 项目
 - d. 图标 / 连线板

International Instruments
不得转载

自测：练习答案



1. 哪个函数最先执行：加还是减？
 - a. **加**
 - b. 减
 - c. 未知
2. 哪个函数先执行：正弦还是除？
 - a. 正弦
 - b. **除**
 - c. 未知
3. 哪个节点最先执行？
 - a. 随机数
 - b. 除
 - c. 加
 - d. **未知**
4. 下列哪个函数最后执行：随机数、减还是加？
 - a. 随机数
 - b. **减**
 - c. 加
 - d. 未知
5. VI 的三个组成部分是什么？
 - a. **前面板窗口**
 - b. **程序框图窗口**
 - c. 项目
 - d. **图标 / 连线板**

笔记

National Instruments
不得转载

疑难解答和调试 VI

要使 VI 运行，必须将 VI 的所有子 VI、函数和结构的接线端连接正确的数据类型。有时，VI 生成的数据和运行的方式与预期不同。LabVIEW 可配置 VI 的运行方式，并找出在程序框图的组织或流经程序框图的数据中存在的问题。

主题

- A. 纠正断开的 VI
- B. 调试技巧
- C. 未定义或未预期的数据
- D. 错误处理

A. 纠正断开的 VI

如某个 VI 无法运行，该 VI 是断开的 VI，也就是说该 VI 无法执行。如正在创建或编辑的 VI 出现错误，**运行**按钮将显示为断开。



如已完成程序框图的连线，但该按钮仍显示为断开，则表示 VI 是断开的且不能运行。

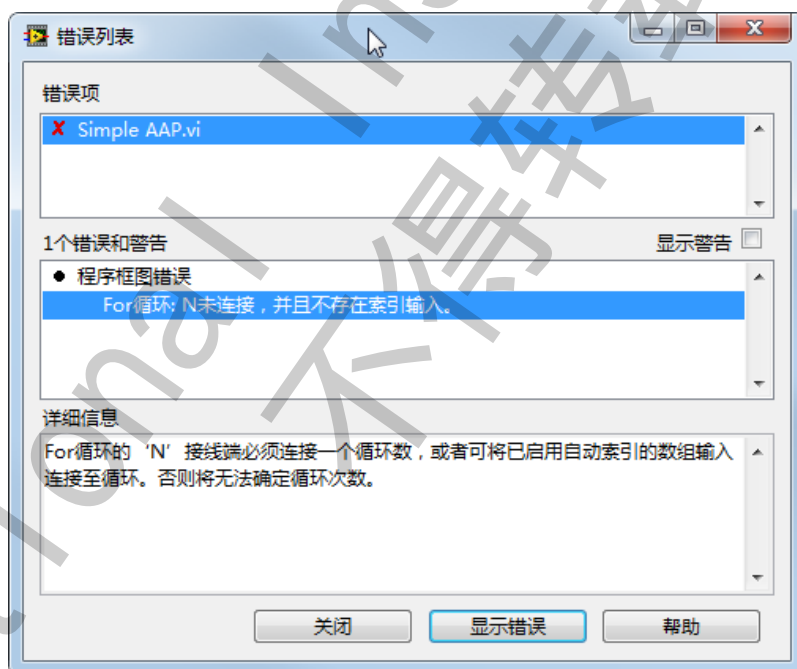
查找 VI 断开的原因

警告并不妨碍 VI 运行。警告仅帮助用户避免 VI 中的潜在问题。而错误会使 VI 断开。运行 VI 之前必须先排除全部错误。

单击断开的**运行**按钮，或选择**查看 » 错误列表**，可查看 VI 断开的原因。**错误列表**列出了所有的错误。**错误项**列出了内存中所有含有错误的项的名称，如 VI 和项目库。如两个或多个项具有相同的名称，则错误项部分会显示每一项的特定应用程序实例。**错误和警告**列出了在**错误项**中选中的 VI 错误和警告信息。**详细信息**描述了错误信息，有时还会建议如何纠正错误。单击**帮助**按钮，可显示 *LabVIEW 帮助* 中对错误的详细描述和纠正错误步骤的相关主题。

单击**显示错误**按钮或双击错误描述，可高亮显示程序框图或前面板中包含错误的区域。

图 2-1. 错误列表对话框的范例



VI 断开的常见原因

下文列出了编辑 VI 时导致 VI 断开的常见原因：

- 数据类型不匹配或存在丢失及未连接的接线端，会导致程序框图含有断线。关于纠正断开的连线的详细信息，见 *LabVIEW 帮助* 的 *纠正断开的 VI* 主题。
- 必需的程序框图接线端未连接。关于设置必需的输入和输出端的详细信息，见 *LabVIEW 帮助* 的 *使用连线连接程序框图各对象* 主题。

- 子 VI 断开或将子 VI 图标放置在 VI 程序框图上之后，对连线板进行了编辑。关于子 VI 的详细信息，见 *LabVIEW 帮助* 的 *创建子 VI* 主题。

B. 调试技巧

如在 VI 未断开状态下得到了非预期数据，可使用下列调试技术发现和纠正 VI 或程序框图数据流的问题：

- 大多数内置 VI 和函数的底部都有错误输入和错误输出参数。这些参数能检测到程序框图上节点产生的错误，并显示是否有错误产生和生成错误的位置。上述参数也可用于用户创建的 VI 中。
- 单击**查看»错误列表**，勾选**显示警告**复选框可查看全部 VI 的警告。取消勾选该复选框可忽略全部 VI 的警告。找到错误原因，在 VI 中进行相应改动。
- 使用定位工具，三击连线以高亮显示其全部路径，并确保连线连接至正确的接线端。
- 使用**即时帮助**窗口检查程序框图上各个函数和子 VI 的默认值。如推荐和可选输入端未连线，VI 和函数将传递默认值。例如，未连线的布尔输入端为 **TRUE**。
- 使用**查找**对话框，在 VI 中查找要修改的子 VI、文本和其他对象。
- 单击**查看»VI 层次结构**查找未连线的子 VI。与未连线的函数不同，除非将输入设置为“必需”，否则未连线 VI 不一定会产生错误。如将未连线的子 VI 误置于程序框图上，程序框图执行时，子 VI 也同时执行。所以，VI 可能执行了多余的操作。
- 使用高亮显示执行过程，查看数据在程序框图中的移动。
- 单步执行 VI 时，可查看运行时程序框图上的每个执行步骤
- 使用探针工具，查看实时数据值、检查 VI 和函数，尤其是进行 I/O 操作的 VI 和函数的错误输出。
- 单击程序框图工具栏上的**保存连线值**按钮，保存连线值以便在探针中使用。该项功能允许用户方便地获得最近通过连线传递的数据。
- 使用断点暂停执行，从而进行单步执行或插入探针。
- 通过中断子 VI 的执行，可编辑输入控件和显示控件的值、控制子 VI 运行次数及返回子 VI 的开始执行点
- 确定某个函数或子 VI 传递的数据是否为未定义。这通常发生在数值型数据中。例如，在 VI 中的某个点上，可能会出现一个数除以零的运算，返回**无穷**，但是其后的函数或子 VI 需要数值输入。
- 如 VI 运行速度低于预期，请确认已关闭子 VI 的高亮显示执行过程功能。此外，不使用子 VI 时应关闭其前面板和程序框图。因为打开窗口会影响执行速度。
- 检查控件的表示法，查看是否出现数据溢出。因为将浮点数转换为整数，或将整数转换为更小的整数时，可能会发生溢出。例如，将 16 位整数连接至只接收 8 位整数的函数。函数将 16 位整数转换为 8 位整数时，会造成数据丢失。
- 确定是否有 For 循环无意中执行了零次循环，产生了空数组。
- 请验证移位寄存器已初始化。除非移位寄存器仅用于保存上一次循环执行的数据，并将数据传递至下一个循环。
- 在源和目标点检查簇元素顺序。LabVIEW 在编辑时能检查到数据类型和簇大小不匹配，但是不能检查到同种类型元素的不匹配。
- 检查节点执行顺序。
- 检查以确认 VI 不包含隐藏子 VI。将一个节点置于另一个节点之上或缩小结构使子 VI 不在视线范围之内，都在无意中隐藏了子 VI。
- 检查 VI 使用的子 VI 列表，与通过**查看»浏览关系»本 VI 的子 VI**和**查看»浏览关系»未打开的子 VI**结果相对比，判定是否存在额外的子 VI。打开 VI 层级结构窗口查看 VI 的子 VI。为避免隐藏 VI 产生的错误结果，可将 VI 的输入指定为“必需”。

高亮显示执行过程

单击**高亮显示执行过程**按钮查看程序框图的动态执行过程。



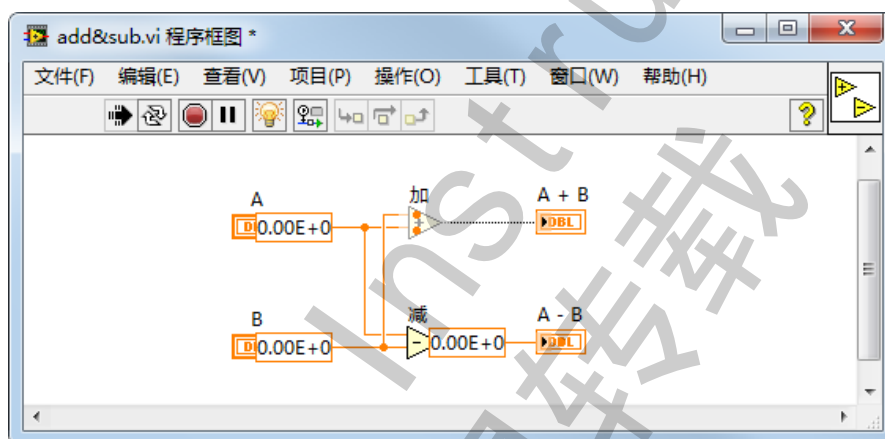
高亮显示执行过程通过沿连线移动的圆点显示数据在程序框图上从一个节点移动到另一个节点的过程。使用高亮显示执行的同时，结合单步执行，可查看 VI 中的数据从一个节点移动到另一个节点的全过程。

（**MathScript RT 模块**）在 MathScript 节点中，脚本行的蓝色箭头闪烁时，表示当前执行到该行。



注 高亮显示执行过程会导致 VI 的运行速度大幅降低。

图 2-2. 使用高亮显示执行过程的范例



单步执行

单步执行 VI 可查看 VI 运行时程序框图上 VI 的每个执行步骤。单步执行按钮仅在单步执行模式下影响 VI 或子 VI 的运行。如下图所示。



单击程序框图工具栏上的**单步步入**或**单步步过**按钮可进入单步执行模式。将鼠标移动到**单步步入**、**单步步过**或**单步步出**按钮时，可看到一个提示框，该提示框描述了单击该按钮后的下一步执行情况。单步执行一个 VI 时，该 VI 的各个子 VI 既可单步执行，也可正常运行。

在单步执行 VI 时，如某些节点发生闪烁，表示这些节点已准备就绪，可以执行。如单步执行 VI 并高亮显示执行过程，执行符号将出现在当前执行的子 VI 的图标上。



探针工具

使用探针工具，在 VI 运行时检查连线上实时传递的值。



如程序框图较复杂且包含一系列每步执行都可能返回错误值的操作，可使用探针工具。利用探针并结合高亮显示执行过程、单步执行和断点，可确认数据是否有误并找出错误数据。如有流经数据，高亮显示执行过程、单步调试或在断点位置暂停时，**探针监视窗口**会立即更新和显示数据。当执行过程由于单步执行或断点而在某一节点处暂停时，可用探针探测刚才执行的连线，查看流经该连线的数值。



提示 如要探针显示 VI 最后一次执行时显示流经连线的数值，单击程序框图工具栏上的**保存连线值**按钮。

探针类型

VI 运行时，可使用通用探针、**控件**选板上的显示控件、内置探针、自定义内置探针或创建一个新探针查看连线上的即时值。



注（**MathScript RT 模块**）可使用 LabVIEW MathScript 探针在 VI 运行时查看 MathScript 节点中脚本的数据。

通用

通用探针可查看流经连线的数据。右键单击连线，从快捷菜单中选择**自定义探针»通用探针**使用通用探针。

通用探针可显示数据。但无法配置通用探针，使其对数据作出响应。

除非已为数据类型指定了一个自定义的或内置的探针，否则右键单击连线并选择**探针**时 LabVIEW 将显示通用探针。

自定义探针可像 VI 一样进行调试。但自定义探针不能探测自身的程序框图，也不能探测其子 VI 的程序框图。调试自定义探针时应使用通用探针。

使用显示控件查看数据

使用显示控件也可查看流经连线的数据。查看数值数据时，可在探针中使用一张图表来查看数据。右键单击连线，从快捷菜单中选择**自定义探针»控件**，并选择要使用的显示控件。或者单击**控件**选板上的**选择控件**图标，从计算机或服务器的共享目录中选择任何已保存的自定义控件或自定义类型。使用自定义类型查看被探测数据时，LabVIEW 将自定义类型视作自定义控件。

如所选显示控件的数据类型与右键单击选中的连线的数据类型不匹配，LabVIEW 将不把显示控件放置在连线上。

内置

内置探针是显示连线中传输数据的综合信息的 VI。例如，VI 引用句柄探针可返回 VI 名、VI 路径和引用的十六进制值等信息。内置探针可根据流经连线的数据作出响应。例如，错误簇中的错误探针可接收状态、代码、错误源和错误描述，并指定在出错或报警时是否需要设置一个条件断点。

内置探针位于**自定义探针**快捷菜单的最上方。右键单击连线，从快捷菜单中选择**自定义探针**可选择内置探针。只有与右键单击的连线的数据类型相匹配的探针才会出现在快捷菜单上。

关于使用内置探针的范例见 labview\examples\general\probes.llb。

自定义

创建新探针对话框用于创建一个基于已有探针的探针或创建一个新探针。右键单击连线，从快捷菜单中选择**自定义探针»新建**，打开**创建新探针**对话框。如需更好地控制 LabVIEW 对流经连线数据的探测方式，可创建一个探针。新建探针的数据类型与右键单击的连线的数据类型相匹配。如需编辑已创建的探针，须先从保存探针的目录打开探针。

从**自定义探针**快捷菜单中选择一个探针，用控件选板中的**选择控件**选项找到 VI 作为探针，或通过**创建新探针**对话框创建一个新探针。该探针将变为该数据类型的默认探针，用右键单击连线并从快捷菜单中选择**探针**时，LabVIEW 会加载该探针。LabVIEW 只加载与右键单击的连线的数据类型完全匹配的探针。即一个双精度浮点数探针不能探测一个 32 位无符号整数的连线，即使 LabVIEW 可转换该数据。



注 如需使自定义探针成为某一特定数据类型的默认探针，则应将该探针保存到 user.lib_probes\default 目录中。请勿将探针保存到 vi.lib_probes 目录，因为在升级或重新安装 LabVIEW 时，LabVIEW 将覆盖这些文件。

断点

使用端点工具在 VI、节点或连线上放置一个断点，程序运行到该处时暂停执行。



在连线上设置断点后，数据流经该连线且**暂停**按钮为红色时程序将暂停执行。在程序框图上放置一个断点，使程序框图在所有节点执行后暂停执行。此时程序框图边框变为红色，断点不断闪烁以提示断点所在位置。

VI 在某个断点处暂停时，LabVIEW 将把程序框图置于顶层显示，同时一个选取框将高亮显示含有断点的节点、连线或脚本。光标移动到断点上时，“断点”工具光标的黑色区域变为白色。

程序执行到一个断点时，VI 将暂停执行，同时**暂停**按钮显示为红色。可进行下列操作：

- 用单步执行按钮单步执行程序。
- 在连线上添加探针查看中间数据。
- 改变前面板控件的值。
- 单击**暂停**按钮可继续运行到下一个断点处或直到 VI 运行结束。

中断执行

中断子 VI 的执行多发生于需编辑输入控件和显示控件的值、控制子 VI 在返回调用程序之前运行的次数，以及返回到子 VI 执行起点的情况。既可挂起子 VI 的所有调用，也可挂起子 VI 的某个特定调用。

如需中断子 VI 的所有调用，打开子 VI 并选择**操作»调用时挂起**。当另一个 VI 调用该子 VI 时，子 VI 会自动挂起。单步运行时，子 VI 不会立即中断。只有在子 VI 被调用时才会挂起。

如需中断子 VI 的一个特定调用，右键单击程序框图中该子 VI 的节点，并从快捷菜单中选择**子 VI 节点设置**。勾选**调用时挂起**复选框，则仅在该子 VI 的实例中中断执行。

单击**查看»VI 层次结构**，在 VI 层次结构窗口将表明是否有 VI 被暂停或中断。

箭头符号表示该 VI 正在正常运行或单步执行。



暂停符号表示该 VI 被暂停或中断。



- 绿色暂停符号或黑白的中空符号表示该 VI 被调用时会暂停。
- 红色暂停符号或黑白的实心符号表示该 VI 当前暂停。

感叹点符号表示子 VI 已被中断。



注 VI 可同时被中断和暂停。

确定子 VI 的当前实例

子 VI 暂停时，工具栏上的**调用列表**下拉菜单将列出从顶层 VI 到子 VI 的一个调用链。单击**查看 » 查看关系 » 本 VI 的调用方**所显示的列表与上述列表不同，此时显示的列表包含了所有调用方 VI，不管 VI 是否当前在运行状态。如程序框图包含的实例多于一个，则通过**调用列表**菜单可确认子 VI 的当前实例。选中**调用列表**菜单中的某个 VI 时，LabVIEW 将打开该 VI 的程序框图并高亮显示子 VI 的当前实例。

使用调用链函数也可查看当前 VI 至顶层 VI 的调用链。

C. 未定义或未预期的数据

非预期数据，即 NaN（非法数字）或 Inf（无穷）会影响后续操作。浮点数据操作返回以下两种符号值用以表明错误的计算或无意义的结果：

- NaN（非法数字）表示无效操作所产生的浮点值。例如，对负数取平方根。
- Inf（无穷）表示无效操作所产生的浮点值。例如，用零除以某个数值。

LabVIEW 不检查整数的上溢或下溢条件。浮点数的上溢和下溢符合 IEEE 754 二进制浮点数算术标准。

浮点运算能可靠地传送 NaN 和 Inf。显式或隐式的转换 NaN 或 Inf 为整数或布尔值，该转换无意义。例如，1 被 0 除时产生 Inf。转换 Inf 为 16 位整数值将生成一个看似正常的 32,767 值。

在将数据转换为整型之前，可先用探针工具查看中间浮点数值是否有效。也可将“非法数字 / 路径 / 句柄？”函数连接到此数值上检查是否为 NaN。

不要依据特殊值，例如 NaN、Inf 或空数组来判定 VI 是否生成了未定义数据。在 VI 可能生成未定义数据的情况下，使 VI 报告错误可确定该 VI 生成了已定义的数据。

例如，创建一个使用了输入数组来自动索引 For 循环的 VI，则需在输入数组为空时确定该 VI 的操作。如数组为空，可生成输出错误码并用确定的数据取代循环生成的数据值，或使用一个数组为空时不执行 For 循环的条件结构。

D. 错误处理

无论 VI 有多完美，也很难预见到用户可能遇到的每一个问题。如没有一个检查错误的机制，则可确定的仅是 VI 存在错误。通过错误检查则可确认发生错误的原因和错误出现的位置。

错误处理是预估、检测以及解决警告和错误的机制。错误处理是 LabVIEW 应用程序开发的一个重要组件。通过错误处理，用户可迅速找到编程错误的源。如没有错误处理，用户可能发现未预期的动作，但不能迅速定位出现问题的位置。

当用户调试应用程序，以确保错误报告有效且在生成错误的情况下，错误处理代码能安全停止应用程序时，错误处理尤其重要。例如，在压力测试中设定超出应用程序常规操作范围的值和条件，出现这些值和条件时通常会导致错误。出现上述错误时，我们希望能够确保应用程序能够正常关闭。

应用程序部署完成后，错误处理仍非常重要。错误处理可帮助检测系统和环境差别—例如，文件系统、内存和磁盘资源的差别。NI 强烈建议使用错误处理。

自动错误处理

默认状态下，LabVIEW 将通过挂起执行、高亮显示出错的子 VI 或函数并显示错误对话框的方式，自动处理每个错误。在错误对话框中，用户可看到每个错误具有一个标识数字代码及相关的错误信息。

单击**文件»VI 属性**，在**类别**下拉菜单中选择**执行**，可禁用当前 VI 的自动错误处理。单击**工具»选项**，从**类别**列表中选择**程序框图**可禁用新建空白 VI 的自动错误处理。如需禁用一个 VI 中的子 VI 或函数的自动处理错误功能，可将其**错误输出**参数与另一个子 VI 或函数的**错误输入**参数连接，或连接到一个**错误输出**显示控件。

手动错误处理

有时候可能需要在不依赖自动错误处理的情况下处理错误。例如，程序框图上有一个 I/O VI 超时，但并不希望整个应用程序都停止运行，同时也不希望错误对话框出现。也可能需要在一段时间内重新运行该 VI。

在 LabVIEW 中，可在 VI 的程序框图上通过下列方法手动处理错误：

- 使用**对话框与用户界面**选板的 LabVIEW 错误处理 VI 和函数通知和提示用户。例如，LabVIEW 遇到了错误，可在不同类型的对话框中显示错误信息。
- 使用错误簇和多数 VI 和函数的**错误输入**和**错误输出**参数管理错误。例如，检测到错误时，用户可通过编程修复错误并连线 VI 或函数的**错误输出**输出端至“清除错误”VI 的**错误输入**输入端。



提示 配合使用错误处理和调试工具发现并管理错误。

错误簇

VI 和函数通过下列方法之一或全部返回错误—数值错误代码或错误簇。通常，函数以数值错误代码返回错误，而 VI 以错误簇，即错误输入和错误输出来返回错误。使用错误簇输入控件和显示控件创建子 VI 错误输入和输出。



提示 程序框图上的全部错误簇通常提供相同的标准错误输入和错误输出功能。

标准**错误输入**和**错误输出**簇包含下列信息组件：

- **状态**是一个布尔值，错误产生时报告 TRUE。
- **代码**是一个 32 位有符号整数，通过数值表示错误。一个非零错误代码和 FALSE **状态**相结合可表示警告但不是错误。
- **源**是用于识别错误发生位置的字符串。

错误

错误被定义为无论**代码**值为多少，**状态**为 TRUE 的错误簇。

如 LabVIEW 检测到错误，则该节点会将错误传递到下一个节点且不执行那一部分代码。

警告

警告被定义为**代码**值非零，**状态**值为 FALSE 的错误簇。尽管多数错误的**代码**值均为负数，警告的**代码**值均为正数。但这并不能被用作通用的真理。因此仍需要依赖**状态**值和**代码**值两者共同判定错误和警告。

警告通常被认为没有错误严重。某些 API 和函数（例如，“匹配正则表达式”函数）仅报告错误。但其他 API（例如，控制独立仪器的 VISA API）仅报告警告。

与错误发生时不同，当 LabVIEW 检测到警告时，节点仍可正常执行。尽管代码可正确运行，但用户在开发过程中应监视警告，以确保应用程序的正确运行。

解释错误对话框

生成错误时，右键单击簇的边框，并从快捷菜单中选择**解释错误**打开**解释错误**对话框。**解释错误**对话框包含错误的相关信息。如 VI 只包含警告，但是没有错误，快捷菜单中就会有一个**解释警告**选项。

或者通过**帮助**»**解释错误**菜单打开**解释错误**对话框。

检测和报告错误

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。

如要完美的执行错误错误，当应用程序中生成错误时，必须判定要执行的动作。首先，必须利用函数和 VI 的错误接线端。由于错误簇被执行为流经参数，应当连线要执行的第一个节点的**错误输出**簇至要执行的下一个节点的**错误输入**簇。必须对后续的节点执行类似的操作。

VI 运行时，LabVIEW 会在每个执行节点检测错误。如 LabVIEW 没有发现任何错误，则该节点将正常执行。如 LabVIEW 检测到错误或警告，节点将错误传递至下一节点。用户创建的子 VI 也应执行该流经动作。

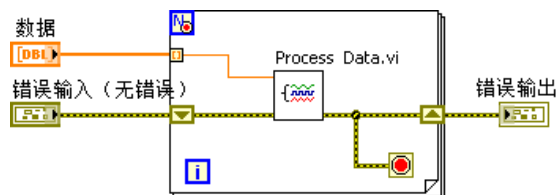


提示 在执行任何形式的输入 / 输出 (I/O) 操作时，都应考虑到发生错误的可能性。几乎所有的 I/O 函数都会返回错误信息。应在 VI 中包括错误检查，尤其对于文件、串口、仪器测量、数据采集和通讯等 I/O 操作更应如此，并提供一个恰当的错误处理机制。

传递错误和警告

如要确保正确传递错误和警告信息，为循环中的错误簇使用移位寄存器非常重要，因而警告信息可被传递至全部循环过程。关于正确使用移位寄存器传递错误和警告至下一循环的信息，见图 2-3。

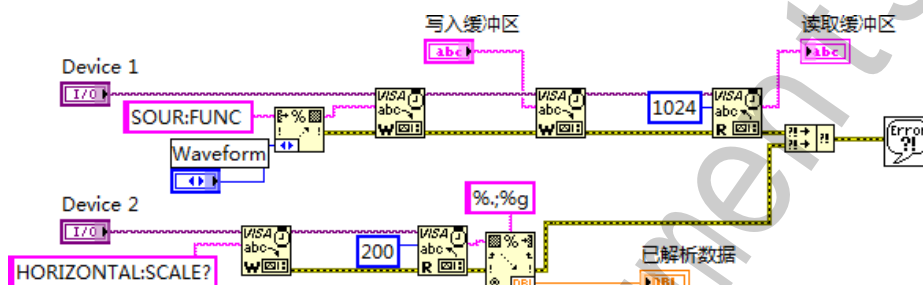
图 2-3. 使用移位寄存器传递错误和警告



合并错误和警告

使用“合并错误”函数从并行结构中合并**错误输出簇**的值。关于并行节点结构合并错误信息的范例，见图 2-4。

图 2-4. 从多个源合并错误



在应用程序的末端，即将全部错误源合并至一个错误簇后，必须使用“简易错误处理器”VI 或其他错误报告机制将错误报告给用户。

简易错误处理器

默认状态下，“简易错误处理器”VI 对话框将显示生成的所有错误的描述，且不会报告警告。但“简易错误处理器”VI 可被配置用于其他错误处理动作。通过下列选项可选择不同类型的对话框：

- **无对话框** — 不显示对话框。有助于通过程序控制错误处理。
- **确定消息（默认）** — 显示只有一个确定按钮的对话框。确认该对话框后，该 VI 将控制返回给主 VI。
- **继续或停止消息** — 显示多按钮对话框，用于停止或继续。如用户选择停止，则该 VI 调用“停止”函数停止执行。
- **确定信息 + 警告** — 显示含有警告和确定按钮的对话框。确认该对话框后，该 VI 将控制返回给主 VI。
- **继续 / 停止 + 警报** — 显示含有多条警告和按钮的对话框，用于停止或继续。如用户选择停止，则该 VI 调用“停止”函数停止执行。

错误代码的范围

LabVIEW 中的 VI 和函数返回数值错误代码。每个产品或 VI 的组合定义一个错误代码的范围。关于错误的数值错误代码表及产品 and VI 组的说明信息，见 *LabVIEW 帮助* 中的 *LabVIEW 错误代码的范围*。

除定义代码范围之外，LabVIEW 将某些错误代码预留给用户应用程序。可在 -8999 至 -8000、5,000 至 9,999 或 500,000 至 599,999 范围中自定义错误代码。

某些错误代码同时适用于一组或多组 VI 和函数。例如，错误 65 同时表示串口错误代码（表示串口超时）和网络错误代码（表示网络连接已建立）。

自测：练习

1. 下列哪项可能导致运行按钮断开？
 - a. 子 VI 已断开
 - b. 程序框图包含零除数
 - c. 必需的子 VI 输入端未连接
 - d. 布尔接线端连接至数值显示控件

2. 下列哪项或哪几项为错误簇组件和数据类型？
 - a. 状态：布尔值
 - b. 错误：字符串
 - c. 代码：32 位整数
 - d. 源：字符串

3. 所有错误的错误代码均为负值，所有警告的错误代码均为正值。
 - a. 对
 - b. 错

4. 合并错误函数可以组合来自多个错误源的错误信息。
 - a. 对
 - b. 错

International Instruments
不得转载

自测：练习答案

1. 下列哪项可能导致运行按钮断开？
 - a. 子 VI 已断开
 - b. 程序框图包含零除数
 - c. 必需的子 VI 输入端未连接
 - d. 布尔接线端连接至数值显示控件

2. 下列哪项或哪几项为错误簇内容？
 - a. 状态：布尔值
 - b. 错误：字符串
 - c. 代码：32 位整数
 - d. 源：字符串

3. 所有错误的错误代码均为负值，所有警告的错误代码均为正值。
 - a. 对
 - b. 错

4. 合并错误函数可以组合来自多个错误源的错误信息。
 - a. 对
 - b. 错

笔记

ational Instruments
不得转载

实现 VI

本课程将介绍如何在 LabVIEW 中实现代码。涉及内容包括设计用户界面、选择数据类型、注释代码、使用循环结构（例如，While 循环和 For 循环）、添加软件定时至代码、显示数据为曲线及在代码中使用条件结构。

主题

- A. 前面板基本介绍
- B. LabVIEW 数据类型
- C. 注释代码
- D. While 循环
- E. For 循环
- F. VI 定时
- G. 循环中的数据反馈
- H. 数据图表绘制一波形图表
- I. 条件结构

A. 前面板基本介绍

在软件开发的设计阶段定义设计的输入和输出。该定义将直接影响前面板窗口的设计。

设计的输入可来自下列动作：

- 采集设备。例如，数据采集设备或万用表
- 从文件直接读取
- 前面板的操作输入控件

通过前面板控件（例如，数值、布尔或字符串控件）显示设计的输入。但并非全部输入均显示在前面板上。

通过显示控件（例如，图、表和 LED）显示设计的输出或将输出记录至文件，或者使用信号生成器输出数据至设备。

设计输入控件和显示控件

选择输入控件和显示控件时，请确保其适用于要执行的任务。例如，判断正弦波的频率可选择转盘控件。显示温度时可选择温度计显示控件。

标签

请确保为输入控件和显示控件制定清晰的标签名称。标签能够帮助用户识别每个输入控件和显示控件的用途。同时，清晰的标签命名能够帮助用户注释程序框图代码。输入控件和显示控件的标签分别对应程序框图的接线端名称，如图 3-1 所示。

图 3-1. 前面板输入控件和显示控件在程序框图上的显示



输入控件和显示控件选项

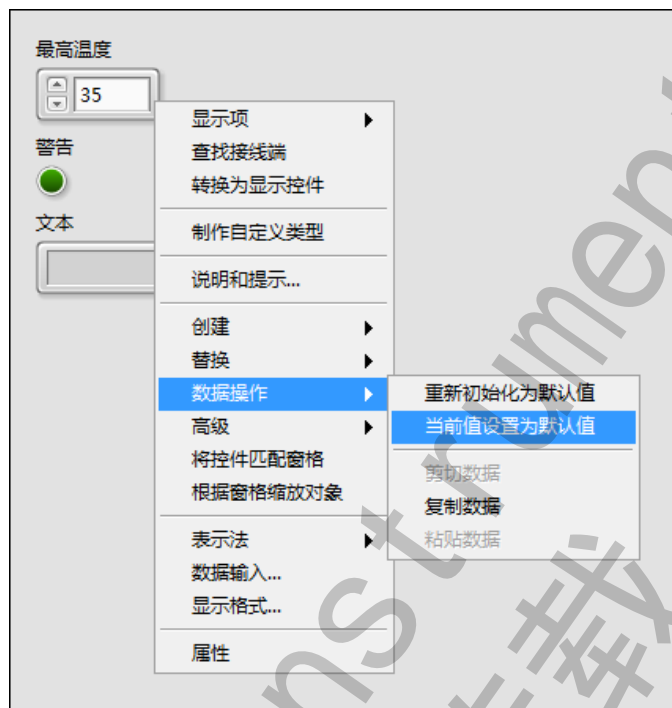
用户可设置输入控件和显示控件的默认值。图 3-2 中是默认值为 35 °C 的**最高温度**控件。通过设置默认值，用户可假设 VI 的可能值（如用户在运行过程中未设置其他值）。

按照下列步骤，设置输入控件或显示控件的默认值。

1. 输入所需的值。

2. 右键单击输入控件或显示控件，从快捷菜单中选择**数据操作**»**当前值设置为默认值**。

图 3-2. 设置默认值



提示 或者可将值重新初始化为默认值。



提示 如要在前面板同时重新初始化全部输入控件和显示控件，从 LabVIEW 菜单中选择**编辑**»**当前值设置为默认值**或**编辑**»**重新初始化为默认值**。

B. LabVIEW 数据类型

数据具有多种不同的数据类型。在第 1，“LabVIEW 导航”中您已经了解了数值、布尔和字符串数据类型。其他数据类型有枚举型和动态数据等。数值型数据本身也具有不同的数据类型。例如，整数和分数。

程序框图接线端和数据类型

程序框图接线端直观的向用户传递接线端表示的数据类型信息。例如，在 3-3 中，**高 (cm)** 为双精度、浮点型数值。这是由接线端的颜色（橙色）及接线端文本信息 **DBL** 指示的。

图 3-3. 接线端的数据类型范例



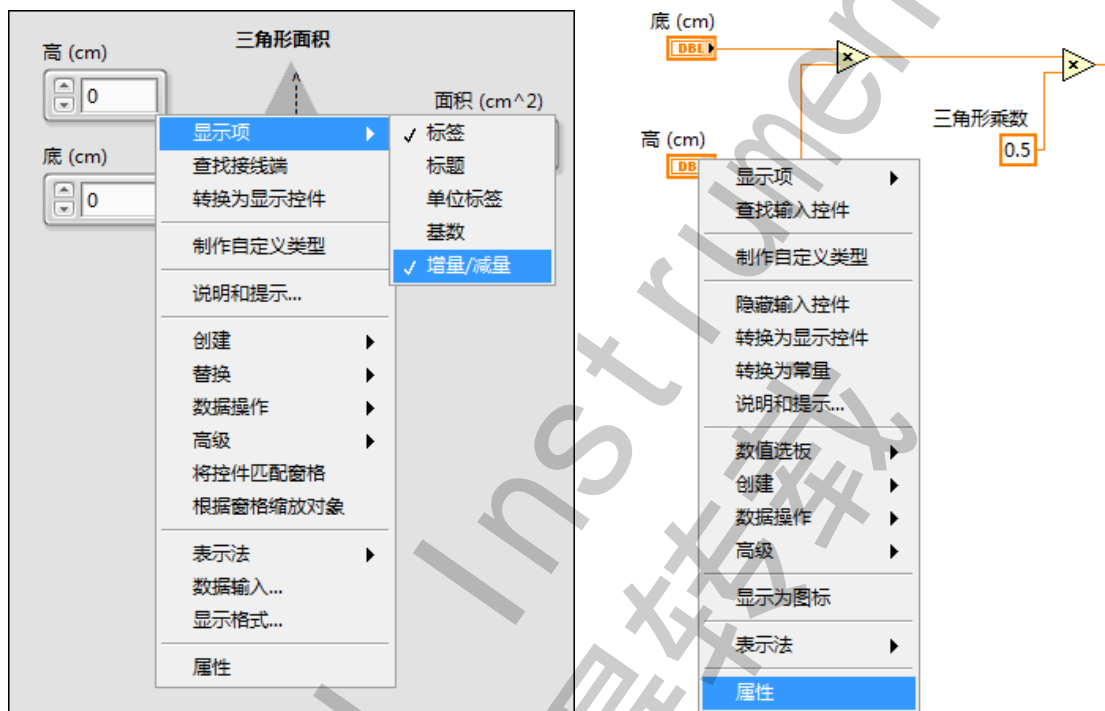
提示 接线端名称对应于前面板输入控件和显示控件的标签。右键单击接线端，从快捷菜单中选择**查找输入控件**或**查找显示控件**，以在前面板上定位输入控件或显示控件的位置。

快捷菜单

所有 LabVIEW 对象都有相关的快捷菜单，也叫即时菜单、弹出菜单或右键单击菜单。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运行方式。右键单击对象，查看快捷菜单。

图 3-4 为输入控件接线端的快捷菜单。

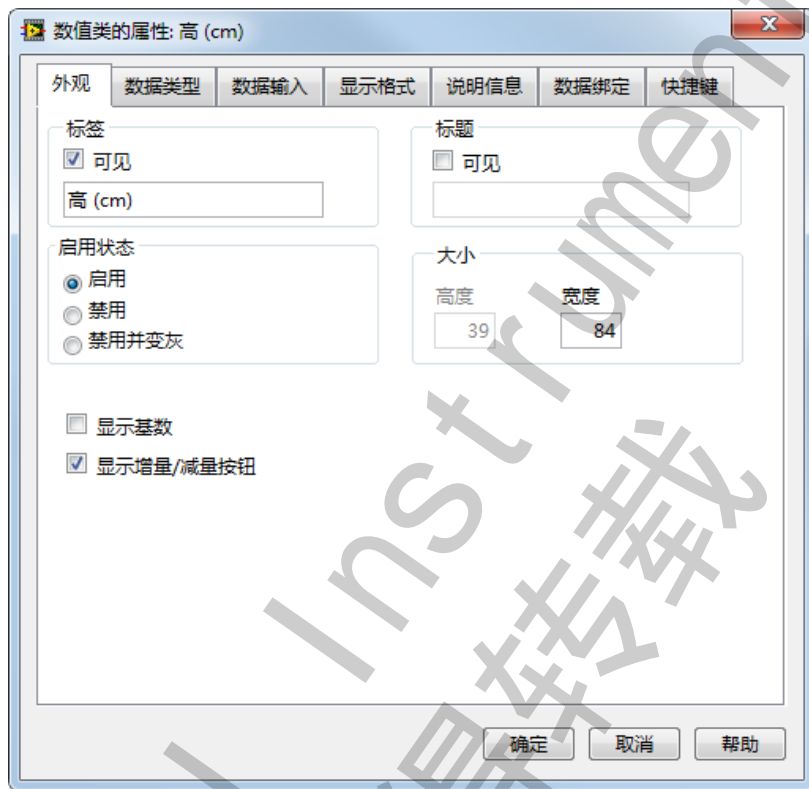
图 3-4. 前面板和程序框图对象的快捷菜单



属性对话框

对象也具有属性对话框，可通过该对话框更改对象的外观或动作。右键单击对象，从快捷菜单中选择**属性**，打开对象的属性对话框。图 3-5 为图 3-4 中的**高 (cm)** 接线端的属性对话框。对象的属性对话框中的可用选项与对象快捷菜单中的选项类似。

图 3-5. 数值接线端的属性对话框



可选择前面板或程序框图上的多个对象，编辑这些对象共有的属性。如要选择多个对象，使用定位工具在将全部要编辑的对象拖放在一个长方形区域内，或按下 <Shift> 键单击选择全部对象。右键单击选中的一个对象，从快捷菜单中选择**属性**，打开**属性**对话框。**属性**对话框仅显示选中对象共有的选项卡和属性。选择相似的对象可显示多个选项卡和属性。如选择的多个对象没有公共属性，**属性**对话框中不会显示任何选项卡或属性。

数值数据类型

数值型数据可表示不同类型的数值。如要更改数值的表示法，右键单击输入控件、显示控件或常量并选择**表示法**。如图 3-6 所示。

图 3-6. 数值表示法



如果把两个或多个不同表示法的数值输入连接到一个函数，函数返回的输出数据将使用涵盖范围较大的格式。函数在执行前会自动将短精度表示法强制转换为长精度表示法。在 LabVIEW 中发生数值转换的接线端端口，将出现一个强制转化点。详细信息见“数值转换”。

数值型数据类型包含下列表示法子类别—浮点型、有符号整数、无符号整数和复数。

浮点数

浮点数表示小数数值。在 LabVIEW 中，浮点数是以橙色表示。

单精度 (SGL) 一单精度浮点数为 32 位 IEEE 单精度格式。内存空间有限，且不会出现数值范围溢出时，应使用单精度浮点数。

双精度 (DBL) 一双精度浮点数为 64 位 IEEE 双精度格式。双精度时数值对象的默认格式。大多数情况下应使用双精度浮点数。

扩展精度 (EXT) — 保存扩展精度数到磁盘时，LabVIEW 将其保存为独立于平台的 128 位格式。内存中，大小和精度根据平台有所不同。仅在必需时，才使用扩展精度的浮点数。扩展精度算术的运行速度根据平台有所不同。

定点数据类型

定点数据类型是一种以二进制数（又称“位”）表示一组有理数的数值数据类型。浮点型数据允许 LabVIEW 表示数值的总比特位数发生变化。而定点数通常被配置为使用固定的比特位数。只能储存和处理限定或固定数量比特数据的硬件和终端也可存储和处理定点数据。可指定定点数据的范围和精度。



注 以浮点数据类型表示有理数时，由于二进制数的基数是 2，故有理数的分母必须为 2 的幂。

不需要浮点表示法的动态功能时，或在使用不支持浮点算术的终端时（如 FPGA 终端），可使用定点数据类型。

可为定点数指定其编码、字长和整数字长，以满足特定的大小要求。

编码—指定定点数的二进制编码方式。可选择有符号或无符号。如选择有符号，则符号位总在表示数据的位字符串的第一位。

字长—一位字符串的总位数，是 LabVIEW 表示定点数据时的位数。LabVIEW 接受的最大字长为 64 位。某些终端可能仅支持字长较短的数据。如在终端上打开一个 VI 而 VI 包含的定点数大于终端所能接受的字长，则 VI 出现断线。参考终端的相关文档，确定终端接受的最大字长。

整数字长—指位字符串中 LabVIEW 用于表示定点数的整数位数，或从最高有效位左侧或右侧的初始位置开始，将二进制小数点移动到最高有效位所需的位数。整数字长的长度可以大于字长，可以为正也可以为负。

整型

整型表示整个数字。有符号整型可以为正数，也可以是负数。如已知整型总是为正数，可使用无符号整型数据类型。在 LabVIEW 中，整型以蓝色表示。

LabVIEW 将浮点型转化为整型时，VI 将把数字舍入到最近的整数。如值为两个整数的中间值，该函数可返回最近的偶数。

单字节整型 (I8) — 单字节整型的存储空间为 8 位，范围是 -128 至 127。

双字节整型 (I16) — 双字节整型的存储空间为 16 位，范围是 -32,768 至 32,767。

长整型 (I32) — 长整型的存储空间为 32 位，范围是 -2,147,483,648 至 2,147,483,647。多数情况下应优先选择使用 32 位整型。

64 位整型 (I64) — 64 位整型的存储空间为 64 位，范围是 -1e19 至 1e19。

单字节整型 (U8) — 单字节无符号整型的存储空间为 8 位，范围是 0 至 255。

双字节整型 (U16) — 双字节无符号整型的存储空间为 16 位，范围是 0 至 65,535。

长整型 (U32) — 无符号长整型的存储空间为 32 位，范围是 0 至 4,294,967,295。

64 位整型 (U64) — 无符号 64 位整型的存储空间为 64 位，范围是 0 至 2e19。

复数

复数是将实部与虚部相连接的浮点数。在 LabVIEW 中，由于复数为浮点型，因此复数也表示为橙色。复数有三种类型。

单精度浮点复数 (CSG) — 单精度浮点复数由 32 位二进制 IEEE 单精度的实数和虚数组成。

双精度浮点复数 (CDB) — 双精度浮点复数由 64 位二进制 IEEE 单精度的实数和虚数组成。

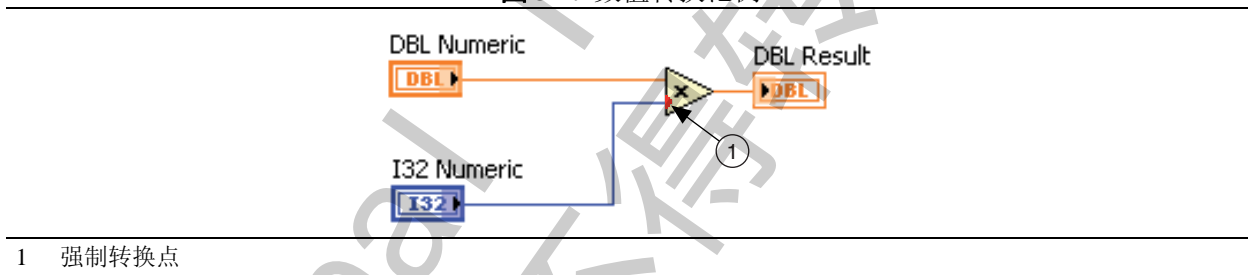
扩展精度浮点复数 (CXT) — 扩展精度浮点复数由 IEEE 扩展精度的实数和虚数组成。在内存中，扩展型精度数值的大小和精度根据平台有所不同。在 Windows 平台使用 128 位 IEEE 扩展型精度格式。

数值转换

LabVIEW 可将数值数据类型表示为有符号或无符号整数、浮点型数值或复数值。一般来说，当函数输入端连接不同表示类型数据时，函数返回的输出数据将使用涵盖范围较大的格式。同时使用有符号整数和无符号整数时，LabVIEW 将强制转换数值为无符号整数。同时使用无符号整数和浮点型数值时，LabVIEW 将强制转换数值为浮点型。同时使用浮点型数值和复数值时，LabVIEW 将强制转换数值为复数值。使用同一数据类型但比特位宽度不同的数值时，LabVIEW 将强制转换数值为两者中数值宽度较大的格式。

如比特位宽度相同，LabVIEW 将在无符号整数和有符号整数中，优先选择无符号整数。例如，连线 DBL 和 I32 至“乘”函数输入端时，输出结果为 DBL。如图 3-7 所示。由于有符号整型数值相对于双精度浮点型数值使用较少的比特位，因此 LabVIEW 将强制转换 32 位有符号整型为双精度浮点型格式。“乘”函数的较低输入端显示了一个红色的点（强制转换点），其表明 LabVIEW 执行了强制转换数据的操作。

图 3-7. 数值转换范例



布尔值

LabVIEW 用 8 位二进制数保存布尔数据。如 8 位值为零，则布尔值为 FALSE。所有非零的值都表示 TRUE。在 LabVIEW 中，布尔数据表示为绿色。

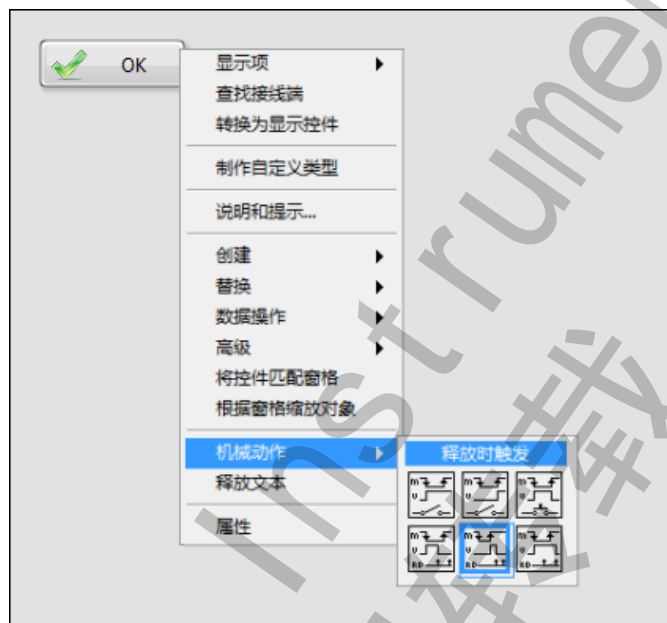
布尔值具有相关联的机械动作。两个主要的动作为触发和转换。有以下按钮动作供选择：

- **单击时转换** — 每次以操作工具单击控件时，控件值改变。VI 读取该控件值的频率与该动作无关。
- **释放时转换** — 仅当在控件的图片边界内单击一次鼠标后放开鼠标按钮时，控件值改变。VI 读取该控件值的频率与该动作无关。
- **保持转换直到释放** — 单击控件时改变控件值，保留该控件值直到鼠标按钮释放。此时控件将返回至其默认值。与门铃相似。VI 读取该控件值的频率与该动作无关。单按钮控件不可选择该动作。
- **单击时触发** — 单击控件时改变控件值，保留该控件值直到 VI 读取该控件。此时，即使长按鼠标按钮控件也将返回至其默认值。该动作与断路器相似，适用于停止 While 循环或令 VI 在每次用户设置控件时只执行一次。单按钮控件不可选择该动作。

- **释放时触发**—仅当在控件的图片边界内单击一次鼠标后放开鼠标按钮时，控件值改变。VI 读取该动作一次，则控件返回至其默认值。该动作与对话框按钮和系统按钮的动作相似。单按钮控件不可选择该动作。
- **保持触发直到释放**—单击控件时改变控件值，保留该控件值直到 VI 读取该值一次或用户释放鼠标按钮，取决于二者发生的先后。单按钮控件不可选择该动作。

如要了解机械动作的详细信息，请练习 NI 范例查找器中的 Mechanical Action of Booleans VI。

图 3-8. 布尔机械动作



字符串

字符串是可显示的或不可显示的 ASCII 字符序列。字符串提供了一个独立于操作平台的信息和数据格式。常用的字符串操作包括：

- 创建简单的文本信息。
- 发送文本命令至仪器，以 ASCII 或二进制字符串的形式返回数据，然后转换为数值，从而控制仪器。
- 将数值数据存储到磁盘。如需将数值数据保存到 ASCII 文件中，须在数值数据写入磁盘文件前将其转换为字符串。
- 用对话框指示或提示用户。

在前面板上，字符串以表格、文本输入框和标签的形式出现。LabVIEW 提供了用于对字符串进行操作的内置 VI 和函数，可对字符串进行格式化、解析字符串等编辑操作。

关于 ASCII 码和转换函数的详细信息，见 *LabVIEW 帮助* 中的 *ASCII 码* 主题。

在 LabVIEW 中，字符串以粉色表示。

右键单击前面板上的字符串输入控件或显示控件，从表 内的显示类型中选择。下表是各种显示类型的范例。

表 3-1. 字符串显示类型

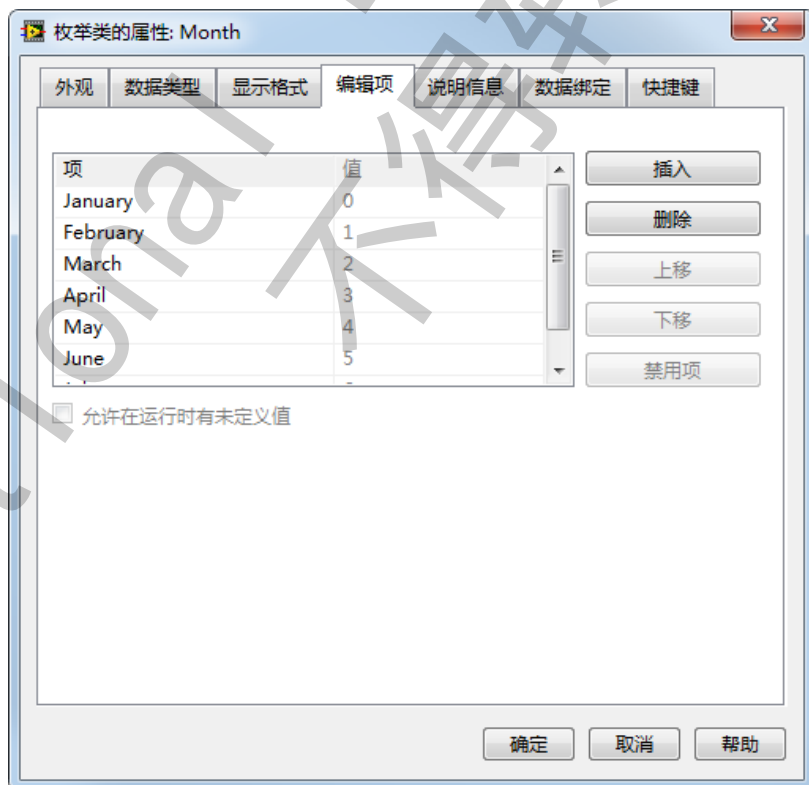
显示类型	说明	消息
正常显示	可打印字符以控件字体显示。不可显示字符通常显示为一个小方框。	There are four display types.\ is a backslash.
'\ ' 代码显示	所有不可显示字符显示为反斜杠。	There\sare\sfour\sdisplay\stypes.\n\\\sis\sa\ sbackslash.
密码显示	每一个字符（包括空格在内）显示为星号 (*)。	***** *****
十六进制显示	每个字符显示为其十六进制的 ASCII 值，字符本身并不显示。	5468 6572 6520 6172 6520 666F 7572 2064 6973 706C 6179 2074 7970 6573 2E0A 5C20 6973 2061 2062 6163 6B73 6C61 7368 2E

LabVIEW 将字符串保存为指向某个结构的指针，该结构包含一个长度为 4 个字节的值和一个一维单字节整数（8 位二进制字符）数组。

枚举型

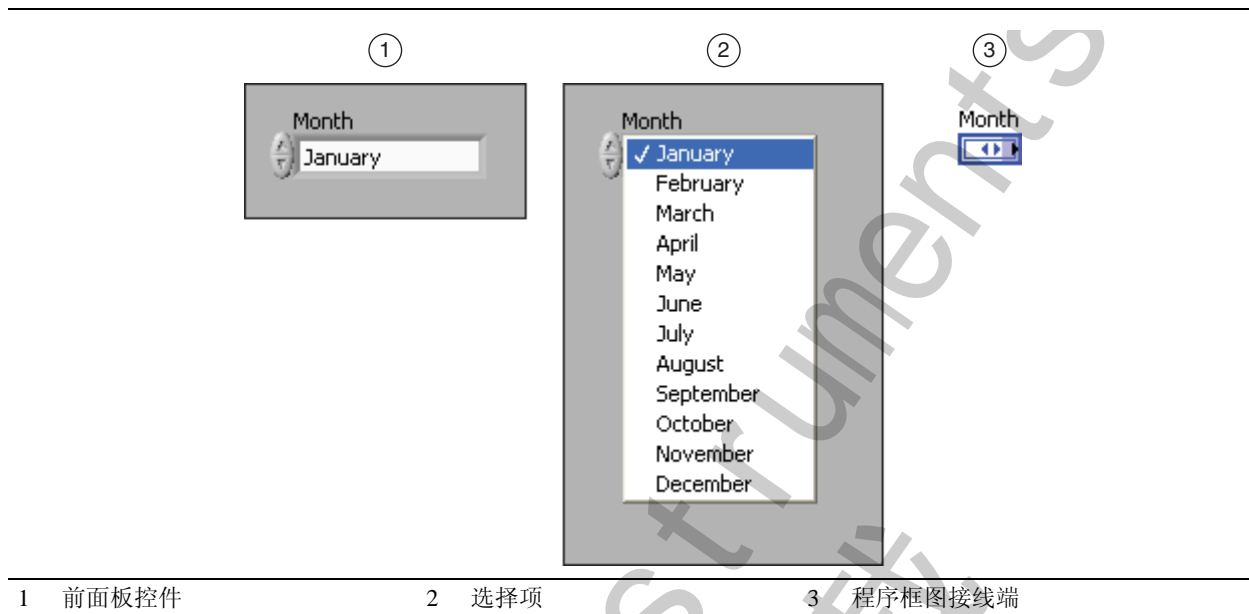
枚举型（枚举型输入控件、常量或显示控件）为数据类型的组合。枚举型表示一对值（字符串和数值数据），枚举数为一组值。例如，创建名称为 **Month** 的枚举值，可能的枚举值对为 January-0、February-1 直至 December-11。图 3-9 为枚举型控件的**属性**对话框中的数据对的范例。

图 3-9. Month 枚举型控件的属性



枚举非常有用，因为在程序框图上处理数字比处理字符串简单得多。图 3-10 为 **Month** 枚举型控件、枚举控件的数据对选择和相应的程序框图接线端。

图 3-10. Month 枚举控件



动态

动态数据保存由 ExpressVI 生成或采集的信息。动态数据类型显示为深蓝色接线端。如下所示。多数 Express VI 可接收及返回动态数据类型。动态数据类型可连接到任何接收数值、波形或布尔数据的显示控件或输入。应将动态数据类型连接到最能表述该数据的显示控件上。如图形、图表或数字显示控件。



LabVIEW 中的多数 VI 和函数不接受动态数据类型。如要使用内置 VI 或函数分析及处理动态数据类型中的数据，必须转换动态数据类型。

“从动态数据转换” Express VI 可将动态数据转换为数值、布尔、波形和数组数据，以便在其他 VI 和函数中使用。在程序框图上放置“从动态数据转换” Express VI 后，将出现**配置从动态数据转换**对话框。**配置从动态数据转换**对话框可指定如何格式化“从动态数据转换” Express VI 返回的数据。



连线动态数据类型至数组显示控件时，LabVIEW 自动添加“从动态数据转换” Express VI 至程序框图。双击“从动态数据转换” Express VI 打开**配置从动态数据转换**对话框，配置数据在数组中的显示形式。

C. 注释代码

维护和修改 VI 的专业开发人员了解清晰的为代码添加注释的重要性。清晰的注释程序框图代码将降低后续修改代码的难度。此外，清晰的注释前面板窗口可以解释 VI 的意义及前面板对象。

使用提示框、说明、VI 属性和优良的设计注释前面板窗口。

提示框和说明

提示框是 VI 运行时，光标移至控件时显示的简单说明。例如，添加提示框说明温度为摄氏度或解释输入端的算法。说明为指定控件提供了附加信息。光标移至对象时，**即时帮助**窗口将出现说明信息。如要添加提示框和说明信息至控件，右键单击控件并从快捷菜单中选择**说明和提示**。

VI 属性

使用 **VI 属性**对话框的注释部分创建 VI 说明，在 VI 和 HTML 或已编译帮助文档之间建立链接。在前面板或程序框图上右键单击 VI 图标，从快捷菜单中 **VI 属性**或选择**文件 » VI 属性**显示 VI 属性。然后从**类别**下拉菜单中**说明信息**。VI 运行时不能访问该对话框。

该页由下列部分组成：

- **VI 属性**—光标移至 VI 图标时，**即时帮助**窗口可显示输入的文本信息。在文本信息的前后分别添加 和 ，可用粗体显示文本信息。或者使用 VI 说明属性，通过编程编辑 VI 属性。
- **帮助标识符**—包含链接至已编译帮助文件的 HTML 文件名或主题的索引关键词。或者使用帮助：文档标识符属性，通过编程设置帮助标识符。
- **帮助路径**—包含链接至“即时帮助”窗口的 HTML 文件，或已编译帮助文件的路径。如该栏为空，即时帮助窗口不显示**详细帮助信息**链接，**详细帮助信息**按钮为灰色。
- **浏览**—显示文件对话框，浏览要链接的 HTML 文件或用作帮助路径的已编译帮助文件。

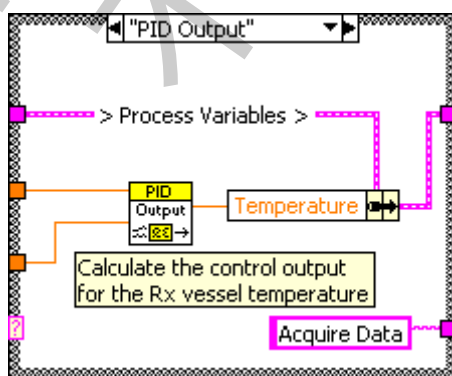
命名输入控件和显示控件

赋予输入控件和显示控件合理的描述性名称可增强前面板的可用性。例如，如命名某输入控件为温度，用户可能未知应使用的温度单位。但如命名控件为温度 °C 就增强了前面板的可知信息。用户可使用公制单位输入温度。

图形化编程

LabVIEW 的图形化特性有助于程序框图的自释性，额外的标注将有助于后续的 VI 的修改。程序框图有两种类型的注释—函数或算法的注释，及位于连线间解释数据意义的注释。两种类型的注释均出现在下列程序框图中。通过标签工具可插入标准标签，或**函数 » 编程 » 结构 » 修饰**插入自由标签。默认情况下，自由标签的背景色填充为黄色。

图 3-11. 程序框图注释



使用下列规则为 VI 添加注释：

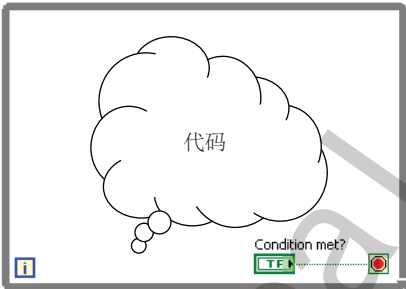
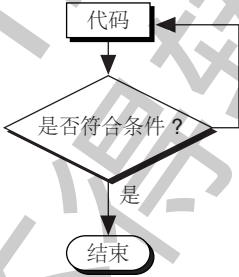
- 在程序框图上添加注释，说明代码要完成的功能。
- 由于 LabVIEW 的图形化代码具有一定的自释性，可使用标签注释程序框图的功能。
- 使用标签识别前面板和程序框图对象。LabVIEW 包含两种类型的标签—自带标签和自由标签。自带标签属于某一特定对象，并随对象移动，仅用于注释该对象。自由标签不附属于任何对象，用户可单独创建、移动、旋转或删除自由标签。
- 请勿显示函数和子 VI 调用的标签，因为他们通常比较大而不够简洁。通过即时帮助窗口，开发人员在查看程序框图时可了解函数或子 VI 的名称。
- 在较长的连线中使用标签标识其功能。对于移位寄存器或贯穿整个程序框图的较长连线使用标签非常有用。关于移位寄存器的详细信息，见本课的“条件结构”章节。
- 对结构使用标签，指定结构的主要功能。
- 对常量使用标签，指定常量的特性。
- 使用自由标签注释程序框图的算法。如使用来自于书籍或其他引用的算法，请提供参考信息。

D. While 循环

与文本编程语言的 Do 循环或 Repeat-Until 循环类似，While 循环将执行子程序框图直到满足某一条件。如图 3-12 所示。

下表分别为 LabVIEW 中的 While 循环，While 循环的对应流程图及 While 循环的伪码范例。

图 3-12. While 循环

 <p>①</p>	 <p>②</p>	<pre>Repeat (code); Until Condition met; End;</pre> <p>③</p>
1 LabVIEW While 循环	2 流程图	3 伪码

While 循环位于**结构**选板。从选板中选择 While 循环，在程序框图上拖拽出所需的矩形 While 循环区域。释放鼠标按钮时，While 循环将围绕刚才所选中的区域。

将程序框图对象拖拽至 While 循环内。

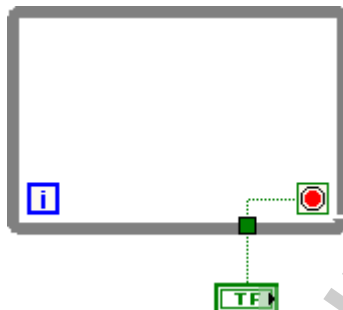


提示 While 循环至少执行一次。

While 循环执行子程序框图直到条件接线端（输入端）接收到某一特定的布尔值。While 循环没有固定的循环总数，如结束条件不发生，While 循环将无限次执行。

如条件接线端是**真 (T) 时停止**，在 While 循环外放置一个布尔控件。循环开始时布尔控件为 FALSE，该循环就是一个无限循环，见下列范例。如将该布尔控件设置为 TRUE，条件接线端为**真 (T) 时继续**，仍然会导致无限循环。如图 3-13 所示。

图 3-13. 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**中止执行**按钮中止整个 VI。

使用 While 循环的条件接线端也可执行基本的错误处理。将错误簇连接到条件接线端时，仅有错误簇中**状态**参数的 TRUE 或 FALSE 值被传递到该接线端，并且**真 (T) 时停止**和**真 (T) 时继续**快捷菜单选项也相应地分别变为**错误时停止**和**错误时继续**。

循环计数接线端是一个输出接线端，表示已完成的循环次数。

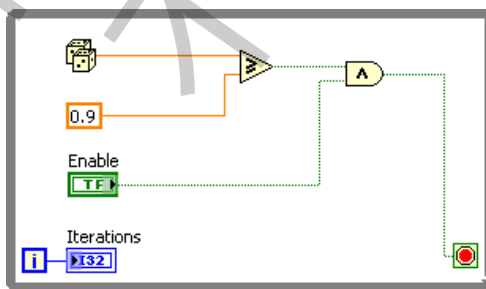
II

While 循环的循环计数始终从零开始。

在下列程序框图中，While 循环执行至“随机数”函数的输出端大于或等于 0.9 且 **Enable** 控件的值为真。仅当两个输入端的值均为真时，“与”函数返回真。否则，返回假。

下列范例增加了无限循环的可能性。通常仅需满足条件之一即可停止循环，而不需要同时满足两个条件。

图 3-14. 可能导致无限循环

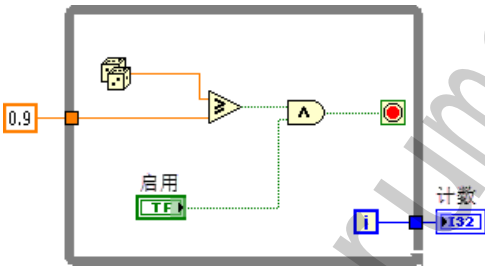


结构隧道

隧道从结构外接收数据及将数据输出结构。隧道在 While 循环的边框上显示为实心方块。该方块的颜色为连接至隧道的数据类型颜色。循环结束后，数据传出隧道。隧道向循环传送数据时，需所有数据均到达隧道后，循环才能执行。

在下列程序框图中，循环计数接线端连接至隧道。While 循环执行结束后，隧道中的值才会被传输至计数显示控件。

图 3-15. While 循环隧道



循环计数接线端的最终值会被显示在计数显示控件中。

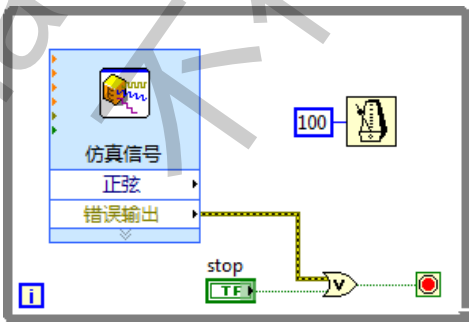
通过 While 循环进行错误检查和错误处理

连线错误簇至 While 循环的条件接线端或带有条件接线端的 For 循环，以停止循环计数。连线错误簇至条件接线端时，仅错误簇的状态参数（TRUE 或 FALSE）会传递至接线端。生成错误时，循环中止。

将一个错误簇连接到条件接线端上时，快捷菜单项**真 (T) 时停止**和**真 (T) 时继续**将变为**错误时停止**和**错误时继续**。

在图 3-16 中，同时使用错误簇和停止按钮判定何时停止循环。这是用于停止多数循环的推荐方式。

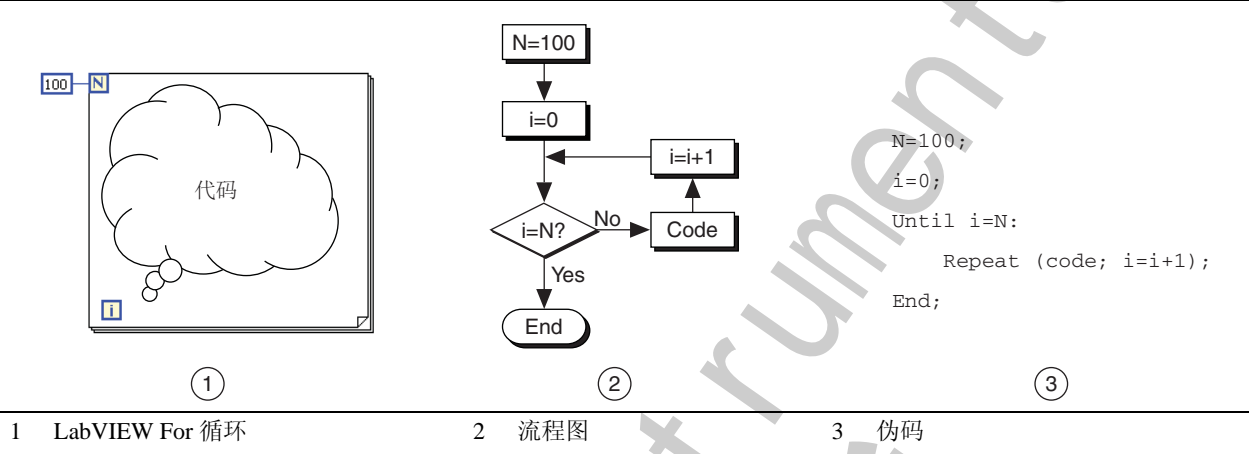
图 3-16. 停止 While 循环



E. 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。图 3-17 分别为 LabVIEW 中的 For 循环、For 循环的对应流程图及 For 循环的伪码范例。

图 3-17. For 循环



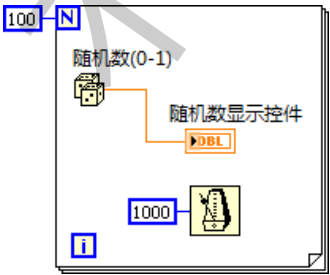
For 循环位于**结构**选板。如用户在程序框图上放置的 While 循环，右键单击 While 循环的边框并从快捷菜单中选择**替换为 For 循环**，可将 While 循环替换为 For 循环。总数接线端是一个输入接线端，它的值表示重复执行该子程序框图的次数。

循环计数接线端是一个输出接线端，表示已完成的循环次数。

For 循环的循环计数始终从零开始。

图 3-18 中的 For 循环每秒生成一个随机数，持续 100 秒并在数值显示控件中显示随机数。

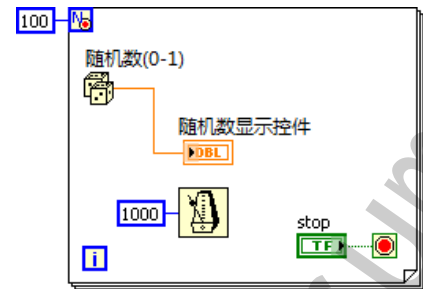
图 3-18. For 循环范例



在 For 循环中添加条件接线端

可为 For 循环添加一个条件接线端，从而在出现布尔条件或发生错误时循环停止。有条件接线端的 For 循环在满足条件或所有循环结束时停止操作，以先实现的条件为准。如 For 循环被配置为条件退出，循环的总数接线端带有一个红色符号，循环的右下角有一个条件接线端。循环与图 3-19 类似。下图中的 For 循环每秒生成一个随机数，持续 100 秒或当用户单击停止按钮时结束。

图 3-19. 配置为条件退出的 For 循环



右键单击 For 循环边框，在快捷菜单中选择条件接线端，可为 For 循环添加条件接线端。连线条件接线端和总数接线端。

图 3-20. 在 For 循环中添加条件接线端



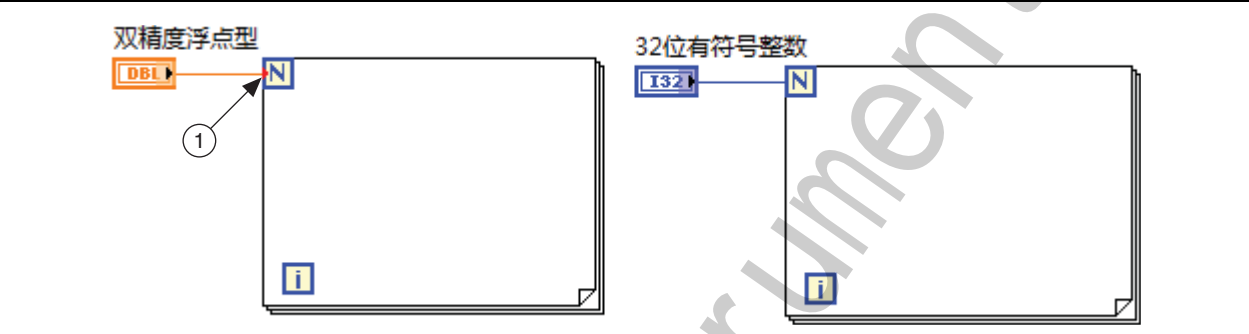
使用 For 循环进行错误检查和错误处理

对于具有条件接线端的 For 循环，还必须为总数接线端连接一个值或对一个输入数组进行自动索引以设置循环的最大次数。当发生一个错误或设置的循环次数完成后，For 循环即停止运行。详细信息见“通过 While 循环进行错误检查和错误处理”。

For 循环中的数值转换

按照“数值转换”中提及的，连线不同的数据类型至函数输入端时，函数返回的输出数据将使用涵盖范围较大的格式。如连线双精度浮点型数值至 For 循环的 32 位总数接线端，LabVIEW 将强制转换较大的数值为较小范围的 32 位有符号整数。该转换不同于常见的数据转换标准，因为 For 循环仅能执行整数次循环。

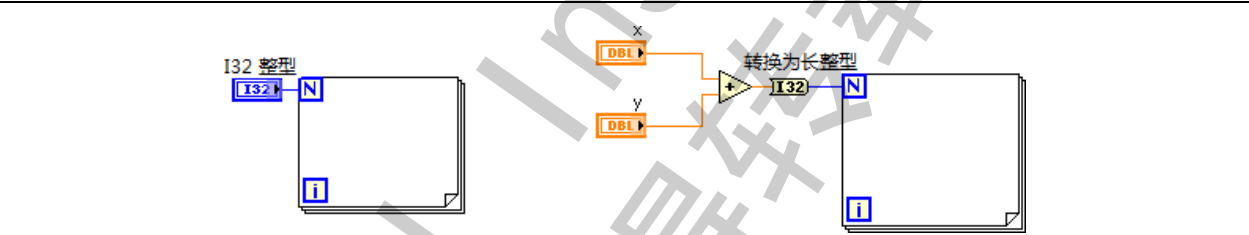
图 3-21. For 循环的强制转换



1 强制转换点

为了获得更好的性能，尽量使用匹配的数据类型或通过编程转换为匹配的数据类型，以避免使用数据强制转换。如图 3-22 所示。

图 3-22. 使用匹配的数据类型，避免数据强制转换



For 循环和 While 循环的比较

For 循环和 While 循环存在几个显著区别。关于 For 循环和 While 循环的区别列表，见表 3-3。

表 3-2. For 循环和 While 循环的比较

For 循环	While 循环
如未添加条件接线端，For 循环按照设定的次数执行	仅当条件接线端接收到符合条件的值时才停止执行
可执行 0 次	至少执行一次
隧道自动输出一个数组	隧道自动输出最后一次执行的值

F. VI 定时

循环完成一次计数后，将立即执行下一次计数。除非遇到停止条件。通常需要控制计数频率或定时。例如，需要每 10 秒采集一次数据，则需要控制循环计数的定时，使其每 10 秒发生一次。

即使动作不需要在特定的频率发生，也必须为处理器提供执行其他任务的时间（例如，响应用户界面）。本章介绍了一些定时循环的方法。

等待函数

循环内部的等待函数可使 VI 在某段时间内处于休眠状态。这段时间内，处理器可分配其他任务。等待函数使用操作系统的毫秒时钟。

“等待下一个整数倍毫秒”函数监视毫秒计数器，等待直至毫秒计数器的值为毫秒倍数中指定的值。该函数用于同步各操作。在循环中使用该函数控制循环的执行速率。如要该函数有效，代码执行时间必须小于该函数中指定的时间。循环的第一次计数的执行速率不确定。



“等待 (ms)”函数等待直至毫秒计数器的值为输入端指定的值。该函数确保循环的执行速率至少为输入端指定的值。



注 “时间延迟”Express VI 与“等待 (ms)”函数类似，但增加了内置的错误簇。关于错误簇详细信息，见第 2 章“疑难解答和调试 VI”。

已用时间

在某些情况下，需要判定 VI 中的已用时间。“已用时间”Express VI 显示自指定的开始时间起的已用时间。VI 执行的过程中，该 Express VI 跟踪记录时间。Express VI 不给处理器处理其他任务的时间。在气象站课程中的项目中将使用“已用时间”Express VI。



G. 循环中的数据反馈

编程循环时，在 LabVIEW 中经常需要访问上一个循环的数据。例如，采集循环的每次计数的各数据并计算每 5 个数据的平均值，用户必须能够获取循环中前一次计数的数据。



注 反馈节点是获取前一次循环信息的方法。关于反馈节点的详细信息，见 *LabVIEW 帮助* 的 *反馈节点* 主题。

移位寄存器

移位寄存器相当于文本编程语言中的静态变量。

移位寄存器可用于将上一次循环的值传递至下一次循环。移位寄存器以一对接线端的形式出现，分别位于循环两侧的边框上，位置相对。



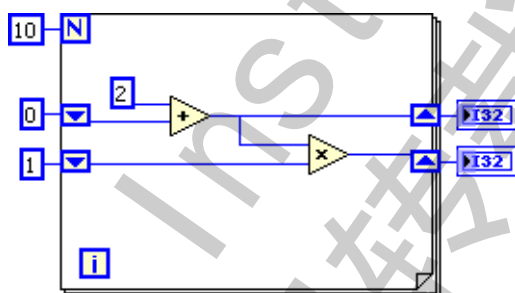
右侧接线端含有一个向上的箭头，用于存储每次循环结束时的数据。LabVIEW 可将连接到右侧寄存器的数据传递到下一次循环中。循环执行后，右侧接线端将返回移位寄存器保存的值。

右键单击循环的左侧或右侧边框，并从快捷菜单中选择**添加移位寄存器**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，并和与其连接的第一个对象的数据类型自动保持一致。连接到各个移位寄存器接线端的数据必须属于同一种数据类型。

循环中可添加多个移位寄存器。如下图所示，如循环中的多个操作都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。

图 3-23. 使用多个移位寄存器



初始化移位寄存器

初始化移位寄存器，即重设 VI 运行时移位寄存器传递给第一次循环的值。将输入控件或常量连接到循环左侧的移位寄存器接线端，即可初始化移位寄存器。如图 3-24 所示。

图 3-24. 初始化移位寄存器

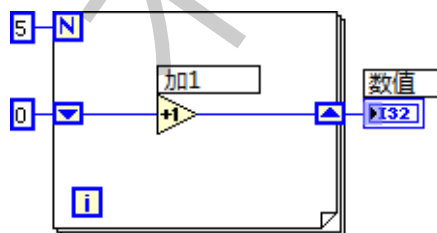


图 3-24 中的 For 循环将执行 5 次，每次循环后，移位寄存器的值都增加 1。For 循环完成 5 次循环后，移位寄存器会将最终值 (5) 传递给显示控件并结束 VI 运行。每次执行该 VI，移位寄存器的初始值均为 0。

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

使用未初始化的移位寄存器还可以保留 VI 多次执行之间的状态信息。图 3-25 即是未初始化的移位寄存器。

图 3-25. 未初始化的移位寄存器

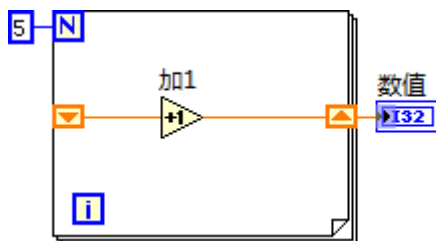


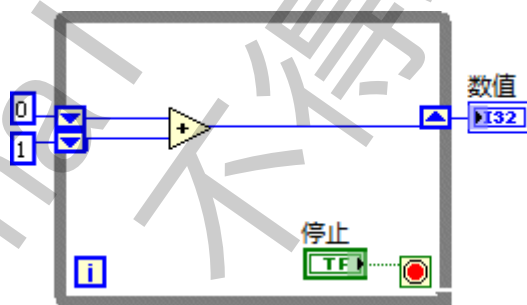
图 3-25 中的 For 循环将执行 5 次，每次循环后，移位寄存器的值都增加 1。第一次运行 VI 时，移位寄存器的初始值为 0，即 32 位整型数据的默认值。For 循环完成 5 次循环后，移位寄存器会将最终值 (5) 传递给显示控件并结束 VI 运行。而第二次运行该 VI 时，移位寄存器的初始值是上一次循环所保存的最终值 5。For 循环执行 5 次后，移位寄存器会将最终值 (10) 传递给显示控件。如果再次执行该 VI，移位寄存器的初始值是 10，依此类推。关闭 VI 之前，未初始化的移位寄存器将保留上一次循环的值。

层叠移位寄存器

层叠移位寄存器可访问以前多次循环的数据。层叠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。如需创建层叠移位寄存器，右键单击左侧的接线端并从快捷菜单中选择**添加元素**。

如图 3-26 所示，层叠移位寄存器只位于循环左侧，因为右侧的接线端仅用于把当前循环的数据传递给下一次循环。

图 3-26. 使用层叠移位寄存器

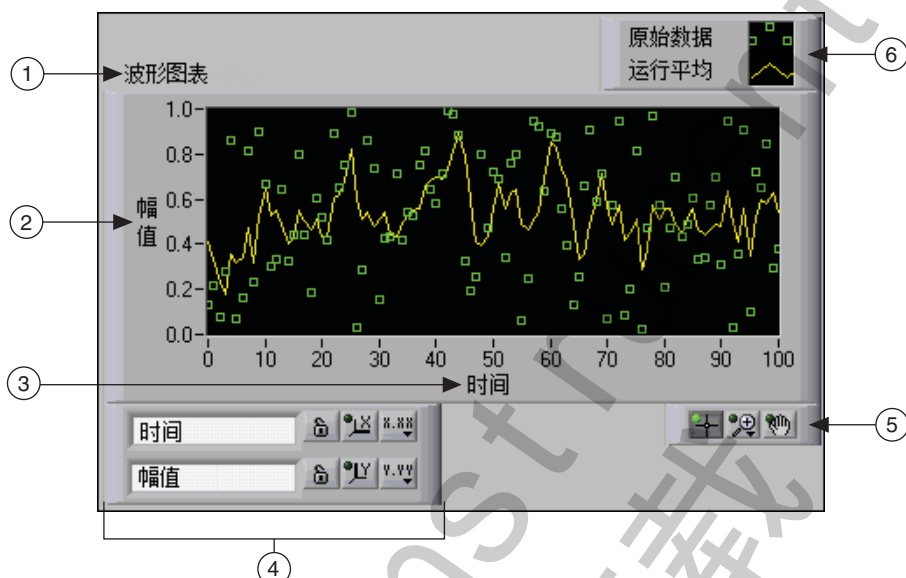


在上述程序框图中，如在左侧接线端上再添加一个移位寄存器，则上两次循环的值将传递至下一次循环中，其中最近一次循环的值保存在上面的寄存器中，而上一次循环传递给寄存器的值则保存在下面的接线端中。

H. 数据图表绘制—波形图表

波形图表是显示一条或多条曲线的特殊数值显示控件，一般用于显示以恒定速率采集到的数据。波形图表可显示单个或多条曲线。图 3-27 为多个图形图表元素。两条曲线分别为：原始数据和运行平均。

图 3-27. 波形图表元素



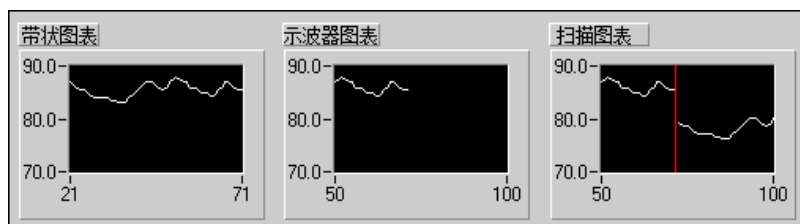
1 标签	3 X 标尺	5 图形工具选板
2 Y 标尺	4 标尺图例	6 图例

配置图表显示新数据的方式。右键单击该图表，从快捷菜单中选择**高级»刷新模式**可配置图表刷新模式。图表的数据显示方式如下：

- **带状图表**—从左到右连续滚动显示运行数据，旧数据在左，新数据在右。带状图表类似纸带图形记录器。图表的默认刷新模式为**带状图表**。
- **示波器图表**—从左到右滚动显示某一项数据（例如，脉冲或波形）。图表将新数值绘制到前一个数值的右边。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**—类似于示波器图表。二者的不同之处在于，扫描图表中旧数据在右新数据在左，并有一条垂直线将这两部分数据隔开；其次，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 (EKG)。

图 3-28 为每个图表刷新模式的范例。示波器图表和扫描图表的重新跟踪显示类似于示波器。由于重新跟踪图表需要较少的系统开销，示波器图表和扫描图表的显示速度明显快于带状图表。

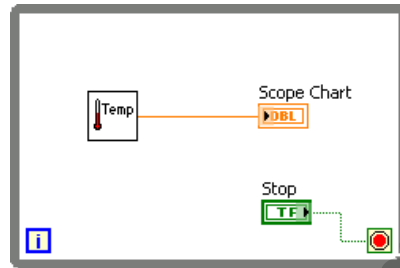
图 3-28. 图表刷新模式



连线图表

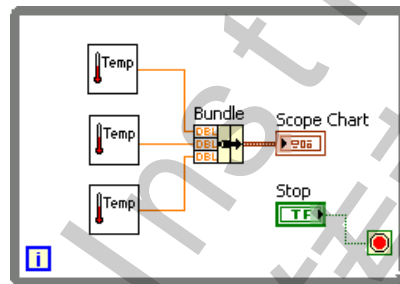
可直接连接标量输出至波形图表。图 3-29 中的波形图表接线端与输入数据类型匹配。

图 3-29. 连线单条曲线至波形图表



使用**簇、类与变体**选板中的“捆绑”函数，波形图表可同时显示多条曲线。在图 3-30 中，“捆绑”函数将三个 VI 的输出曲线捆绑至一个波形图表。

图 3-30. 连线多条曲线至波形图表



波形图表接线端改变以匹配“捆绑”函数的输出。如要添加更多的曲线，可使用定位工具缩放“捆绑”函数。

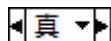
I. 条件结构

条件结构包括两个或两个以上子程序框图（也称“条件分支”）。



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

条件结构顶部的条件选择器标签是由结构中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。

将一个输入值连接至选择器接线端，即可选择需执行的条件分支。



可为选择器接线端连接一个整数、布尔值、字符串和枚举型值。条件选择器接线端可置于条件结构左边框的任意位置。如选择器接线端的数据类型是布尔值型，则该结构包括“真”和“假”分支。如果选择器接线端是一个整数、字符串或枚举型值，则该结构可以包括任意个分支。



注 默认情况下，连接至选择器接线端的字符串区分大小写。如要取消字符大小写区分，将字符串值连接至选择器接线端，右键单击条件结构的边框，然后从快捷菜单中选择**不区分大小写匹配**。

如未指定条件结构默认用于处理范围外数值的条件分支，则必须列出所有可能的输入值。例如，如果分支选择器的数据类型是整型，并且已为 1、2 和 3 指定了相应的分支，则必须指定一个默认分支用于处理当输入数据为 4 或任何有效整数值时的情况。



注 如将一个布尔控件连接至选择器，则不能指定默认分支。如右键单击分支选择器标签，快捷菜单中不出现**本分支设置为默认分支**选项。将布尔控件设置为 TRUE 或 FALSE，确定执行的分支。

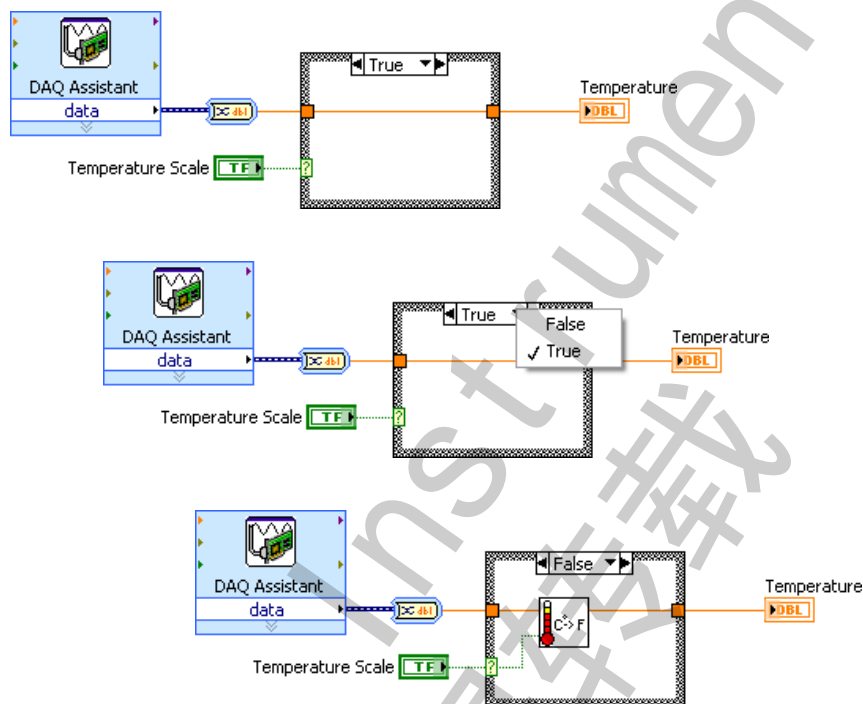
右键单击顺序结构，在快捷菜单中选择**替换为层叠式顺序**，可将条件结构替换为层叠式顺序结构。

右键单击条件结构边框添加、复制、删除、重排及选择默认分支。

分支选择

图 3-31 为根据用户选择 °C 或 °F 作为温度单位，通过条件结构执行不同代码的 VI。最上端的程序框图为顶层“真”条件分支。中间的程序框图为“假”条件分支。如要选择分支，在条件选择器标签内输入值或使用标签工具编辑值。选择其他分支后，分支将出现在程序框图上。如图 3-31 底部程序框图所示。

图 3-31. 更改条件结构的分支显示



如输入的选择器值与连接到选择器接线端的对象不是同一类型，则选择器值为红色。这表示只有编辑或删除该值后 VI 才可运行。同样由于浮点算术运算可能存在四舍五入误差，因此浮点数不能作为条件选择器值。如果将一个浮点数连接到条件分支，LabVIEW 将对其进行舍入到最近整数。如果在分支选择器标签中输入浮点值，数值将变成红色，表示在执行结构前必须删除或编辑该值。

条件结构的输入和输出隧道

可为条件结构创建多个输入输出隧道。所有输入都可供条件分支选用，但条件分支不需使用每个输入。但是，必须为每个条件分支定义各自的输出隧道。

程序框图上的条件结构包括输出隧道，但存在至少一个分支的隧道未连接任何值。如运行该分支，则 LabVIEW 无法判断应输出的值。LabVIEW 以空心隧道表示该错误。程序框图中当前可能没有显示未连线的分支。

如需解决该问题，可切换至包含未连线输出隧道的分支并将输出连接至该隧道。右键单击输出隧道并从快捷菜单中选择**未连线时使用默认**，所有未连线的隧道将使用隧道数据类型的默认值。所有分支的输出均已连线时，输出隧道可显示实心颜色。

避免使用**连接时使用默认**选项。使用该选项将影响程序框图的注释，且可能会使用该代码的其他编程人员造成困扰。**连接时使用默认**也将增加调试代码的难度。使用该选项时，请注意使用的默认值为连线至隧道的数据类型的默认值。例如，如隧道为布尔数据类型，默认值为 `FALSE`。关于数据类型的默认值列表，见表 3-3。

表 3-3. 数据类型的默认值

数据类型	默认值
数值型	0
布尔	<code>FALSE</code>
字符串	空 ("")

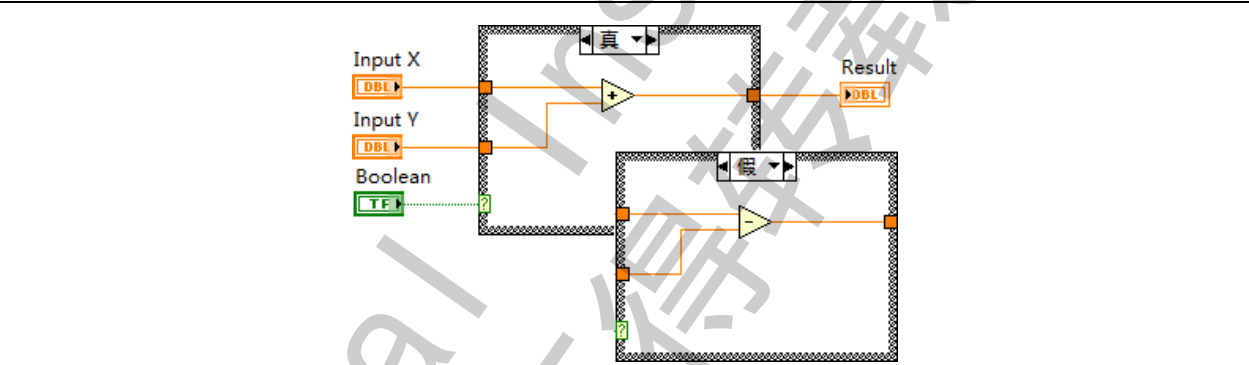
范例

在下列范例中，数值的值通过隧道进入条件结构，根据选择器接线端的连线值执行加或减。

布尔条件结构

图 3-32 为布尔条件结构。分支层叠显示以简化显示。

图 3-32. 布尔条件结构



如连线至选择器接线端的布尔控件为“真”，VI 对数值执行加法。否则，VI 对数值执行减法。

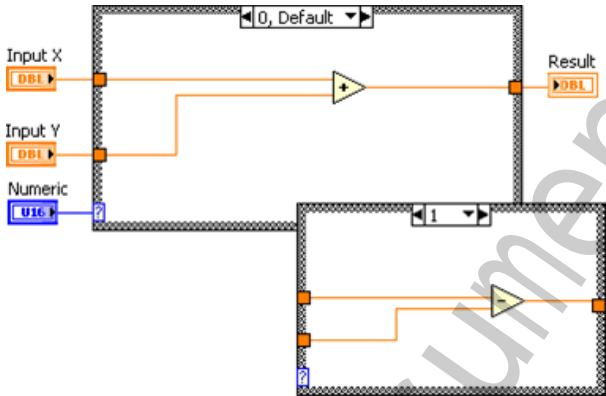
如仅需根据布尔值选择两个值中的一个，可使用“选择”函数替代布尔条件结构。



整数条件结构

图 3-33 为整数条件结构。

图 3-33. 整数条件结构

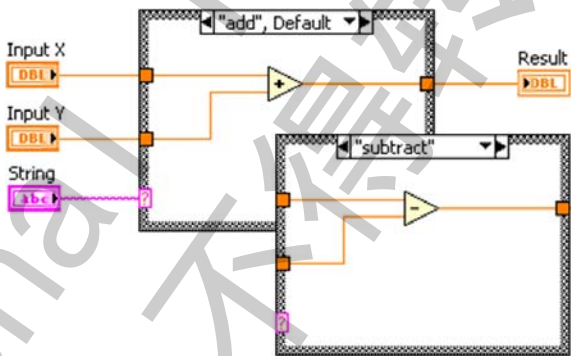


整数为**文本输入控件**选板中的文本下拉列表控件，可显示与文本项相关的数值。如连线至选择器接线端的整数为 0(加)，VI 对数值执行加法。如值为 1（减），VI 对数值执行减法。如整数为 0（加）或 1（减）以外的值，VI 对数值执行加法。因为其为默认分支。

字符串条件结构

图 3-34 为字符串条件结构。

图 3-34. 字符串条件结构

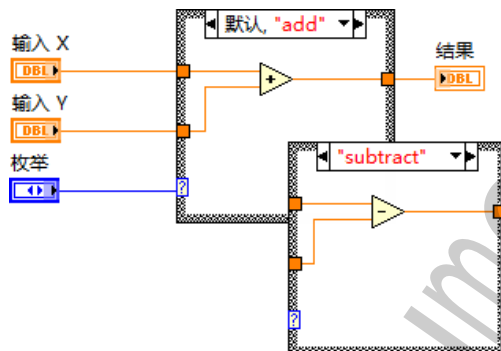


如**字符串**为加，VI 对数值执行加法。如**字符串**为减，VI 对数值执行减法。

枚举型条件结构

图 3-35 为枚举型条件结构。

图 3-35. 枚举型条件结构



枚举控件用于向用户提供一个可供选择的项列表。枚举控件的数据类型包括控件中所有项的数值和字符串标签的相关信息，从条件结构快捷菜单中选择**为每个值添加分支**，分支选择器为枚举控件的每个项显示字符串标签。条件结构根据枚举型控件的当前项，执行相应的分支子框图。在上一个程序框图中，如 **Enum** 为加，VI 对数值执行加法。如 **Enum** 为减，VI 对数值执行减法。

使用条件结构进行错误处理

下列范例为使用错误簇定义分支的条件结构。

图 3-36. 无错误分支

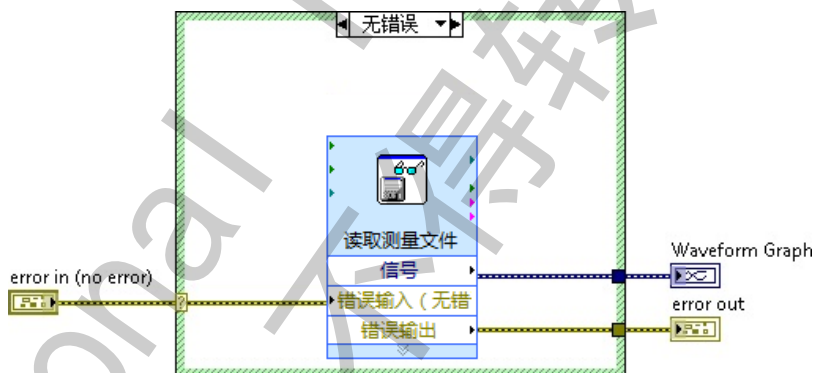
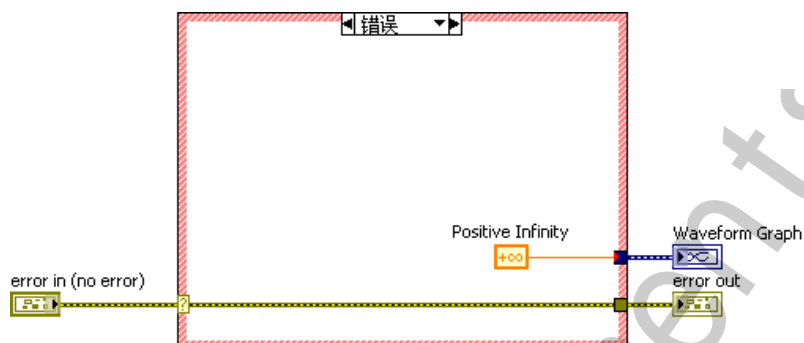


图 3-37. 错误分支



将错误簇连接到“条件结构”的条件选择器接线端时，条件选择器标签将显示两个选项—错误和无错误。错误时边框为红色，无错误时边框为绿色。发生错误时，条件结构将执行错误子程序框图。

连线错误簇至选择器接线端时，条件结构仅识别簇的**状态**布尔值。

International Instruments
不得转载

自测：练习

1. 如函数的输入端标记有一个红点（强制转换点），该点表明下列哪种信息？
 - a. 数据被传输至结构。
 - b. For 循环被配置了条件接线端。
 - c. For 循环计数接线端未连线。
 - d. 传输至节点的值被转换为其他表示法。

2. 哪种结构必须运行至少一次？
 - a. While 循环
 - b. For 循环

3. 下列哪一项 仅位于程序框图中？
 - a. 输入控件
 - b. 常量
 - c. 显示控件
 - d. 连线板

4. 哪一种机械动作会在按下布尔控件时将布尔值由“假”转换为“真”，且松开控件后，直至 LabVIEW 读取该值前均保持“真”值？
 - a. 保持转换直到释放
 - b. 释放时转换
 - c. 保持触发直到释放
 - d. 释放时触发

International Instruments
不得转载

自测：练习答案

1. 如函数的输入端标记有一个红点（强制转换点），该点表明下列哪种信息？
 - a. 数据被传输至结构。
 - b. For 循环被配置了条件接线端。
 - c. For 循环计数接线端未连线。
 - d. **传输至节点的值被转换为其他表示法。**

2. 哪种结构必须运行至少一次？
 - a. **While 循环**
 - b. For 循环

3. 下列哪一项 仅位于程序框图中？
 - a. 输入控件
 - b. **常量**
 - c. 显示控件
 - d. 连线板

4. 哪一种机械动作会在按下布尔控件时将布尔值由“假”转换为“真”，且松开控件后，直至 LabVIEW 读取该值前均保持“真”值？
 - a. 保持转换直到释放
 - b. 释放时转换
 - c. 保持触发直到释放
 - d. **释放时触发**

笔记

ational Instruments
不得转载

开发模块化应用程序

本课程介绍了开发模块化应用程序的方法。LabVIEW 的优势在于 VI 的层次化结构。可将新创建的 VI 用于程序框图上的另一个 VI。层次化结构对层的数量没有限制。模块化编程不仅利于对程序修改进行管理，也便于程序框图的快速调试。

主题

- A. 理解模块化概念
- B. 创建图标和连线板
- C. 使用子 VI

A. 理解模块化概念

模块化就是将程序分为若干区块。这样，对程序某个模块的修改就不会影响到其他模块。LabVIEW 中模块称为子 VI。

当一个 VI 应用在其他 VI 中，则称为子 VI。子 VI 相当于文本编程语言中的子程序。双击子 VI 时，可显示前面板和程序框图，而非用于配置选项的对话框。前面板包含输入控件和显示控件。程序框图包含连线、前面板图标、函数、可能的子 VI 及其他熟悉的 LabVIEW 对象。

前面板和程序框窗口图右上角均显示 VI 的图标。该图标与放置在程序框图上的 VI 的图标相同。

创建 VI 时，有些操作需频繁执行。可考虑创建子 VI 或循环结构来执行重复操作。例如，下图所示的程序框图包含两个完全相同的操作。

图 4-1. 带有两个相同操作的程序框图

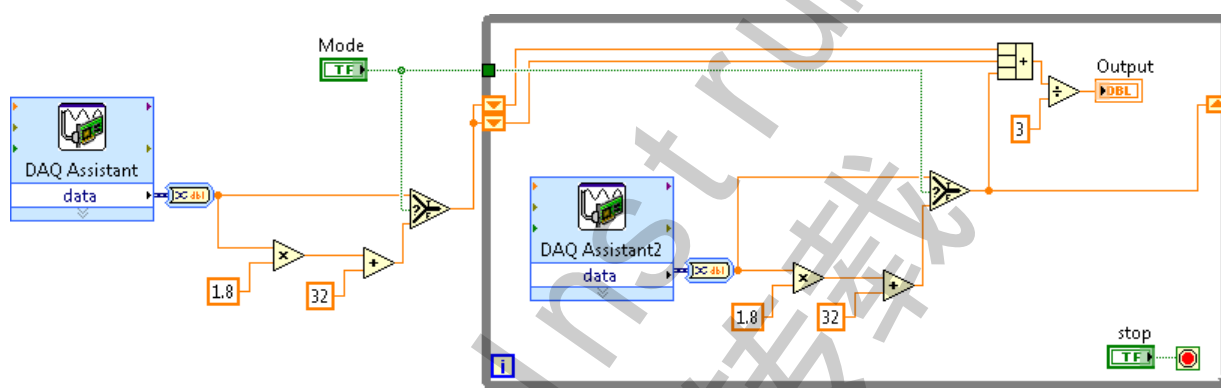
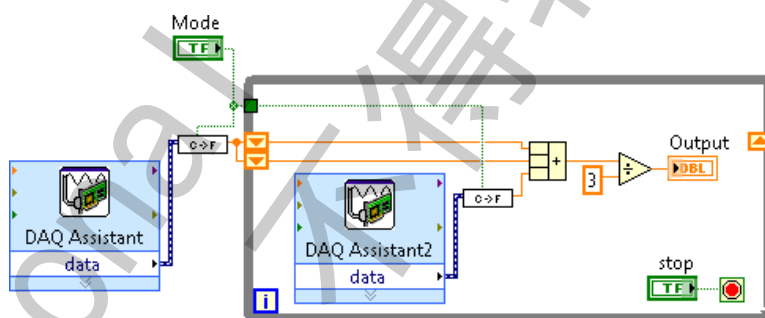
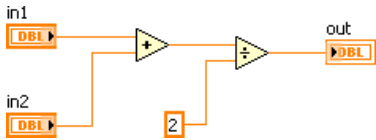
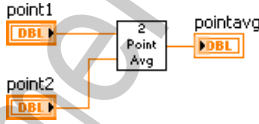


图 4-2. 带有用于执行相同操作的子 VI 的程序框图



该范例调用 Temperature VI 为子 VI，在程序框图上执行两次。子 VI 功能与上一程序框图相同。子 VI 也可重用于其他 VI。

下列伪码和程序框图说明了子 VI 和子程序间的区别。

功能代码	调用程序代码
<pre>function average (in1, in2, out) { out = (in1 + in2)/2.0; }</pre>	<pre>main { average (point1, point2, pointavg) }</pre>
子 VI 程序框图	调用 VI 程序框图
	

B. 创建图标和连线板

创建 VI 的前面板和程序框图后，应创建图标和连线板，以便将该 VI 用作子 VI。图标和连线板相当于文本编程语言中的函数原型。每个 VI 的前面板和程序框图窗口右上角均显示一个图标。



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 用作子 VI，程序框图上将显示代表该子 VI 的图标。如将 VI 添加至选板，VI 图标也将出现在函数选板上。双击前面板或程序框图窗口的图标，可自定义或编辑图标。

 **注** 推荐用户自己定制图标，但这个操作不是必须的。使用默认的 LabVIEW 图标不会影响功能。

如需将 VI 当作子 VI 使用，还需创建连线板。



连线板用于显示 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。连线板标明了可与该 VI 连接的输入和输出端，以便将该 VI 作为子 VI 调用。连线板在其输入端接收数据，然后通过前面板控件将数据传输至程序框图的代码中，从前面板的显示控件中接收运算结果并传递至其输出端。

创建图标

图标为 VI 的图形化表示。

每个 VI 的前面板和程序框图窗口右上角均显示一个图标。



默认 VI 图标中包含的数字表示启动 LabVIEW 后打开新 VI 的个数，最大值为 9。如要取消这些数字，选择**工具 » 选项 » 前面板**，取消勾选**在新建 VI 的图标中使用数字编号 (1-9)** 复选框。

图标中可包含文本或图像。如果将一个 VI 用作子 VI，程序框图上将显示代表该子 VI 的图标。如将 VI 添加至选板，VI 图标也将出现在函数选板上。

使用**图标编辑器**对话框编辑 VI 图标。双击前面板或程序框图右上角的图标，打开**图标编辑器**对话框。

创建图标，以图形化方式表示 VI 或自定义图形化控件。使用**图标编辑器**对话框创建或编辑图标。

如使用特定标识区分相关 VI，建议将特定标识保存为模板。然后可将该模板用于相关 VI 的图标，修改 VI 图标的主体部分，提供 VI 的特定信息。

将特殊标识保存为模板

按照下列步骤，将特殊标识保存为 VI 的图标模板。

1. 双击前面板或程序框图右上角的图标，或右键单击图标并从快捷菜单中选择**编辑图标**，打开**图标编辑器**对话框。
2. 按下 <Ctrl-A> 组合键选中图标的所有用户图层，然后按下 <Delete> 键删除选中的部分。默认图标只有一个用户图层，名为 **VI 图标**。
3. 在**模板**页，从 **VI» 框架** 目录中选择 `_blank.png` 图标。可按类型或关键词浏览模板。
4. 使用**图标编辑器**对话框右边的填充工具，在图标的特殊标识位置填充颜色。
5. 使用文本工具在图标的特殊标识处输入文本。文本处于活动状态时，可使用方向键移动文本。
6. 选择**文件»另存为»模板**，打开**保存图标模板**对话框，将图标另存为模板供日后使用。LabVIEW 将图形模板保存为 256 色 .png 文件。

从模板创建 VI 图标

按照下列步骤，基于模板创建一个 VI 图标。

1. 按下 <Ctrl-A> 组合键选中图标的所有用户图层，然后按下 <Delete> 键删除选中的部分。
2. 在**模板**页上，选择创建的模板。可按类型或关键词浏览模板。
3. 在**图标文本**页，在图标的主体部分输入四行图标文本。可配置文本的字体、对齐方式、大小和颜色。如勾选**文本垂直居中**复选框，**图标编辑器**对话框在图标的主体部分将文本内容居中对齐。
4. 在**符号**页，拖拽符号至**预览**区域。
按下 <F> 键或 <R> 键水平翻转符号或顺时针翻转符号。也可双击符号，将符号放在图标的左上角。可按类型或关键词浏览符号。
5. 使用移动工具移动符号。每个符号都分别位于不同的图层上，分别移动。选中一个符号后，图标的其它部分以灰色显示。用户可确定将要移动的对象。
6. 使用**图标编辑器**对话框右侧的编辑工具，可在需要时编辑图标。

图标编辑器对话框为不连续的编辑工具操作创建新用户图层。选择**图层»创建新图层**，在连续使用编辑工具时创建一个新的用户图层。



注 不能使用**图标编辑器**对话框右侧的编辑工具修改图标模板或图标文本。使用**模板**页和**图标文本**页修改图标模板和图标文本。

7. (可选) 选择**图层»显示图层页**，显示**图层**页。使用图层页配置图标图层的名称、透明度、可见性和顺序。
8. 单击**确定**按钮，将图标信息与 VI 一同保存并关闭**图标编辑器**对话框。
也可从文件系统的任何位置拖动一个图形放置在前面板或程序框图的右上角。可拖放 .png、.bmp 或 .jpg 文件。



注 如通过从文件系统中拖放一个图片修改图标，LabVIEW 将创建一个名为“VI 图标”的用户图层，并从**图标编辑器**对话框中删除其他已有图标信息。

设置连线板

将前面板上的输入控件和显示控件分配至连线板的每个接线端，从而定义连接。连线板位于前面板窗口右上角的 VI 图标左侧。LabVIEW 打开时显示为默认的连线板模式。

连线板上的每个方格都代表一个接线端。使用各个单元格分配输入和输出。默认连线板模式为 $4 \times 2 \times 2 \times 4$ 。右键单击连线板，从快捷菜单中选择模式，可选择其他连线板模式。多余的接线端可以保留，当需要为 VI 添加新的输入或输出端时再进行连接。

图 4-3 中的连线板带有 4 个输入控件和 1 个显示控件。因此 LabVIEW 连线板显示为 4 个输入接线端和 1 个输出接线端。

图 4-3. Slope VI 前面板



选择和更改接线端模式

右键单击连线板并从快捷菜单中选择**模式**，可为 VI 选择不同的接线端模式。例如，可选择有一个有附加接线端的连线板模式。空置出附加接线端，需要时再进行连接。接线板应有一定的灵活性，使改变对 VI 层次结构的影响减到最小。

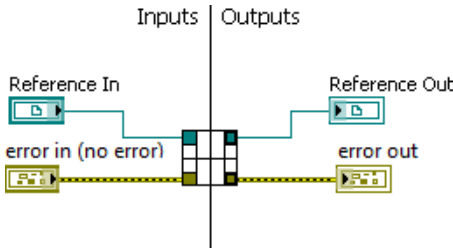
前面板输入控件和显示控件数量可多于接线端。连线板中最多可设置 28 个接线端。

最常见的连线板模式如下所示。该模式有助于简化连线的标准。



图 4-4 为用于接线端模式的标准布局的范例。顶部的输入和输出端通常用于传输引用，底部的输入和输出通常用于错误处理。

图 4-4. 范例接线端模式布局





注 连线板的接线端最好不要超过 16 个接线端。接线端较多的连线板可能用途很多，但是连线困难。如要传递很多数据，请使用簇。

指定接线端的输入控件和显示控件

在为连线板选择模式后，必须为连线板的各个接线端指定一个前面板控件。将输入控件或显示控件连接到连线板时，应把输入端放在左边，输出端放在右边，以免出现复杂或难以理解的连线模式。

如要为接线端指定前面板控件，单击连线板的接线端，然后单击要分配给该接线端的前面板输入控件或显示控件。单击前面板窗口的空白处。接线端的颜色变为相应控件的数据类型颜色，显示接线端已连接。

也可先选取控件，再选择接线端。



注 虽然为连线板上的接线端指定前面板输入控件和显示控件时使用的是连线工具，但连线板和控件之间实际并无连线存在。

C. 使用子 VI

如要在程序框图上放置子 VI，单击**函数**选板上的**选择 VI** 按钮。找到要用作子 VI 的 VI，双击该 VI 以将其放置在程序框图上。

也可将一个打开的 VI 放置在另一打开的 VI 的程序框图上。使用定位工具单击要用作子 VI 的 VI 的前面板或程序框图右上角，将该 VI 图标拖放到其他 VI 的程序框图中。

打开和编辑子 VI

如要在调用 VI 中显示子 VI 的前面板，可使用“操作”工具或“定位”工具双击程序框图的子 VI。如要在调用 VI 中显示子 VI 的前面板，按下 <Ctrl> 键并使用“操作”工具或“定位”工具双击程序框图的子 VI。

使用“操作”或“定位”工具双击程序框图上的子 VI，即可编辑该子 VI。保存子 VI 时，子 VI 的改动将影响到所有调用该子 VI 的程序，而不只是当前程序。

设置必需、推荐和可选输入和输出

在**即时帮助**窗口，必需接线端的标签显示为粗体，推荐接线端显示为纯文本，可选接线端显示为灰色。如在**即时帮助**窗口单击**隐藏可选接线端和完整路径**，则不会出现可选接线端的标签。



输入和输出可设置为必需、推荐或可选，避免用户忘记连接子 VI 接线端。

右键单击连线板中的某个接线端，从快捷菜单中选择**接线端类型**，勾选符号表明接线端的当前设置。有**必需**、**推荐**和**可选**三种类型供选择。也可选择**工具 » 选项 » 前面板**，勾选**连线板接线端默认为必需**复选框。该选项将连线板上的接线端从“推荐”改为“必需”。适用于使用连线工具创建的连接，以及至通过**创建子 VI** 创建的子 VI 的连接。



注 对于动态分配成员 VI，可选择**动态分配输入（必需）**或**动态分配输出（推荐）**。

如未连接“必需”输入端，该子 VI 所在的程序框图将出现断线。输出接线端不存在“必需”选项。如未连接“推荐”或“可选”输入或输出接线端，该子 VI 所在的程序框图仍然可以执行。上述接线端未连线时，VI 不会产生任何警告。

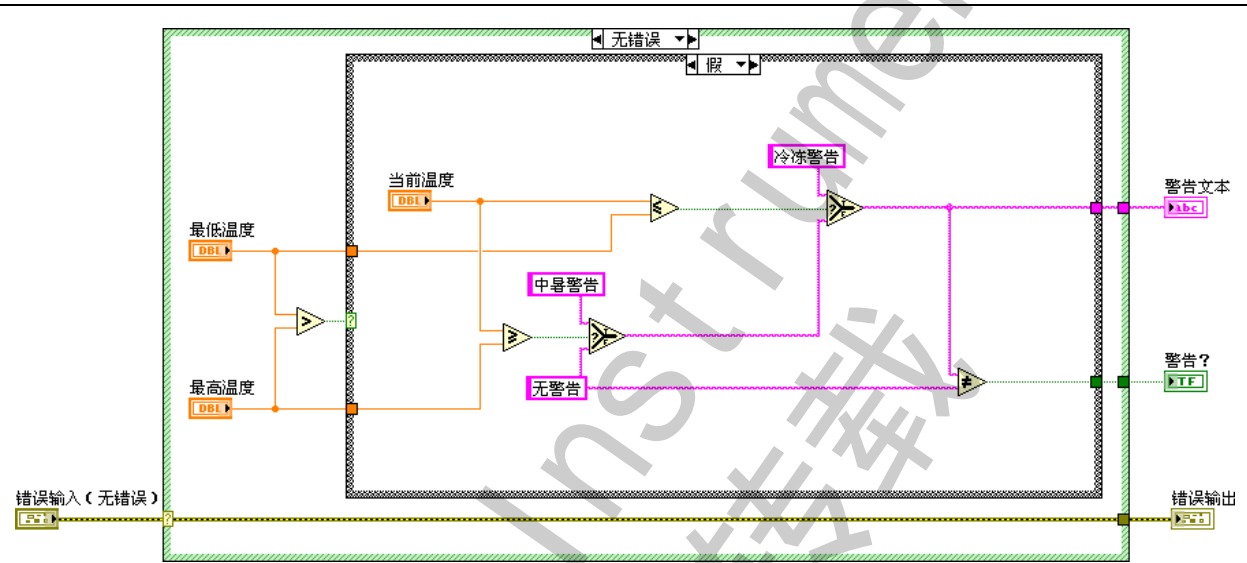
vi.lib 中的 VI 输入输出接线端已被标识为**必需**、**推荐**或**可选**。LabVIEW 将所创建的 VI 的输入和输出默认设置为**推荐**。如某个输入或输出对 VI 的正常运行是必不可少的，该接线端应设置为“必需”。

在子 VI 中处理错误

通过错误簇在子 VI 中传入和传出错误。通过条件结构，通过“无错误”和“错误”条件分支处理传输至子 VI 的错误。

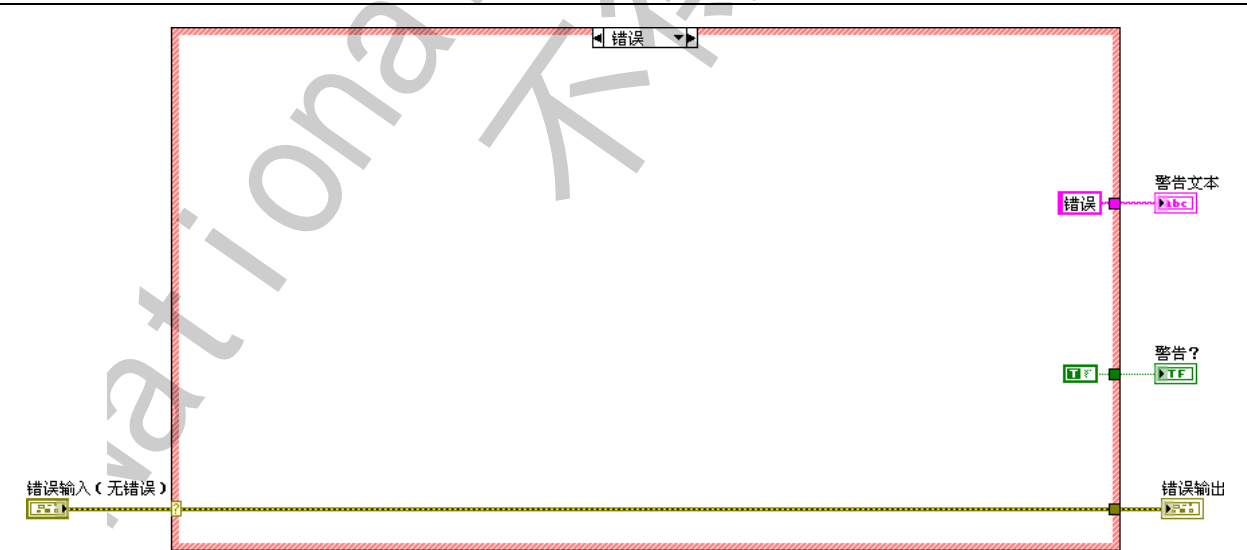
“无错误”分支包含子 VI 的常见操作代码，如图 4-5 所示。

图 4-5. 范例子 VI 的无错误分支



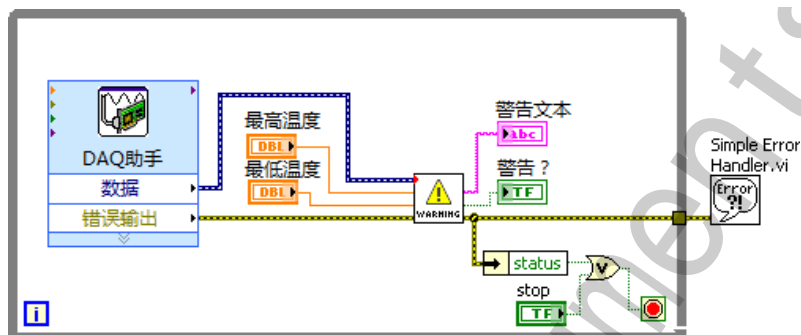
“错误”分支通常经过**错误输入**簇输入控件传输错误至**错误输出**簇显示控件，且包含处理错误的必需代码。如图 4-6 所示。

图 4-6. 范例子 VI 的错误分支



在子 VI 内部应避免使用“简易错误处理器”VI 和“通用错误处理器”VI。如有需要，请在调用 VI 中使用上述 VI。如图 4-7 所示。

图 4-7. 调用 VI 的程序框图

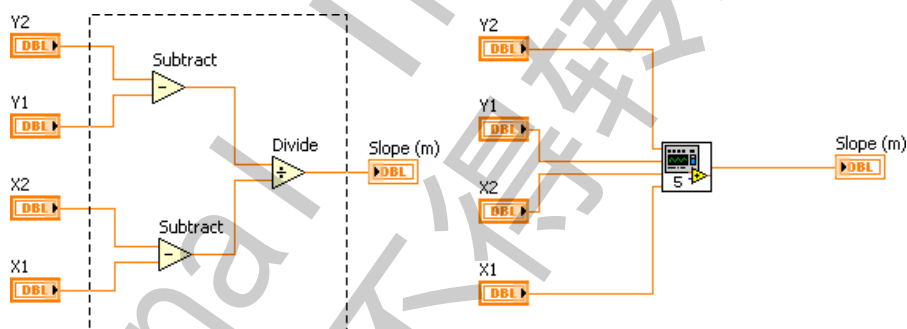


基于已有 VI 创建子 VI

将程序框图上的某部分转换为子 VI 可简化程序框图。使用定位工具选择需重复使用的部分程序框图，选择**编辑**»**创建子 VI**，可将部分 VI 转换成子 VI。出现一个新子 VI 的图标替换这部分被选中的程序框图。LabVIEW 可为新的子 VI 创建输入控件和显示控件，并根据所选控件的数目自动配置连线板，将子 VI 与现有的连线对接。

图 4-8 为转换某部分 VI 为子 VI 的方法。

图 4-8. 新建子 VI



新建的子 VI 使用连线板的默认模式及默认图标。双击子 VI 可编辑连线板和图标并保存子 VI。



注 创建子 VI 时选中的对象不得多于 28 个，因为连线板最多可连接 28 个对象。如前面板上的控件不止 28 个，可将其中的一些对象组合为一个簇，然后将该簇分配至连线板上的一个接线端。

自测：练习

1. 在子 VI 中，未完成下列哪些接线端连线将导致错误？
 - a. 必需
 - b. 推荐
 - c. 可选

2. VI 被用作子 VI 时，必须先创建自定义图标。
 - a. 对
 - b. 错

International Instruments
不得转载

自测：练习答案

1. 在子 VI 中，未完成下列哪些接线端连线将导致错误？

- a. **必需**
- b. 推荐
- c. 可选

2. VI 被用作子 VI 时，必须先创建自定义图标。

- a. 对
- b. **错**

将 VI 用作子 VI 时，无需先创建自定义图标。但最好创建自定义图标，以增强代码的可读性。

笔记

ational Instruments
不得转载

创建和使用数据结构

在某些情况下，将关联性数据组合在一起是有益处的。使用数组和簇在 LabVIEW 中组合关联性数据。数据将同一数据类型的数据组合至一个数据结构，簇将多个不同的数据类型组合至一个数据结构。使用自定义类型定义自定义数组和簇。本课程将介绍数据、簇和自定义类型，及使用这些类型有益于应用程序的情况。

主题

- A. 数组
- B. 常见数组函数
- C. 多态性
- D. 自动索引
- E. 簇
- F. 自定义类型

A. 数组

数组由元素和维度组成。元素是组成数组的数据。维度是数组的长度、高度或深度。数组可以是一维或多维的，在内存允许的情况下每一维度可有多达 $(2^{31}) - 1$ 个元素。

可以创建数值、布尔、路径、字符串、波形和簇等数据类型的数组。对一组相似的数据进行操作并重复计算时，可考虑使用数组。数组最适于存储从波形采集而来的数据或循环中生成的数据（每次循环生成数组中的一个元素）。



注 LabVIEW 中的数组索引从零开始。无论具有几个维度的数组，其索引的第一个元素均为零。

限制

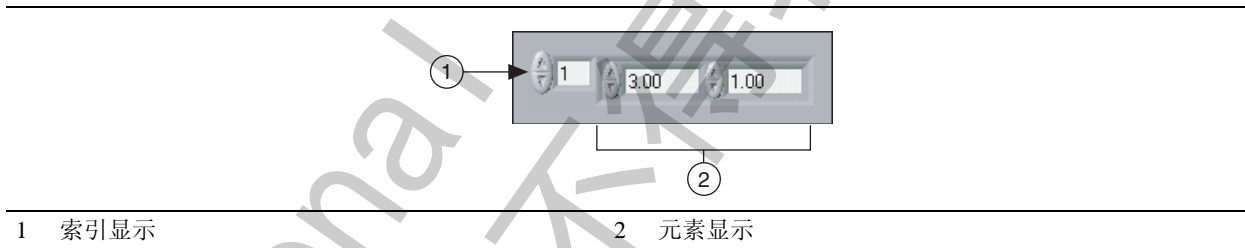
数组中不能再创建数组。但可以创建多维数组或创建一个簇数组，其中每个簇均含有一个或多个数组。不能创建元素为子面板控件、选项卡控件、.NET 控件、ActiveX 控件、图表或多曲线 XY 图的数组。关于簇的更多信息见本课程的相关章节。

包含 12 个月份的文本数组是一个简单的数组的范例。LabVIEW 将其表示为 12 个元素的一维字符串数组。

数组元素是有序的。数组通过索引访问数组中任意一个特定的元素。索引以零开始，即索引的范围是 0 到 $n - 1$ ，其中 n 是数组中元素的个数。例如，每年 12 个月份中的 n 等于 12，因此索引范围是 0 至 11。三月份是一年中的第三个月，其索引为 2。

图 5-1 为数值型数组的范例。数组中的第一个元素 (3.00) 的索引为 1，第二个元素 (1.00) 的索引为 2。图中未显示索引为 0 的元素，因为在索引显示中选择了元素 1。索引显示框中的元素总是对应元素显示框左上角的元素。

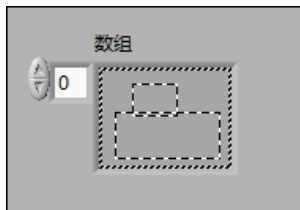
图 5-1. 数值数组控件



创建数组控件

通过以下方式可在前面板上创建一个数组输入控件或数组显示控件：在前面板上放置一个数组外框，然后将一个数据对象或元素拖曳到该数组外框中。数据对象或元素可以是数值、布尔、字符串、路径、引用句柄、簇输入控件或显示控件。

图 5-2. 在数组外框内放置数值控件



如试图在数组外框中放入一个无效的输入控件或显示控件，该操作将无法实现。

在程序框图上使用数组前，必须在数组外框中插入对象。否则，数组接线端显示为黑色且不具有相关的数据类型。

创建数组常量

如需在程序框图中创建数组常量，则先从函数选板中选择数组常量，将数组外框放置于程序框图上，然后将字符串常量、数值常量、布尔常量或簇常量放入数组外框。数组常量可存储常量数据或同另一个数组进行比较。

二维数组

二维数组元素存储在网格中。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，并且这两个索引都从零开始。图 5-3 显示了一个 8 列 8 行的二维数组，其中包含 $8 \times 8 = 64$ 个元素。

图 5-3. 二维数组

		列索引							
		0	1	2	3	4	5	6	7
行索引	0								
	1								
	2								
	3								
	4								
	5								
	6								
	7								

如需在前面板上添加一个多维数组控件，则右键单击索引框并从快捷菜单中选择**添加维度**。用户也可以直接拖拽索引显示边框至所需维数。

初始化数组

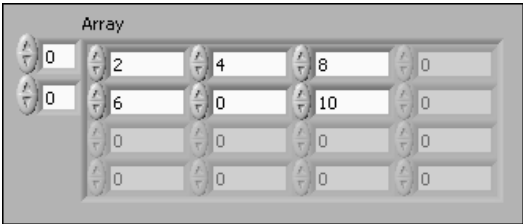
用户可对数组执行初始化操作或保留原样。初始化数组时需定义每个维数的元素个数及元素内容未初始化的数组具有固定大小的维度，但不包含元素。图 5-4 为未初始化的二维数组控件。注意，元素显示为灰色。这表明数组未初始化。

图 5-4. 二维未初始化数组



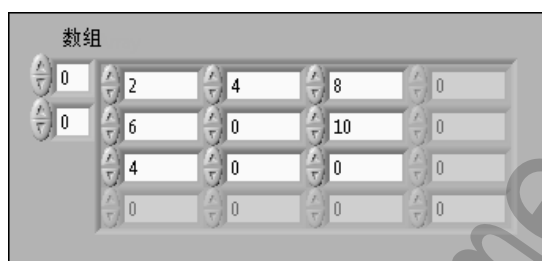
在图 5-5 中初始化了 6 个元素。

图 5-5. 带有 6 个初始化元素的二维数组



在二维数组中，如某行中的某元素被初始化，该行中的其他元素将被初始化，并被填充为同一数据类型的默认值。例如，在图 5-6 中的第一列第三行输入 4，第三行的第二列和第三列中的元素将被自动填充为 0。

图 5-6. 自动填充零的数组



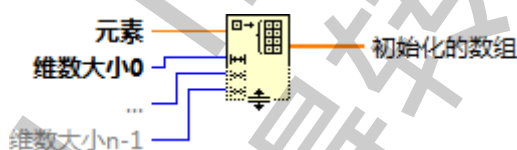
B. 常见数组函数

LabVIEW 将函数组合在一起，用户可使用这些函数对**数组**选板中的数组进行操作。下列函数为执行数组操作的最常见函数。

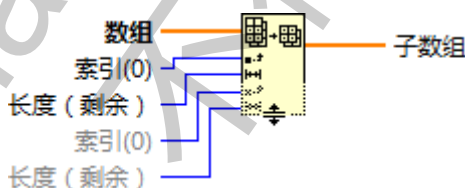
- 数组大小—返回**数组**每个维度中元素的个数。连线板可显示该多态函数的默认数据类型。



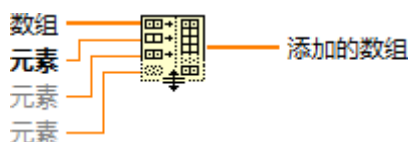
- 初始化数组—创建 n 维数组，每个元素都初始化为**元素**的值。通过定位工具可调整函数的大小，增加输出数组的维数（元素、行、列或页等）。连线板可显示该多态函数的默认数据类型。



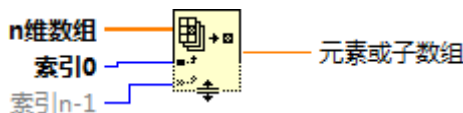
- 数组子集—返回**数组**的一部分，从**索引**处开始，包含**长度**个元素。



- 创建数组—连接多个数组或向 N 维数组添加元素。也可使用替换数组子集函数，修改现有数组。连线板可显示该多态函数的默认数据类型。



- 索引数组一返回 **n 维数组** 在 **索引** 位置的 **元素或子数组**。连线数组到该函数时，函数可自动调整大小，在 **n 维数组** 中显示各个维度的 **索引** 输入。也可通过调整函数大小，添加 **元素或子数组** 接线端。连线板可显示该多态函数的默认数据类型。



C. 多态性

多态是指 VI 和函数能够自动适应不同类型输入数据。函数多态的程度各不相同，可以是全部或部分输入多态，也可以是完全没有多态输入。有的函数输入可接收数值或布尔值，有的函数输入可接收数值或字符串，有的函数输入不仅接收数值标量还接收数值数组，数值簇或数值簇构成的数组等数据，还有的函数输入仅仅接收一维数组，数组的元素可以是任意数据类型。另外，有的函数输入可接收所有数据类型，包括复数值。

算术函数的多态性

算术函数的输入都是数值型数据。除了函数说明中所指明的一些特例以外，默认的输出数据通常和输入数据保持相同的数值表示方法，如果输入数据包含多种不同的数值表示方法，那么默认输出数据的类型是输入数据的类型中较大的那种类型。例如，如将一个 8 位整数和一个 16 位整数相加，则默认的输出将是一个 16 位整数。如配置数值函数的输出，则指定的设置将覆盖原有的默认设置。

算术函数是对数值、数值数组、数值簇、数值簇构成的数组，以及复数等数据对象的操作。对以上函数允许的输入类型进行归纳，得到以下定义：

数值型 = 数值标量 OR 数组 [数值型] OR 簇 [各种数值型]

数值标量可以是浮点型数字、整型数字或实部和虚部都为浮点数的复数。LabVIEW 中，元素为数组的数组是非法的。

数组的维数和大小是任意的。簇中元素的数量也是任意的。函数输出和输入的数值表示法一致。对于只有一个输入端的函数，函数将处理数组或簇中的每一个元素。

对于有两个输入的函数，用户可以使用如下方式组合：

- 两个结构输入类似**—当两个输入结构相同时，输出的结构与输入相同。
- 两个输入中有一个标量**—当两个输入中有一个数值标量，而另一个是数组或簇时，输出为数组或簇。
- 两个输入指定了某种类型的数组**—当两个输入中有一个数值数组，另一个是数值类型时，输出为数组。

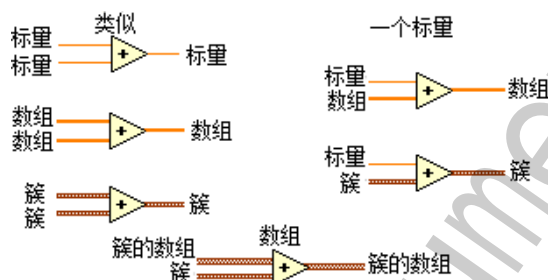
对于两个输入结构类似的情况，LabVIEW 将处理两个输入结构中的每一个元素。例如，将两个数组中的元素一一相加。此时，必须保证两个数组维数相同。也可以再对这两个维数相同的数组添加不同个数的元素；两个维数不同的数组作为输入相加时，输出的结果数组的维数和输入数组中维数较小的一致。两个簇相加的时候，必须拥有相同的元素个数，并且每对相应元素的类型必须相同。

对于两个输入包含一个标量和一个数组（或簇）的情况，LabVIEW 的函数将处理输入标量和输入数组（或簇）中的每一个元素。例如，LabVIEW 可以将数组中的每个元素减去一个特定的数，无论数组的维数有多大。

对于两个输入中一个是数值类型，另一个是这种指定类型元素构成的数组的情况，LabVIEW 函数将处理指定数组的每个元素。例如，图形由点组成的数组，每个点为两个数值类型（x 和 y）的簇。如要将图形在 x 方向偏移 5 个单位，在 y 方向偏移 8 个单位，可添加点 (5, 8) 至图形。

图 5-7 列举了加法函数可能出现的多态组合。

图 5-7. 加法函数可能出现的多态组合



布尔函数的多态性

逻辑函数的输入可以是布尔值、数值和错误簇。如果输入是数值型，那么 LabVIEW 将对输入数据进行位运算操作。如果输入是整型，那么输出数据是和输入相同表示的整型。如果输入是浮点型，那么 LabVIEW 会将它舍入到一个 32 位整型数字，而输出结果也将是 32 位整型。如输入是一个错误簇，LabVIEW 只传递错误簇中状态的 TRUE 或 FALSE 值至输入端。

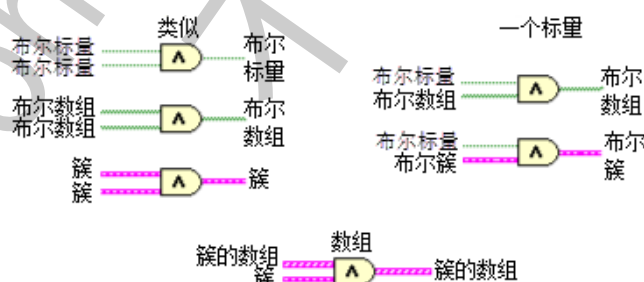
逻辑函数可以处理数值或布尔型的数组、数值或布尔型的簇、数值簇或布尔簇构成的数组等类型的数据。

允许的输入类型的正式的递归性定义如下（复数和数组型数组不被允许）：

逻辑型 = 布尔标量 OR 数值标量 OR 数组 [逻辑型] OR 簇 [多个逻辑型]

如果一个逻辑函数有两个输入，那么可以用和算术函数相同的方式组合这两个输入。但是，逻辑函数还受到一个更为严格的限制：只能对两个布尔值或两个数值进行基本操作。例如，不能在布尔值和数值之间进行“与 (AND)”运算。图 5-8 列举了“与”函数中两个布尔值输入的几种组合方式。

图 5-8. “与”函数中布尔值输入的几种组合方式



D. 自动索引

如将数组连接至 For 循环或 While 循环，启用自动索引功能可有序对应循环与数组中元素。如已使用自动索引功能，隧道图标将由方形变为自动索引的图标。右键单击隧道，从快捷菜单中选择**启用索引**或**禁用索引**切换隧道的状态。

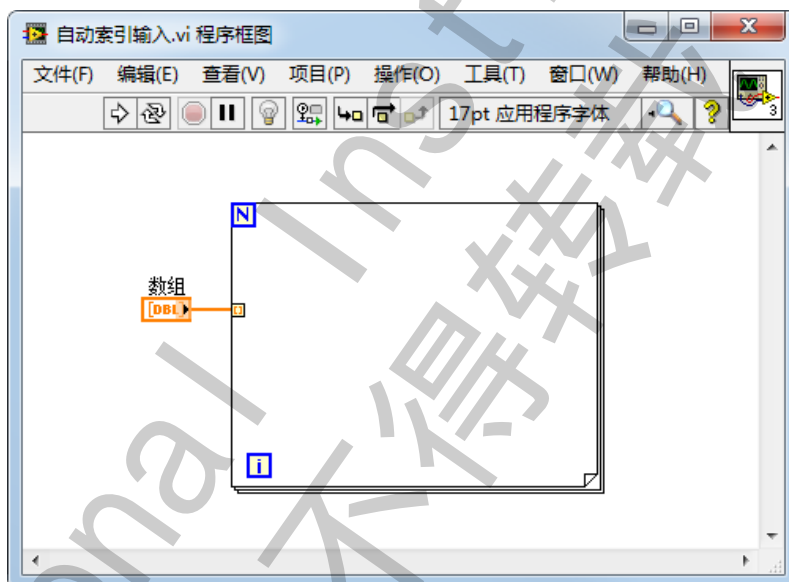


数组输入和自动索引

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成与数组大小一致，因此用户无需为总数接线端连接数值。For 循环每次处理数组中的一个元素，所以，自动索引相当于给 For 循环的总数接线端连接了一个数组大小的值。如不需要每次处理数组中的一个元素，可以禁用自动索引。

在图 5-9 中，For 循环执行次数等于数组中元素个数。通常，For 循环的总数接线端未连线，运行箭头将显示为断开。但此时运行箭头未显示为断开。

图 5-9. 设置 For 循环总数的数组



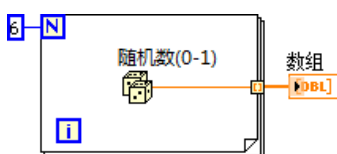
如果有多个隧道启用自动索引，或对计数接线端进行连线，实际的循环次数将取其中较小的值。例如，如两个数组进入 For 循环，分别含有 10 个和 20 个元素，同时将值 15 连接到总数接线端，这时该循环仍将只执行 10 次，For 循环索引第一个数组的所有元素，索引第二个数组中的前 10 个元素。

数组输出和自动索引

启用数组输出隧道的自动索引功能时，该输出数组从每次循环中接收一个新元素。因此，自动索引的输出数组的大小等于循环的次数。

隧道输出至数组显示控件的连线在循环边框将变粗，且输出隧道将包含表示数组的方框。如图 5-10 所示。

图 5-10. 自动索引输出



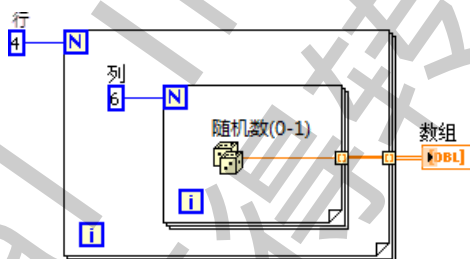
右键单击循环边框上的隧道，从快捷菜单中选择**启用索引**或**禁用索引**，可启用或禁用自动索引。While 循环默认为禁用自动索引。

例如，如仅需传出隧道的最后一个值，可禁用自动索引

创建二维数组

将 2 个 For 循环嵌套在一起创建 2 维数组。外层的 For 循环创建行元素，内层的 For 循环创建列元素。如图 5-11 所示。

图 5-11. 创建二维数组

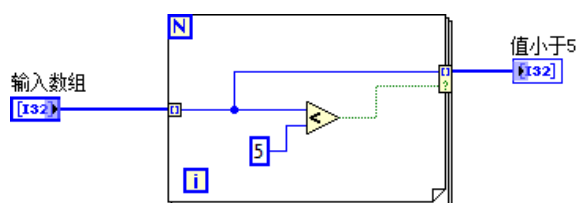


带有条件隧道的自动索引

右键单击循环的输出隧道，从快捷菜单中选择**隧道模式 » 条件**，可使 LabVIEW 根据条件将循环的输出值写入输出隧道。

在图 5-12 中，数组**输入数组**包含下列元素：7、2、0、3、1、9、5 和 7。由于条件隧道，循环完成全部计数后，**值小于 5**的数组仅包含 2、0、3 和 1 元素。

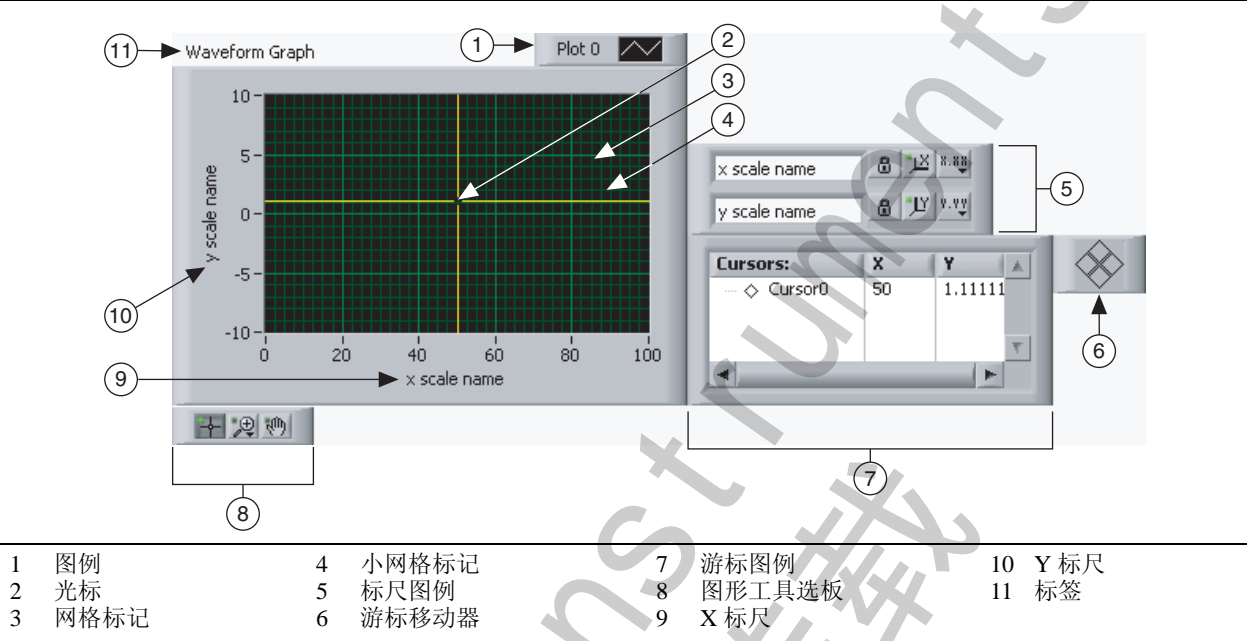
图 5-12. 带有条件隧道的自动索引 For 循环



波形图

含有图形的 VI 通常先将数据采集到数组中，再将数据绘制到图形中。图 5-13 显示了一个图形的元素。

图 5-13. 波形图



“图形显示控件”选板中的图形包括波形图和 XY 图。波形图仅绘制单值函数 $y = f(x)$ ，各点沿 x 轴均匀分布。例如一个随时间变化的波形。XY 图可显示任何均匀采样或非均匀采样的点的集合。

改变图例的大小可显示多条曲线。使用多条曲线可节省前面板空间及进行曲线间的比较。XY 和波形图均自动适应多曲线。

单曲线波形图

波形图接收多种数据类型以显示单条曲线。对于一个数值数组，其中每个数据被视为图形中的点，从 $x=0$ 开始以 1 为增量递增 x 索引。波形图接受包含初始 x 值、 Δx 及 y 数据数组的簇。波形图也接收波形数据类型，该类型包含了波形的数据、起始时间和时间间隔 (Δt)。

关于波形图所接收的数据类型，见 labview\examples\general\graphs\gengraph.llb 中的 Waveform Graph VI 范例。

多曲线波形图

波形图接收多种数据类型以显示多条曲线。波形图接收二维数值数组，数组中的一行即一条曲线。波形图将数组中的数据视为图形上的点，从 $x=0$ 开始以 1 为增量递增 x 索引。将一个二维数组数据类型连接到波形图上，右键单击波形图并从快捷菜单中选择**转置数组**，则数组中的每一列便作为一条曲线显示。多曲线波形图尤其适用于 DAQ 设备的多通道数据采集。DAQ 设备以二维数组的形式返回数据，数组中的一列即代表一路通道的数据。

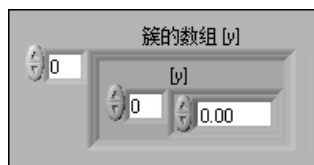
关于接收该数据类型的图形范例见 labview\examples\general\graphs\gengraph.llb 中 Waveform Graph VI 的 (Y)Multi Plot 1 图形。

波形图还接收包含了初始 x 值、 Δx 和 y 二维数组的簇。波形图将 y 数据作为图形上的点，从 x 初始值开始以 Δx 为增量递增 x 索引。该数据类型适用于显示以相同速率采样的多个信号。关于接收该数

据类型的图形范例见 labview\examples\general\graphs\gengraph.11b 中 Waveform Graph VI 的 (Xo = 10, dX = 2, Y) Multi Plot 2 图形。

波形图接收包含簇的曲线数组。每个簇包含一个包含 y 数据的一维数组。内部数组描述了曲线上的各点，外部数组的每个簇对应一条曲线。图 5-14 的前面板显示了这样的 y 簇的数组。

图 5-14. y 簇的数组



如每条曲线所含的元素个数都不同，应使用曲线数组而不要使用二维数组。例如，从几个通道采集数据且每个通道的采集时间都不同时，应使用曲线数组而不是二维数组，因为二维数组每一行中元素的个数必须相同。簇数组内部数组的元素个数可各不相同。关于接收该数据类型的图形范例见 labview\examples\general\graphs\gengraph.11b 中 Waveform Graph VI 的 (Y)Multi Plot 2 图形。

波形图接收一个簇，簇中有初始值 x 、 Δx 和簇数组。每个簇包含由 y 数据组成的一维数组。捆绑函数可将数组捆绑至簇，或用创建数组函数将簇嵌入数组。关于簇和捆绑函数的详细信息，见“簇”。

创建簇数组函数可创建一个包含指定输入内容的簇数组。关于接收该数据类型的图形范例见 labview\examples\general\graphs\gengraph.11b 中 Waveform Graph VI 的 (Xo = 10, dX = 2, Y) Multi Plot 3 图形。

波形图接收包含了 x 值、 Δx 值和 y 数据数组的簇数组。这种数据类型为多曲线波形图所常用，可指定唯一的起始点和每条曲线的 x 标尺增量。关于接收该数据类型的图形范例见 labview\examples\general\graphs\gengraph.11b 中 Waveform Graph VI 的 (Xo = 10, dX = 2, Y) Multi Plot 1 图形。

波形图还接收动态数据类型，用于 Express VI。动态数据类型除包括对应于信号的数据外，还包括信号信息的各种属性，如信号名称、数据采集日期和时间等。属性决定了信号在波形图中的显示方式。当动态数据类型包含多个通道时，波形图可显示每个通道的曲线并自动格式化图例以及图形 x 标尺的时间标识。

单曲线 XY 图

XY 图接收三种数据类型以显示单曲线 XY 图。XY 图接收包含 x 数组和 y 数组的簇。关于接收该数据类型的图形范例见 labview\examples\general\graphs\gengraph.11b 中 XY Graph VI 的 (X and Y arrays) Single Plot 图形。

XY 图接收点数组，其中每个点是包含 x 值和 y 值的一个簇。关于接收该数据类型的图形范例见 labview\examples\general\graphs\gengraph.11b 中 XY Graph VI 的 (Array of Pts) Single Plot 图形。XY 图形接收复数数组，其中 X 轴和 Y 轴分别显示实部和虚部。

关于簇的详细信息见“簇”。

多曲线 XY 图

XY 图接收三种数据类型以显示多条曲线。XY 图接收曲线数组，其中每条曲线是包含 x 数组和 y 数组的一个簇。关于接收该数据类型的图形范例见 labview\examples\general\graphs\gengraph.11b 中 XY Graph VI 的 (X and Y arrays) Multi Plot 图形。

XY 图接收曲线簇数组，其中每条曲线为一个点数组。每一个点是包含 x 值和 y 值的一个簇。关于接收该数据类型的图形范例见 labview\examples\general\graphs\gengraph.11b 中 XY Graph VI 的

(Array of Pts) Multi Plot 图形。XY 图也接收曲线簇数组，其中每条曲线是一个复数数组，X 轴和 Y 轴分别显示复数的实部和虚部。

关于簇的详细信息见“簇”。

E. 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。

将几个数据元素捆绑成簇可消除程序框图上的混乱连线，减少子 VI 所需的连线板接线端的数目。连线板最多可拥有 28 个接线端。如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，则应将其中的一些对象组成一个簇，然后为该簇分配一个连线板接线端。

程序框图上的绝大多数簇的连线样式和数据类型接线端为粉红色。错误簇的连线样式和数据类型终端显示为深黄色。由数值控件组成的簇，有时也称为点，其连线样式和数据类型接线端为褐色。褐色的数值簇可连接到数值函数（例如，加或平方根函数），以便对簇中的所有元素同时进行同样运算。

簇元素顺序

簇和数组元素都是有序的，必须使用“解除捆绑”函数一次取消捆绑所有元素。也可使用“按名称解除捆绑”函数按名称解除捆绑簇元素。如使用“按名称解除捆绑”函数，则每个簇元素都必须带有标签。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，簇包含输入控件或显示控件。但簇不能同时包含输入控件和显示控件。

创建簇控件

在前面板上通过以下方式创建一个簇输入控件或簇显示控件：在前面板上添加一个簇外框，再将一个数据对象或元素拖曳到簇外框中，数据对象或元素可以是数值、布尔、字符串、路径、引用句柄、簇输入控件或簇显示控件。

放置簇外框时，拖拽光标调整簇的大小。

图 5-15. 创建簇控件

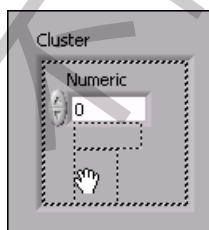
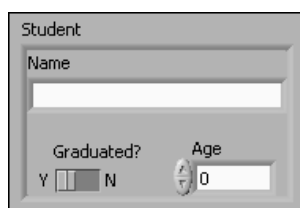


图 5-16 为包含 3 个控件的簇的范例：字符串、布尔开关和数值。与数组类似，簇只能包含输入控件或显示控件，不能同时包含两种控件。

图 5-16. 簇控件范例



创建簇常量

如需在程序框图中创建一个簇常量，则从**函数**选板中选择一个簇常量，将该簇外框放置于程序框图上，再将字符串常量、数值常量、布尔常量或簇常量放置到该簇外框中。簇常量用于存储常量数据或同另一个簇进行比较。

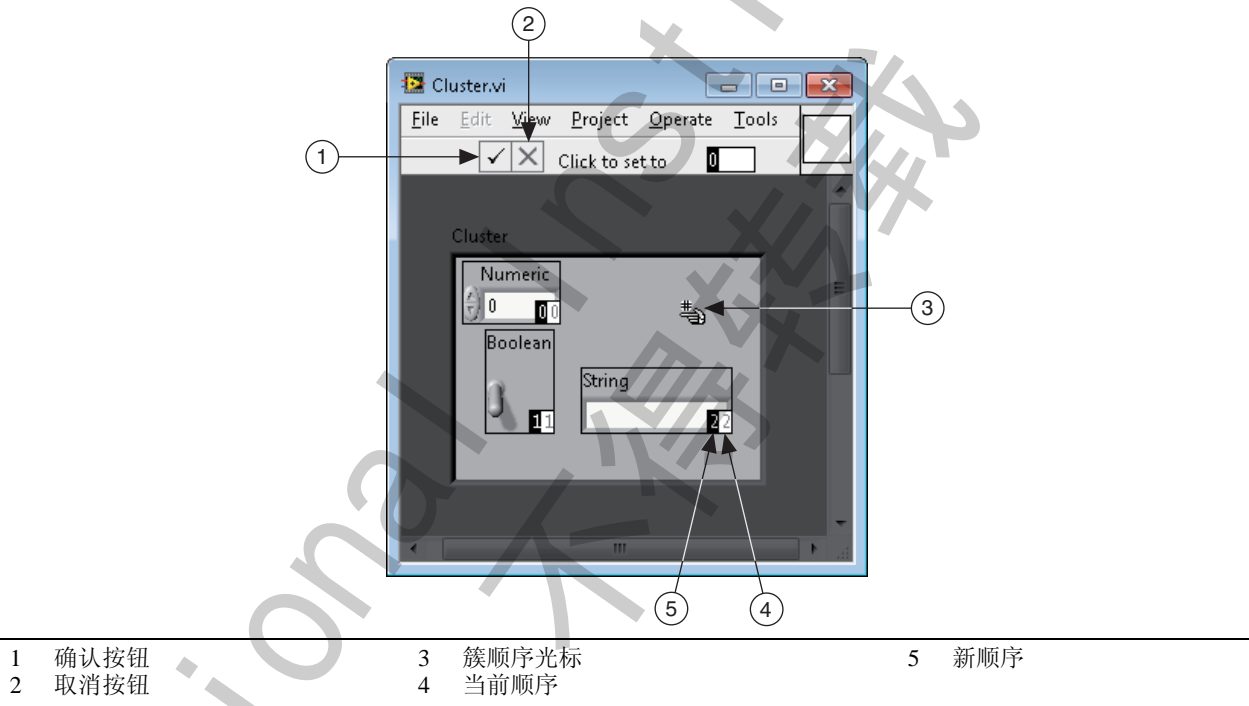
如前面板窗口已存在簇控件，在程序框图上要创建包含相同元素的簇常量，可从前面板拖拽簇至程序框图或在程序框图上右键单击簇，从快捷菜单中选择**创建 » 常量**。

簇顺序

簇元素有自己的逻辑顺序，与它们在簇外框中的位置无关。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。如删除某个元素，顺序会自动调整。簇顺序决定了簇元素在程序框图中的“捆绑”和“解除捆绑”函数上作为接线端出现的顺序。右键单击簇边框，从快捷菜单中选择**重新排序簇中控件**可查看和修改簇顺序。

工具栏和簇的更改如图 5-17 所示。

图 5-17. 重新排序簇



白色框显示元素在簇顺序中的当前位置。黑色框显示元素的新的顺序。如要设置簇元素的顺序，在**单击设置**文本框中输入新的顺序编号并单击元素。元素的簇顺序发生改变，其他元素的簇顺序也进行相应的调整。单击工具栏上的**确认**按钮保存更改。单击**取消**按钮恢复原始的顺序。

自动调整簇的大小

按照下列步骤，使簇外框的大小符合其内容。

1. 右键单击簇的边框，从快捷菜单中选择**自动调整大小 » 调整为匹配大小**。
2. 向簇外框添加一个元素。选中**自动调整大小 » 调整为匹配大小**时，为簇添加新的元素时，簇将根据需要自动调整大小。
3. 右键单击簇外框，从快捷菜单中选择**自动调整大小 » 水平排列**或**自动调整大小 » 垂直排列**，水平或垂直排列簇元素。

使用簇函数

簇函数用于创建和操作簇。例如，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。

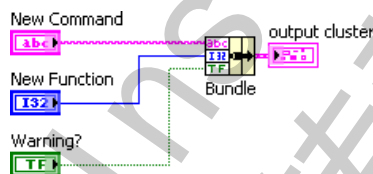
使用“捆绑”函数组合簇，“捆绑”函数和“按名称捆绑”函数更改簇，通过“解除捆绑”函数和“按名称解除捆绑”函数分解簇。

在程序框图上右键单击簇接线端，从快捷菜单中选择**簇、类与变体选板**在程序框图上放置“捆绑”函数、“按名称捆绑”函数、“解除捆绑”函数和“按名称解除捆绑”函数。“捆绑”函数和“解除捆绑”函数自动包含了正确的接线端数据。“按名称捆绑”函数和“按名称解除捆绑”函数显示簇中的第一个元素。通过定位工具调整“按名称捆绑”函数和“按名称解除捆绑”函数的大小，以显示其他簇元素。

组合簇

使用“捆绑”函数将单个元素组合为簇，也可使用该函数改变现有簇中独立元素的值，而无需为所有元素指定新值。通过定位工具调整函数的大小，或右键单击元素输入并从快捷菜单中选择**添加输入**。

图 5-18. 组合簇的程序框图



分解簇

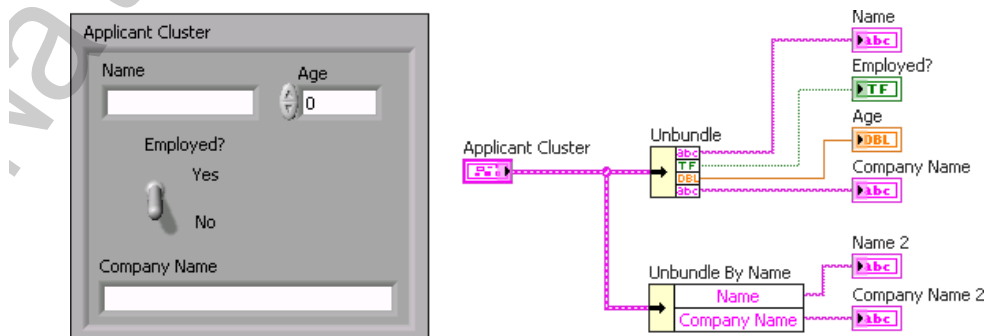
使用“解除捆绑”函数将簇分解为独立的元素。

使用“按名称解除捆绑”函数返回指定名称的簇元素。输出接线端的数量不取决于输入簇中的元素数量。

“解除捆绑”函数显示簇中的全部元素，“按名称解除捆绑”函数可设置为仅显示指定簇元素。使用操作工具单击函数的输出接线端，从下拉菜单中选择元素可指定“按名称解除捆绑”函数的显示元素。也可右键单击输出接线端，从快捷菜单中选择**选择元素**。

例如，在图 5-19 中使用“解除捆绑”函数，其带有 4 个输出接线端并分别对应簇的 4 个控件。用户必须了解簇的顺序，以正确关联解除捆绑簇的布尔接线端和相对应的簇开关。在本范例中，元素从 0 开始由顶部至底部排序。如使用“按名称解除捆绑”函数，用户可随机选择输出接线端数量并以任意顺序访问独立的元素。

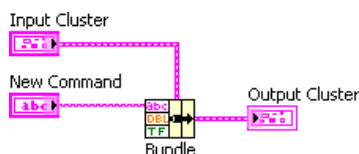
图 5-19. 解除捆绑和按名称解除捆绑



更改簇

连线簇输入时，仅可连线要更改的元素。例如，图 5-20 中的输入簇包含 3 个控件。

图 5-20. 用于更改簇的捆绑



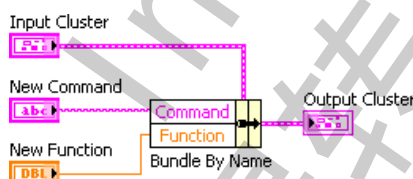
在了解簇的顺序的情况下，可使用“捆绑”函数，通过连线图 5-20 中所示元素更改 **Command** 的值。

或者使用“按名称捆绑”函数替换或访问现有簇的标签元素。“按名称捆绑”函数与“捆绑”函数类似，但前者是按照簇的顺序引用簇元素，后者是按照簇的标签引用簇元素。只能依据自带标签对元素进行访问。输入的数量不需要与**输出簇**的元素数量匹配。

“捆绑”函数显示簇中的全部元素，“按名称捆绑”函数可设置为仅显示指定簇元素。使用操作工具单击函数的输出接线端，从下拉菜单中选择元素可指定“按名称捆绑”函数的显示元素。也可右键单击输出接线端，从快捷菜单中选择**选择元素**。

在图 5-21 中，使用“按名称捆绑”函数更新 **Command** 和 **Function** 的值为 **New Command** 和 **New Function**。

图 5-21. 使用“按名称捆绑”更改簇



“按名称捆绑”函数用于开发过程中可能变化的数据结构。如添加新元素至簇或更改其顺序，用户无需重新连线“按名称捆绑”函数，因为其名称仍有效。

错误簇

LabVIEW 包含自定义的簇，称为错误簇。LabVIEW 使用错误簇返回错误信息。

关于错误簇的详细信息，见本手册的第 2，“疑难解答和调试 VI”和 *LabVIEW 帮助* 的**错误处理**主题。

F. 自定义类型

使用自定义类型定义自定义数组和簇。自定义类型为可用于多个 VI 的自定义数据类型（自定义输入控件、显示控件或常量）的主副本。使用自定义类型，可一次编辑即更新全部自定义数据类型的实例和副本。

自定义输入控件和显示控件

自定义输入控件和显示控件是对现有前面板对象集的扩展。用户可创建外观与内置 LabVIEW 控件不同的自定义控件。将这种自定义输入控件或显示控件保存在某个目录或 LLB 中，就可以在其他前面板上使用该自定义控件。也可自定义输入控件或显示控件创建图标，将图标添加到**输入控件**选板。

关于创建和使用自定义输入控件和自定义类型的详细信息，请查看 *LabVIEW 帮助* 的**创建自定义输入控件、显示控件和自定义类型**主题。

通过控件编辑器窗口自定义输入控件和显示控件。例如，改变控件的大小、颜色，控件中各元素的相对位置，向控件导入图像等。

可通过以下方式打开控件编辑器窗口：

- 右键单击前面板上的输入控件或显示控件，从快捷菜单中选择**高级 » 自定义**。
- 使用定位工具选中前面板上的某个输入控件或显示控件后，选择**编辑 » 自定义控件**。
- 使用**新建**对话框。

控件编辑器窗口中显示选中的前面板对象。控件编辑器具有两种模式：编辑模式和自定义模式。

控件编辑器窗口的工具栏用于提示用户当前处于编辑模式还是自定义模式。控件编辑器窗口以编辑模式打开。单击**切换至自定义模式**按钮可切换至自定义模式。单击**切换至编辑模式**按钮可切换至编辑模式。也可通过选择**操作 » 切换至自定义模式**或**操作 » 切换至编辑模式**实现两种模式间的切换。



与在前面板的编辑模式下的操作相同，控件编辑器的编辑模式可改变控件的大小及颜色，或在控件的快捷菜单中调整相应选项。

自定义模式可通过改变控件的各个部件实现控件的大幅改动。

编辑模式

在编辑模式下，可右键单击该控件并执行在 LabVIEW 编程环境中的操控设置。

1 编辑模式	5 分布对象
2 控件类型	6 调整对象大小
3 文本设置	7 重新排序
4 对齐对象	

自定义模式

在自定义模式下可改变控件的各个部件。单击**窗口 » 显示部件窗口**可查看自定义模式下可操控的部件列表。

1 自定义模式	5 分布对象
2 控件类型	6 调整对象大小
3 文本设置	7 重新排序
4 对齐对象	

自定义控件的一种为更改控件类型。根据**控件**下拉菜单中选择的可见项，可保存控件为控件、自定义类型和严格自定义类型。控件选项与在**控件**选板中的可选控件相同。根据需要更改控件，且用户更改的每个副本均保留其独立的属性。

保存自定义控件

创建自定义控件后，可保存该控件以做后续使用。默认情况下，控件在磁盘上以 `.ctl` 扩展名保存。

控件编辑器也可用于保存具有用户指定默认设置的控件。例如，通过控件编辑器更改波形图的默认属性，并保存该更改以在其他 VI 中调用。

自定义类型

自定义类型和严格自定义类型用于将所有自定义输入控件或显示控件实例与已保存的自定义输入控件或显示控件文件相连接。编辑已保存的自定义输入控件或显示控件文件可修改自定义输入控件或显示控件实例，以便在若干个 VI 中使用相同的自定义输入控件或显示控件。

在 VI 中使用自定义输入控件或显示控件后，该 VI 中自定义控件的实例与所保存的控件间的连接将不复存在。自定义控件的每个实例是一个单独、独立的副本。因此，改变自定义控件并不影响正在使用该自定义控件的 VI。如需使自定义输入控件或显示控件的实例与自定义输入控件或显示控件文件相连接，可将该自定义输入控件或显示控件另存为一个自定义类型或严格自定义类型。一个自定义类型或严格自定义类型的所有实例与其原始文件相连。

将自定义控件另存为自定义类型或严格自定义类型后，对该自定义类型或严格自定义类型所作的任何数据类型改动将对所有使用这些自定义类型或严格自定义类型的 VI 实例造成影响。与此同时，对严格自定义类型所作的外观改动也将影响前面板上该严格自定义类型的所有实例。

自定义类型为自定义输入控件或显示控件的每个实例指定了正确的数据类型。如自定义类型的数据类型发生改变，则该自定义类型的所有实例将自动更新。换言之，在使用了该自定义类型的每个 VI 中，各实例的数据类型将改变。然而，由于自定义类型仅规定了数据类型，仅有数据类型那部分的值被更新。例如，数值控件中的数据范围便不是数据类型的一部分。因此，数值控件的自定义类型并不定义该自定义类型实例的数据范围。同时，由于下拉列表控件各选项的名称没有定义其数据类型，因此在自定义类型中对下拉列表控件中各选项的名称进行改动将不会改变自定义类型实例中各项的名称。如在一个枚举型控件的自定义类型中改变其选项名称，由于选项名称也是枚举型控件数据类型的一部分，因此其实例将发生更新。自定义类型实例可拥有其唯一的标签、描述、默认值、大小、颜色或设定其风格为输入控件或显示控件，如滑动杆或旋钮。

如改变一个自定义类型的数据类型，LabVIEW 将尽可能把该自定义类型实例的原有默认值转换为新的数据类型。如数据类型被改为一个不兼容的类型，如数值控件被替换为字符控件，则 LabVIEW 将无法保留实例的默认值。如自定义类型的数据类型被改为其先前所无法兼容的数据类型，LabVIEW 将把该自定义类型的默认值设置为用户在 `.ctl` 文件中指定的值。如用户未指定默认值，LabVIEW 将使用相应数据类型的默认值。例如，如自定义类型从数值改为字符串，则 LabVIEW 将把与先前数值型数据类型相关的所有默认值替换为空字符串。

严格自定义类型

严格自定义类型将把实例除标签、描述和默认值外每个方面强制设置为与严格自定义类型相同。对于自定义类型，严格自定义类型的数据类型将在任何使用该严格自定义类型的场合下保持不变。严格自定义类型也对其他值进行了定义，如对数值控件及下拉列表控件中控件名称的范围检查。严格自定义类型可使用的 VI 服务器属性仅限于对控件外观产生影响的属性，包括可见、禁用、键选中、闪烁、位置和边界。

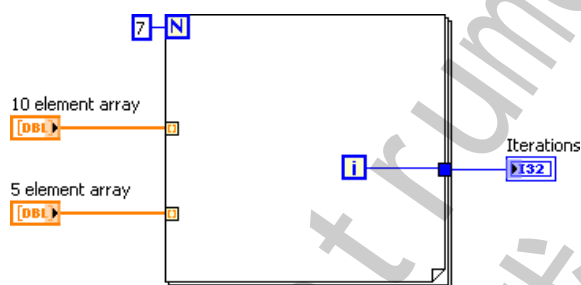
将实例与严格自定义类型移除连接，可阻止自定义类型实例进行自动更新。

自定义类型和严格自定义类型创建使用多个控件组成的簇的自定义控件。如需添加新的控件并传递值至每个子 VI，可添加新控件至自定义控件簇。该替换必须添加新控件至每个 VI 的前面板并重新连线。

自测：练习

- 用户可以创建由数组组成的数组。
 - 对
 - 错
- For 循环具有 2 个输入数组。两个隧道的自动索引均已启用。一个数组具有 10 个元素，第二个数组具有 5 个元素，总数接线端连接的值为 7。如图 5-22 所示。VI 运行结束后，循环显示控件的计数值等于多少？

图 5-22. 循环显示控件的值等于多少？



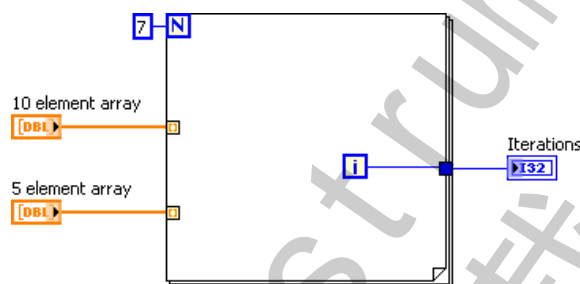
- 下列哪些自定义控件设置定义了控件的全部实例数据类型，但允许不同的颜色和字体样式？
 - 控件
 - 自定义类型
 - 严格自定义类型
 - 簇控件
- 表示圆的输入数据：X 位置、Y 位置和半径。将来可能需要更改数据以包含圆的颜色。在应用中应当使用何种数据类型表示圆？
 - 使用三个独立控件表示 2 个位置量和半径值。
 - 使用簇，簇中包含所有数据元素。
 - 包含簇的自定义控件。
 - 包含簇的自定义类型控件。
 - 包含 3 个元素的数组。

International Instruments
不得转载

自测：练习答案

- 用户可以创建由数组组成的数组。
 - 对
 - 错**
 错。不能拖曳任何数组数据类型至数据外框内。但可创建 2 维数组。
- For 循环具有 2 个输入数组。两个隧道的自动索引均已启用。一个数组具有 10 个元素，第二个数组具有 5 个元素，总数接线端连接的值为 7。如图 5-23 所示。VI 运行结束后，**循环**显示控件的计数值等于多少？

图 5-23. 循环显示控件的值等于多少？



显示控件的计数值 = 4

LabVIEW 不能超出数组的大小。这有助于避免编程错误。LabVIEW 数学函数的工作方式与此相同——一连包含 10 个元素的数组至“加”函数的 **x** 输入端，包含 5 个元素的数组至“加”函数的 **y** 输入端，输出为包含 5 个元素的数组。

尽管循环执行 5 次，但循环计数是由零开始的。因此循环计数显示控件的值为 4。

- 下列哪些自定义控件设置定义了控件的全部实例数据类型，但允许不同的颜色和字体样式？
 - 控件
 - 自定义类型**
 - 严格自定义类型
 - 簇控件
- 表示圆的输入数据：**X 位置、Y 位置和半径**。将来可能需要更改数据以包含圆的颜色。在应用中应当使用何种数据类型表示圆？
 - 使用三个独立控件表示 2 个位置量和半径值。
 - 使用簇，簇中包含所有数据元素。
 - 包含簇的自定义控件。
 - 包含簇的自定义类型控件。**
 - 包含 3 个元素的数组。

笔记

Practical Instruments
不得转载

管理文件和硬件资源

您已经学习了采集数据和显示数据的方法，但存储数据通常为项目非常重要的一部分。您也已经学习了设置硬件及在 MAX 中配置它们。在本课程中，您将学习存储数据、通过 DAQmx API 编程基本的 DAQ 应用程序、通过 VISA API 控制独立仪器和 LabVIEW 中的仪器驱动。

主题

- A. 硬件和软件资源
- B. 文件 I/O
- C. 使用 DAQ 系统采集测量
- D. 仪器控制

A. 硬件和软件资源

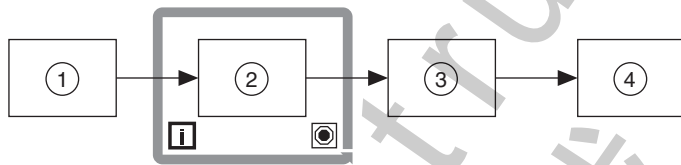
资源是指系统上可分配的文件、硬件设置、对象或网络连接。系统通过路径、名称、端口或其他标识符指定资源。通过上述标识符和 LabVIEW 提供的 VI 及函数，可编程系统去访问上述资源。

访问 LabVIEW 资源

通常资源访问包含下列操作，如图 6-1 所示：

1. 创建资源。
2. 读取和写入资源。
3. 关闭资源。
4. 检查资源是否生成错误。

图 6-1. 常规资源访问



- 1 打开、初始化和创建资源—LabVIEW 创建一个引用句柄，其用作资源的唯一标识符。
- 2 读取 / 写入资源
- 3 关闭资源—引用句柄随即失效
- 4 检查错误—显示资源是否出错



提示 与创建资源相关的函数或 VI 名称通常包含下列动词之一：打开、初始化或创建。创建资源的函数或 VI 将资源路径或设备名称用作输入，并为资源创建唯一的标识符。

引用句柄

引用句柄是资源的唯一标识符。打开一个文件、设备或网络连接时，LabVIEW 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、设备或网络连接进行的所有操作均使用引用句柄来识别每个对象。引用句柄控件用于将一个引用句柄传进或传出 VI。例如，引用句柄控件可在不关闭或不重新打开文件的情况下修改其指向的文件内容。

由于引用句柄是一个打开资源的临时指针，因此它仅在资源打开期间有效。如关闭资源，LabVIEW 会将引用句柄与资源分开，引用句柄随即失效。如再次打开资源，LabVIEW 将创建一个与第一个引用句柄不同的新引用句柄。LabVIEW 将为引用句柄指向的资源分配内存空间。关闭引用句柄，该资源就会从内存中释放。

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，因此可以对单一资源执行并行但相互独立的操作。如一个 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，如果用户在结束使用引用句柄时就立即将其关闭，可最有效地利用内存空间和其它资源，这是一个良好的编程习惯。关闭引用句柄的顺序与打开时相反。例如，如果获得了对象 A 的一个引用句柄，然后对对象 A 调用方法以获得对象 B 的引用句柄，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。

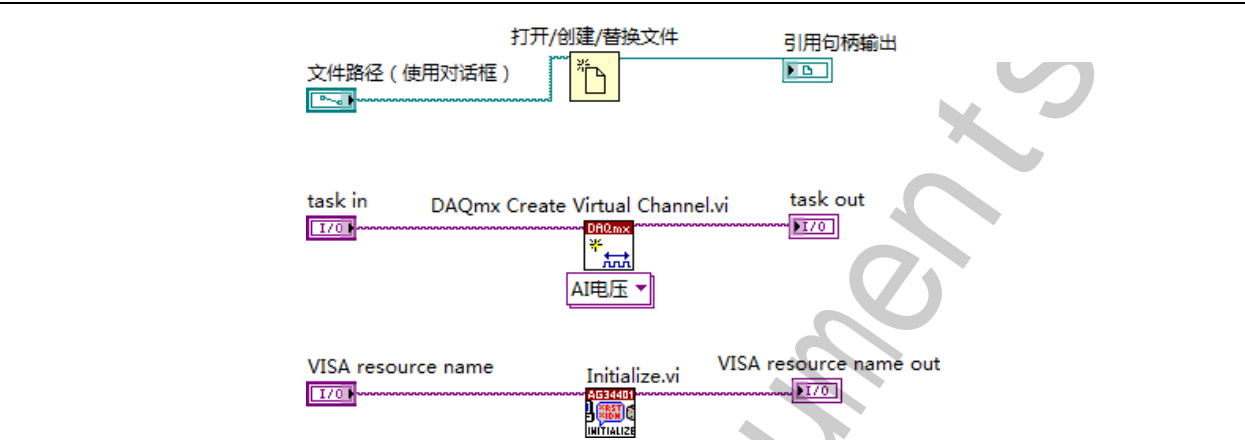
引用句柄 API 范例

使用 DAQmx 驱动程序 API 连接 DAQ 设备时，引用被称为任务。任务是一个或多个虚拟通道以及定时、触发等属性的集合。就概念而言，任务是要执行的信号测量或信号生成。

通过 VISA API 控制仪器时，引用被称为 VISA 会话。VISA 会话由“VISA 资源名称”输入控件保持。

图 6-2 为不同的 API 及 API 用于访问资源的引用句柄的示意图。

图 6-2. 引用句柄范例



B. 文件 I/O

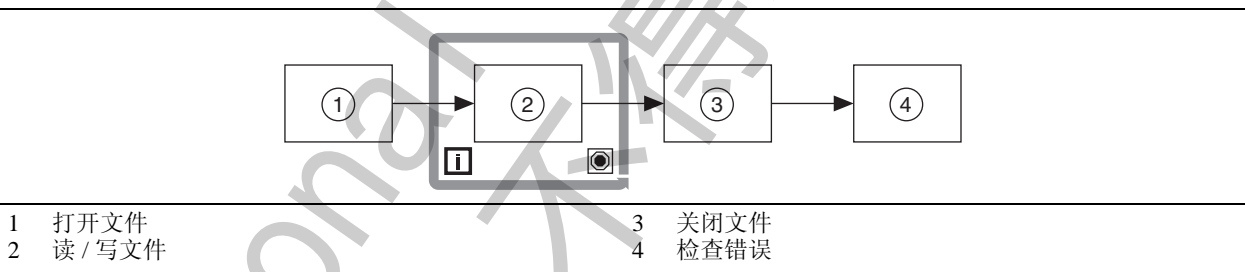
文件 I/O 操作可传输数据至文件或从文件传出数据。使用文件 I/O VI 和函数实现文件 I/O 的全部功能。

文件 I/O

常规的文件 I/O 操作包括以下流程，如图 6-3 所示。

- 1. 创建或打开一个文件。文件打开后，引用句柄是该文件的唯一标识符。
- 2. 文件 I/O VI 或函数从文件中读取或向文件写入数据。
- 3. 然后关闭该文件。

图 6-3. 常规文件 I/O 操作的步骤



文件格式

LabVIEW 可使用或创建下列文件格式：二进制、ASCII、LVM 和 TDMS。

- **二进制**—二进制文件是所有其他文件格式的底层格式。
- **ASCII**—ASCII 文件是二进制文件的一种特殊类型，二进制文件是多数程序的标准文件格式。其包含一系列的 ASCII 代码。ASCII 文件也称为文本文件。
- **LVM**—LabVIEW 测量数据文件 (.lvm) 是用制表符分隔的文本文件，可以用电子表格应用程序或文本编辑应用程序打开。 .lvm 文件包含诸如数据生成日期和时间的信息。该文件格式是用于 LabVIEW 的特殊 ASCII 文件类型。
- **TDMS**—用于 NI 产品的特殊二进制文件格式。它实际上由两个独立的文件组成。其中包含数据和数据存储属性的二进制文件，及二进制索引文件。该索引文件提供二进制文件中的所有属性和指针信息。

通过本课程，您将学习创建文本 (ASCII) 文件的方法。如磁盘空间、文件 I/O 操作速度和数字精度不是主要考虑因素，或无需进行随机读写，应使用文本文件存储数据。

在第 1 课，“LabVIEW 导航”中使用 LVM 文件。如要了解更多二进制和 TDMS 文件的详细信息，见 *LabVIEW 帮助* 或 *LabVIEW 核心教程*（二）。

高级文件 I/O

某些文件 I/O VI 执行文件 I/O 的全部功能—打开、读 / 写和关闭。执行上述全部步骤的 VI 被称为高级 VI。但高级 VI 不一定比底层 VI 和设计用于某指定步骤的函数更加高效。写入循环中的文件时，请使用底层文件 I/O VI。写入独立操作中的文件时，可使用高级文件 I/O VI。

LabVIEW 中的高级文件 I/O VI 包括：

- **写入电子表格文件**—转换二维或一维双精度数值数组为文本字符串并将字符串写入新的 ASCII 文件，或添加字符串至现有文件。同时也可以转置数据。VI 在向文件中写入数据之前，将先打开或创建该文件，并且在完成写操作时，关闭该文件。可使用该 VI 创建多数电子表格应用程序可读取的文本文件。
- **读取电子表格文件**—在数值文本文件中从指定字符偏移量开始读取指定数量的行或列，并将数据转换为双精度的二维数组。VI 在从文件中读取数据之前，先打开该文件，并且在完成读操作时，关闭该文件。可使用该 VI 读取以文本格式存储的电子表格文件。
- **写入测量文件**—写入数据至文本测量文件 (.lvm) 或二进制测量文件 (.tdms) 的 Express VI。用户可指定保存方式、文件格式 (.lvm 或 .tdms)、段首类型和分隔符。
- **读取测量文件**—从文本测量文件 (.lvm) 或二进制测量文件 (.tdms) 读取数据的 Express VI。用户可指定文件名称、文件类型和段大小。



提示 避免在循环内使用高级 VI，因为高级 VI 每次运行时均执行打开和关闭操作。

底层文件 I/O

底层文件 I/O VI 和函数每个仅执行一项文件 I/O 功能。例如，一个函数打开 ASCII 文件，一个函数读取 ASCII 文件，另一个函数关闭 ASCII 文件。文件 I/O 位于循环内部时，使用底层函数。

底层函数和流盘

“文件 I/O”函数还可用于流盘操作，它可以减少函数因打开和关闭文件与操作系统交互的次数，从而节省内存资源。流盘是一项在进行多次写操作时保持文件打开的技术，如在循环中使用流盘。

如将路径控件或常量连接至“写入文本文件”、“写入二进制文件”或“写入电子表格文件”函数，则函数将在每次函数或 VI 运行时打开关闭文件，增加了系统开销。避免经常性打开和关闭一个文件，VI 的效率更高。

如要避免打开和关闭同一个文件，用户需要传递文件的引用至循环。打开一个文件、设备或网络连接时，LabVIEW 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、设备或网络连接进行的所有操作均使用引用句柄来识别每个对象。

图 6-4 和图 6-5 中的范例为使用流盘的优势。在图 6-4 中，VI 必须在每个循环计数打开和关闭文件。图 6-5 使用流盘减少 VI 与操作系统间的交互，交互用于打开和关闭文件。每次循环开始前打开文件，循环结束后关闭文件。每个计数循环可节省两次文件操作。

图 6-4. 非流盘范例

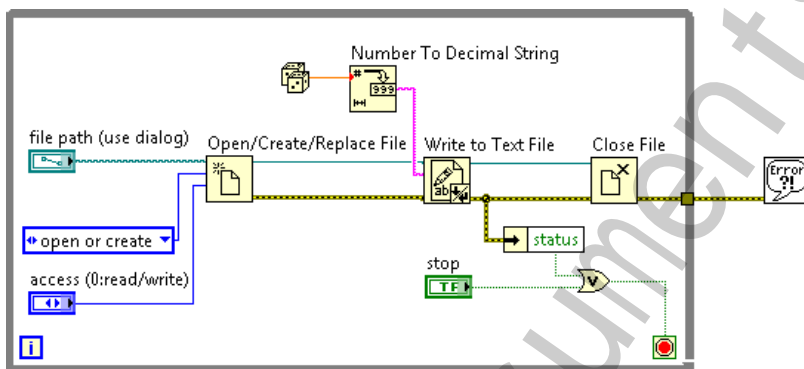
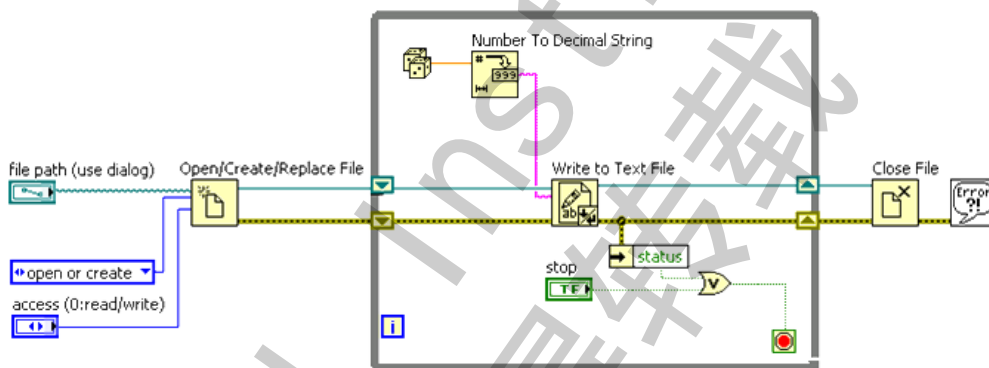


图 6-5. 流盘范例



LabVIEW 数据目录

默认的 LabVIEW Data 目录可存储 LabVIEW 生成的数据文件。例如，.lvn 或 .txt 文件。LabVIEW 会将 LabVIEW Data 目录安装在操作系统中默认的文件目录中，以便于管理和查找 LabVIEW 生成的数据文件。默认情况下，“写入测量文件” Express VI 将所生成的 .lvn 文件保存在该目录，且“读取测量文件” Express VI 可从该目录读取相关文件。“默认数据路径”常量和“应用程序：默认：数据目录”属性也会默认返回 LabVIEW Data 目录。



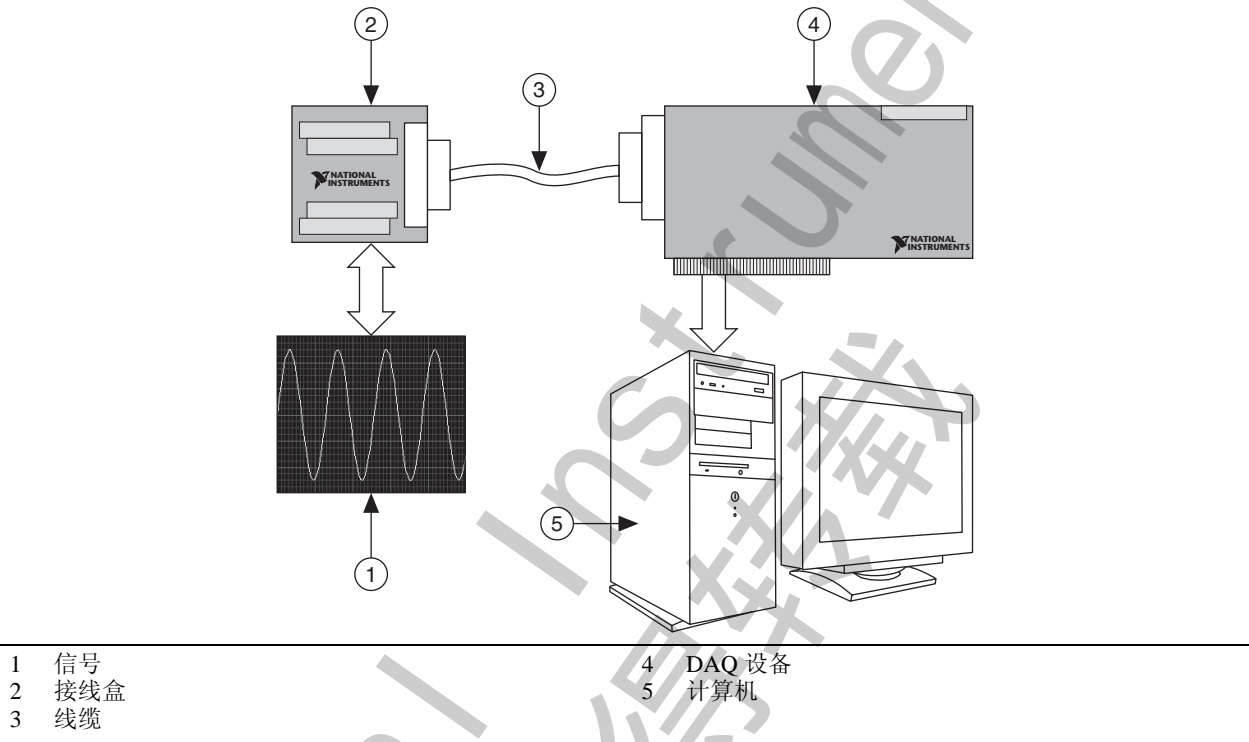
选择**工具»选项**，从**类别**列表中选择**路径**指定其他默认数据路径。该默认数据目录不同于默认目录，后者是指定用来保存所创建的新 VI、用户自定义控件、VI 模板或其它所创建的 LabVIEW 文档的目录。

C. 使用 DAQ 系统采集测量

数据采集 (DAQ) 系统使用数据采集设备传输调理电信号至计算机，用于软件分析和数据记录。用户可选择使用 PCI 总线、PCI Express 总线、PXI 总线、计算机 USB 或 IEEE 1394 端口。本章介绍了用于数据采集系统的硬件及配置设备的方法。

典型的 DAQ 系统包含 3 个基础硬件类型—接线盒、线缆和 DAQ 设备。如图 6-6 所示。

图 6-6. 典型 DAQ 系统



转换物理现象为可测量的信号后，无论是否进行信号调理，均需获取该信号。通过接线盒、线缆、DAQ 设备和计算机采集信号。上述硬件将标准计算机用作测量和自动化系统。

使用接线盒和线缆

接线盒用于连接信号。它包含用于连接信号的螺丝或弹簧接线端和线缆连接器，连接器用于连接接线盒至 DAQ 设备。接线盒具有 100、68 或 50 个端子。选择接线端的类型取决于下列 2 个因素—设备及测量的信号数量。68 接线端的接线盒相对于 50 接线端的接线盒，能够提供更多的地接线端。具有更多的地接线端可避免将导线一个地接线端上重叠，其可能导致信号干扰。

接线盒可为屏蔽或非屏蔽式。屏蔽式接线盒提供更好的噪声保护。某些接线盒包含其他功能。例如，冷端补偿。冷端补偿对正确测量热电偶非常有用。

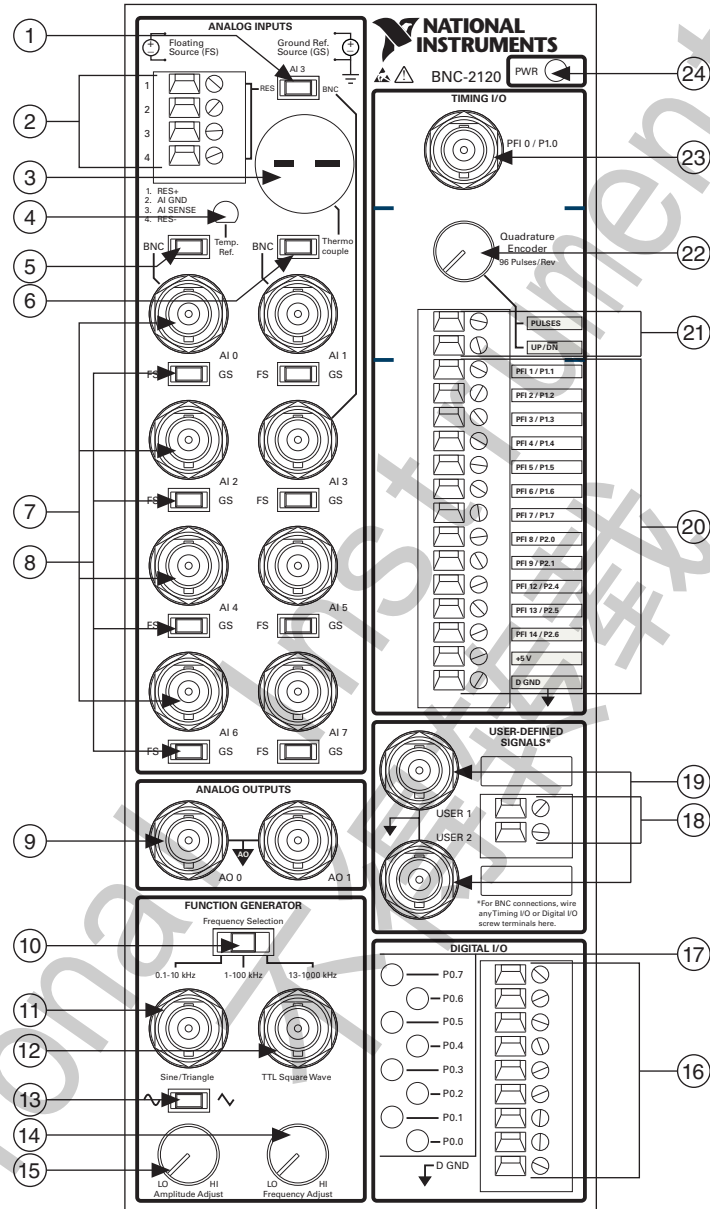
线缆从接线端传输信号至 DAQ 设备。线缆为 100、68 和 50 引脚规格。根据使用的 DAQ 设备和接线盒选择相应的配置。线缆与接线盒类似，可为屏蔽或非屏蔽式。

关于 NI 产品目录的 DAQ 章节，或访问 ni.com/products 了解接线盒和线缆类型的详细信息。

BNC-2120

图 6-7 为 BNC-2120。BNC-2120 用于完成本课程中的练习。

图 6-7. BNC-2120



- | | |
|----------------------------|-------------------|
| 1 RES/BNC 开关 (AI 3) | 13 正弦 / 三角波开关 |
| 2 电阻测量螺栓端子 | 14 频率调整旋钮 |
| 3 热电偶输入连接器 | 15 幅值调整旋钮 |
| 4 温度参考 | 16 数字 I/O 线螺栓端子 |
| 5 BNC/Temp. Ref. 开关 (AI 0) | 17 数字 I/O LED |
| 6 BNC/ 热电偶开关 (AI 1) | 18 用户定义螺栓端子 |
| 7 模拟输入 BNC 连接器 | 19 用户定义 BNC 连接器 |
| 8 FS/GS 开关 | 20 定时 I/O 螺栓端子 |
| 9 模拟输出 BNC 连接器 | 21 正交编码器螺栓端子 |
| 10 频率量程选择开关 | 22 正交编码器旋钮 |
| 11 正弦 / 三角波 BNC 连接器 | 23 定时 I/O BNC 连接器 |
| 12 TTL 方波 BNC 连接器 | 24 电源显示 LED |

使用 DAQ 设备

多数 DAQ 设备均具有下列 4 个基础组成—模拟输入、模拟输出、数字 I/O 和计数器。

通过不同的总线结构，用户可将 DAQ 设备的信号测量结果传递至计算机。例如，使用插入计算机 PCI 或 PCI Express 总线的 DAQ 设备。DAQ 设备连接至桌面计算机的 DAQ 设备，或 DAQ 设备连接至计算机的 USB 端口。或者使用 PXI/CompactPCI 创建一个便携、多功能及坚固的测量系统。

如不具有 DAQ 硬件设备，可在 Measurement and Automation Explorer (MAX) 中模拟设备，进行软件测试。在本课程的“仿真 DAQ 设备”部分将学习仿真设备。

关于 DAQ 设备具体类型的详细信息，见 NI 产品目录的 DAQ 章节或访问 ni.com/products。

模拟输入

模拟输入是测量模拟信号，将测量结果传输至计算机用于分析、显示或存储的过程。模拟信号为连续变化的信号。模拟输入通常用于测量电压或电流。不同类型的设备可用于模拟输入。例如，多功能 DAQ (MIO) 设备、高速数字化仪、数字万用表 (DMM) 和动态信号采集 (DSA) 设备。

通过计算机采集模拟信号需要执行模数转换，该过程采集电信号并将其转换为计算机可处理的数字数据。模数转换器 (ADC) 是将电压电平转换为一系列 0 和 1 信号的电路组件。

ADC 在每个采样时钟的上升沿或下降沿采样模拟信号。在每个周期，ADC 记录模拟信号的快照，信号可被测量并转换为数字值。采样时钟控制输入信号的采样频率。由于输入未知的信号为具有有限精度的实际信号，ADC 使用固定精度贴近信号。ADC 获得该近似值后，将该值转换为一系列数字值。某些转换方法不需要该步骤，因为 ADC 接近该近似值，转换直接生成数字值。

模拟输出

模拟输出是由计算机生成电信号的过程。模拟输出是通过数模 (D/A) 转换生成的。任务可用的模拟输出类型为电压和电流。执行电压或电流任务时，必须安装可生成该类信号的兼容设备。

数模转换是模数转换的反过程。在数模转换中，计算机生成数据。数据可能已经使用模拟输入采集，或已通过计算机上的软件生成。数模转换器 (DAC) 接收数据，并使用该数据随时间更改输出端电压。DAC 生成一个模拟信号，可被传输至其他设备或电路。

DAC 具有更新时钟，用于通知 DAC 生成新值的时间。更新时钟函数与 ADC 的采样时钟函数类似。在每个时钟周期，DAC 转换数字值为模拟电压，并在引脚端生成一个电压输出。使用高速时钟时，DAC 可生成接近平稳持续变化的信号。

数字 I/O

数字信号为通过线缆传输数字数据的电信号。上述信号通常具有两个状态—开和关（通常也称为高和低），或是 1 和 0。通过线缆传输数字信号时，发送方给线缆施加电压，接收方通过电压值判定发送的值。每个数字值的电压量程取决于所用的电压电平标准。数字信号用途较多；最简单的数字信号应用为控制或测量数字或有限个状态的设备。例如，开关和 LED。数字信号也可传输数据；它们可被用于编程设备或设备间通信。此外，数字信号可被用作时钟或触发器，控制和同步其他测量。

通过 DAQ 设备的数字信号线采集数字信号值。该采集是基于软件定时。某些设备的数据线可根据需要配置为测量或生成数字样本。每条数据线对应于任务的一个通道。

通过 DAQ 设备的数字端口从多根数据线采集数字信号值。该采集是基于软件定时。可根据需要将端口配置为测量或生成数字样本。每个端口对应于任务的一个通道。

计数器

计数器是数字定时设备。通常计数器用于事件计数、频率测量、周期测量、位置测量和脉冲生成。

配置计数器用于简单事件计数时，源收到有效边沿时，计数器的值递增一次。为使计数器在有效边沿递增，计数器必须已启用。计数器具有固定的计数上限，该值取决于计数器的分辨率。例如，一个 24 位计数器的最大计数值为：

$$2^{(24 \text{ (计数器分辨率)})} - 1 = 2^{24} - 1 = 16,777,215$$

当 24 位计数器达到 16,777,215 时，其到达计数终点。下一个有效边沿将强制计数器回零，并从 0 重新开始计数。

使用 DAQ 软件

NI 数据采集设备带有一个驱动程序引擎，其与设备和应用程序软件通信。LabVIEW 可用于与上述驱动程序引擎通信。NI-DAQmx 包含 VI、函数和控制测量设备的开发工具。关于编程 DAQmx 的详细信息见“在 LabVIEW 中编程 DAQ”。

此外，MAX 可用于配置数据采集设备。本章节将介绍驱动程序引擎和使用 MAX 配置数据采集设备的方法。

DAQ 硬件配置

使用数据采集设备前，必须确定通过配置设备，软件可与设备进行通信。在本课程中，设备已配置好。

Windows

Windows Configuration Manager 中将列出全部安装在计算机上的硬件，包括 NI DAQ 设备。对于即插即用 (PnP) 设备（例如，E 系列 MIO 设备），Windows Configuration Manager 将自动检测和配置该设备。对于非即插即用设备或传统设备，必须通过 Windows 控制面板中的**添加新硬件**手动配置设备。

访问设备管理器可验证 Windows 配置。查看**数据采集设备**，其中将列出计算机上安装的全部 DAQ 设备。双击 DAQ 设备，显示带有选项卡页面的对话框。**常规**选项卡将显示设备的全部信息。**驱动程序**选项卡指定驱动程序版本和 DAQ 设备的位置。**详细信息**选项卡包含硬件配置的额外信息。**资源**选项卡指定用于设备的系统资源。例如，中断电平、DMA 和用于软件配置设备的基地址。

Measurement & Automation Explorer

MAX 建立全部设备和通道配置参数。在计算机上安装 DAQ 设备后，必须运行该配置程序工具。MAX 读取设备管理器写入 Windows 注册表的信息，然后分配逻辑设备编号给每个 DAQ 设备。通过设备编号指向 LabVIEW 中的设备。双击 MAX 的桌面快捷方式或在 LabVIEW 中选择**工具 » Measurement & Automation Explorer** 启动 MAX 应用程序。下列窗口为 MAX 窗口的主窗口。MAX 也可用于 SCXI 和 SCC 配置。

图 6-8. MAX 主窗口



使用配置程序工具可设置的设备参数取决于设备。MAX 在 Windows 注册表中保存逻辑设备编号和配置参数。

在计算机上安装设备时，Windows 的即插即用功能自动检测和配置无开关 DAQ 设备（例如，NI PCI-6024E）。

仿真 DAQ 设备

在 NI-DAQmx 7.4（或更高版本）中创建 NI-DAQmx 仿真设备。使用 NI-DAQmx 仿真设备，在不具备硬件的应用程序中实现 NI 产品功能。具有可用硬件设备时，可通过 MAX 便携配置向导将 NI-DAQmx 仿真设备配置导入至物理设备。通过 NI-DAQmx 仿真设备，可导入物理设备的配置至未安装物理设备的系统。使用 NI-DAQmx 仿真设备，可在便携系统上使用应用程序并将数据返回原始系统。用户可轻松导入应用程序相关内容。

在 LabVIEW 中编程 DAQ

您现在已经了解了传感器、信号、DAQ 设备配置和 MAX，可以开始学习使用 LabVIEW 开发数据采集应用程序。NI-DAQmx 支持除 LabVIEW 之外的软件包，但本课程中仅讨论通过 LabVIEW 执行的 DAQ 应用程序开发。

DAQmx 名称控件

DAQmx 名称控件选板包含任务名控件、通道名、物理通道、接线端、换算名、设备名和开关。右键单击 DAQmx VI 的相应输入接线端，选择 **创建 » 输入控件** 也可创建上述控件。关于上述控件的详细信息，见 *NI-DAQmx 帮助*。

图 6-9. DAQmx 名称控件选板



DAQmx - 数据采集 VI

使用 NI-DAQmx VI 和 NI-DAQ 硬件设备开发仪器测量、采集和控制应用程序。关于 NI-DAQmx 支持的完整设备的列表，见 *DAQ 入门指南* 或 *NI-DAQ 自述文件*。

DAQmx 一数据采集选板包含下列常量和 VI。

常量

- DAQmx 任务名称常量—列出所有使用 DAQ 助手创建并保存的任务。右键单击常量，从快捷菜单中选择过滤 I/O 名称，限制常量可显示的任务和在常量中的输入。
- DAQmx 全局通道—列出所有使用 DAQ 助手创建并保存的全局通道。单击并选择浏览，选择多个通道。右键单击常量，从快捷菜单中选择过滤 I/O 名称，限制常量可显示的通道和在常量中的输入。

VI

- DAQmx 创建虚拟通道 VI—创建单个或多个虚拟通道，并将其添加至任务。多态 VI 的实例对应于通道的 I/O 类型（例如，模拟输入、数字输出或计数器输出）；测量或生成（例如，温度测量、电压生成或事件计数）或在某些情况下使用的传感器（例如，用于温度测量的热电偶或 RTD）。
创建虚拟通道时，函数的设置与用户在 MAX 中的配置相同。如应用程序可能周期性的更改物理通道的连接，但不会更改其他重要设置（例如，接线端配置模式或自定义换算）时可使用该函数。通过物理通道下拉菜单指定 DAQ 设备的设备编号及信号连接的实际物理通道。
- DAQmx 读取 VI—读取用户指定任务或虚拟通道中的采样。该多态 VI 的实例指定返回采样的格式、是否立即读取单个采样或多个采样以及是否读取一个或多个通道。右键单击“DAQmx 读取”VI 选择实例，单击“选择类型”为读取操作包含额外的配置选项。
- DAQmx 写入 VI—在用户指定的任务或虚拟通道中写入采样数据。该多态 VI 的实例指定要写入采样的格式、是否写入单个或多个采样及是否写入一个或多个通道。右键单击“DAQmx 写入”VI 选择实例，单击“选择类型”为写入操作包含额外的配置选项。
- DAQmx 结束前等待 VI—等待测量或生成操作完成。该 VI 确保用户停止任务前，指定的操作已结束。
- DAQmx 定时 VI—配置要获取或生成的采样数，并创建所需的缓冲区。该多态 VI 的实例对应于任务所用的定时类型。
- DAQmx 触发 VI—配置任务的触发。该多态 VI 的实例对应于触发器和配置的触发器类型。

- DAQmx 开始任务 VI —使任务处于运行状态，开始测量或生成。该 VI 适用于某些应用程序。
- DAQmx 停止任务 VI —停止任务，使其返回 DAQmx 开始任务 VI 尚未运行或 DAQmx 写入 VI 运行时自动开始输入值为 TRUE 的状态。
- DAQmx 清除任务 VI —清除任务。在清除之前，VI 将中止该任务，并在必要情况下释放任务保留的资源。清除任务后，将无法使用任务的资源。必须重新创建任务。

图 6-10. DAQmx 数据采集选板



D. 仪器控制

在基于 PC 机的自动化测试系统中，可使用任何类型的仪器。LabVIEW 允许你控制任意类型的虚拟仪器。例如，放大器、分析器、校准器、示波器、电源、开关、信号生成器和传感器等。

可根据需要将各种类型的仪器在系统中组合使用。最常见的仪器接口包括 GPIB、USB、以太网、串口和模块化仪器。其他类型的仪器包括图像采集、运动控制、并行端口和 NI-CAN 等。本课程主要使用 GPIB。

通过 PC 机控制仪器时，需要先了解仪器的属性，例如仪器使用的通信协议及设备使用的命令集。关于仪器属性的详细信息，见仪器程序员手册或用户手册。

仪器控制的优势

LabVIEW 简化了经总线控制和采集仪器数据的过程。通过设备的组合可自动化测量、分析采集到的数据并将结果记录至磁盘。该自动化过程可节省大量的时间和金钱。还可使用单个应用程序控制多个任务，且可通过仪器增强和同步上述任务。

LabVIEW 带有多個幫助簡化儀器控制編程的工具，包括儀器驅動、範例、向導和 Express VI。這些工具通過減少重複性的耗時任務，節省了用戶的時間。用戶可將更多精力放在開發應用程序部分。幾乎全部儀器的驅動程序均為免費。

GPIB

ANSI/IEEE Standard 488.1-1987, 即通用接口总线 (GPIB) 为多个厂商支持的仪器和控制器通信的标准接口。GPIB 仪器为测试和制造工程师提供了最大的供应商和通用仪器选择范围, 以专业化纵向市场测试应用程序。GPIB 仪器常用作独立的台式仪器, 在这类仪器上进行的测量是手动完成的。使用个人电脑控制 GPIB 仪器, 可以使这些测量自动完成。

IEEE 488.1 包含电子、机械和功能规范。ANSI/IEEE 标准 488.2-1992 通过定义总线通信协议、常用数据代码和格式集、常用设备命令的通用集，扩展了 IEEE 488.1。

GPIB 是数字、8 位并行通信接口，数据传输速率为 1 Mbyte/s 或更高并使用三线握手。总线支持一个系统控制器（通常为计算机），最多接 14 个额外设备。

GPIB 协议将设备分为控制器、通话器和侦听器，以确定哪一个设备拥有总线的控制权。每台设备都有一个唯一的 GPIB 物理地址，范围在 0 和 30 之间。控制器定义了通信链接，对需要服务的设备作出响应，发送 GPIB 命令和传递 / 接收总线控制权。控制器指示通话器发话，并将数据放在 GPIB 上。每次只能有一台设备通话。控制器选定侦听器侦听，从 GPIB 读取数据。可令几台设备同时侦听。

数据传输终止

通知侦听器所有数据传输完毕。通过下列三种方式可终止 GPIB 数据传输：

- GPIB 包含结束或识别线 (EOI) 硬件线，传输数据最后一个字节后将 EOI 置为有效。这是推荐的终止方式。
- 在所传输数据的起始位置放置一个串尾符 (EOS)。某些仪器使用这种终止方法。
- 允许侦听器通过握手计算传输字节数，到达计数上限时停止读取操作该方法通常为默认的终止方法，根据 EOI、EOS（如使用）和字节计数的逻辑或终止传输。侦听器端的字节计数通常被设置为高于预期的字节计数，以避免丢失采样。

数据传输速率

如要实现 GPIB 设计用于的高数据传输速率，必须限制总线端的设备数量和设备间的物理距离。

用 HS488 设备和控制器可以更快地传输数据。HS488 是对 GPIB 的扩展，大部分 NI 控制器都支持 GPIB。



注 关于 GPIB 的详细信息，请访问 NI GPIB 支持网站 ni.com/support/gpibsupp.htm。

GPIB 仪器的配置

通过 LabVIEW 进行仪器控制的软件架构类似与 DAQ 架构。仪器接口（类似 GPIB）包含驱动程序的集合。使用 MAX 配置接口



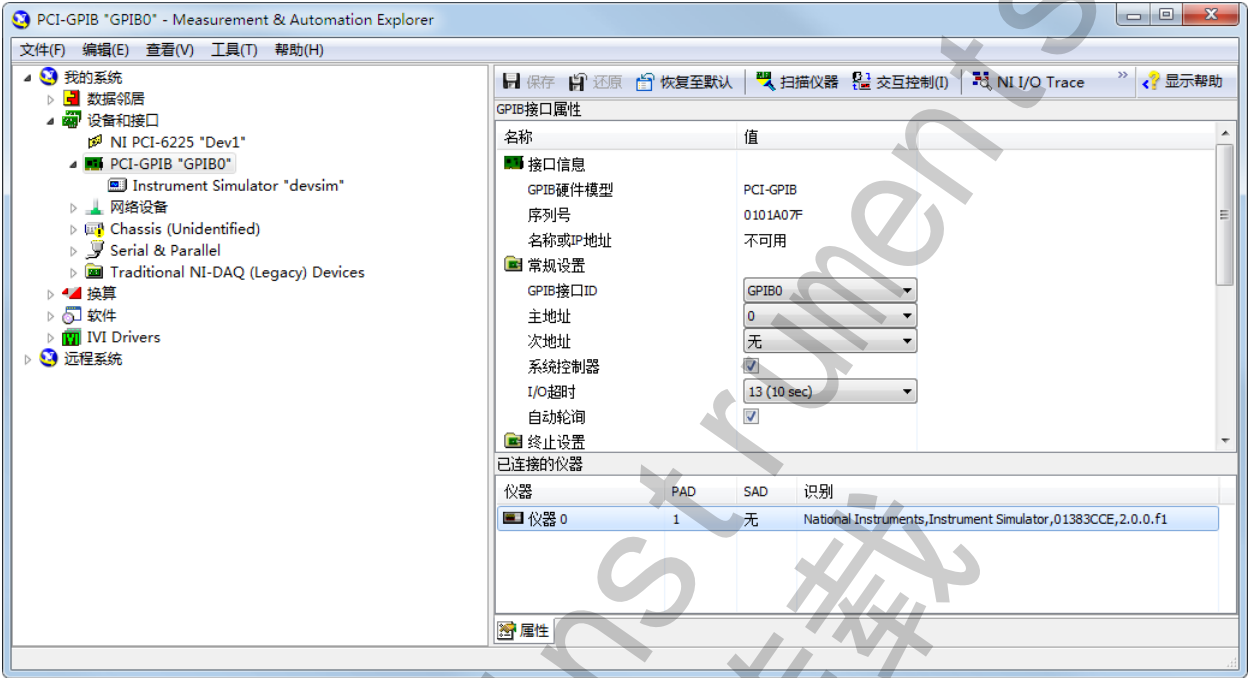
注 多数 GPIB 驱动程序可通过 ni.com/support/gpib/versions.htm 下载。除非发行说明中声明，否则总是安装上述驱动程序的最新版本。

MAX (Windows: GPIB)

通过 MAX 配置和测试 GPIB 接口。MAX 与随驱动程序安装的多种诊断和配置工具交互，MAX 还与 Windows 注册表和设备管理器交互。驱动程序软件的格式为 DLL，且包含全部与 GPIB 接口通信的函数。仪器 I/O VI 和函数直接调用驱动程序软件。

双击 MAX 的桌面快捷方式或在 LabVIEW 中选择**工具 »Measurement & Automation Explorer**，启动 MAX 应用程序。下列范例显示了在 MAX 中单击工具栏上的**扫描仪器**后，显示的 GPIB 接口。

图 6-11. MAX 中的 GPIB 接口

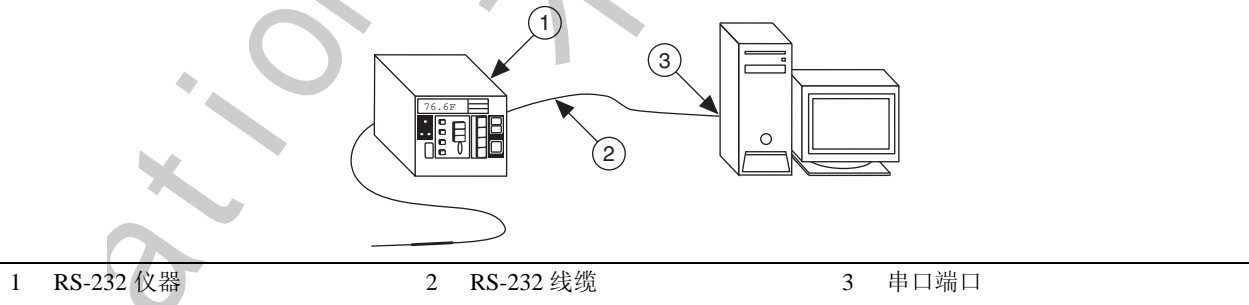


在 MAX 中右键单击每个项，从快捷菜单中选择选项配置对象。

串口

串口通信在一台计算机和一个外围设备（如一台可编程仪器或另一台计算机）间传输数据。串口通信使用发送器每次向接收器发送一位数据，数据经过一条通信线到达接收器。如数据传输率较低，或数据传输的距离较长，应使用这种方法。大部分计算机都有一个或多个串口，因此除了需要用电线连接设备和计算机或两台计算机以外，不需要其他多余的硬件。

图 6-12. 串口仪器范例



使用串口通信前必须指定四个参数：传送的波特率、对字符编码的数据位数、可选奇偶校验位的奇偶性和停止位数。一个字符帧将每个传输过来的字符封装成单一的起始位后接数据位的形式。

波特率是对使用串口通信时数据在仪器间传送速度的测量。

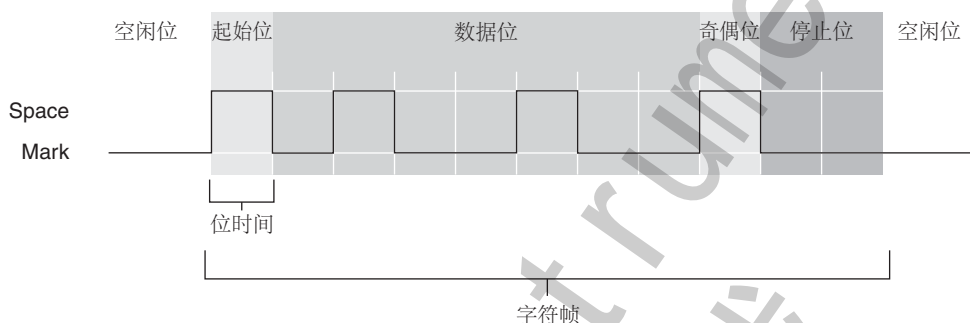
数据位是倒置和反向传输的，即数据位使用的是反向逻辑，而传输的顺序是从最低有效位 (LSB) 到最高有效位 (MSB)。解析字符帧中的数据位时，必须从右到左读取，在负电压时读取 1，正电压时读取 0。

字符帧中，数据位之后紧随一个可选的奇偶校验位。奇偶校验位如果存在，也遵循反逻辑。校验位是检查错误的方法之一。事先指定传输的奇偶性。如果奇偶性选为奇性，那么设置奇偶校验位，使包括数据位和奇偶校验位在内的所有数位中，1 的个数合计为奇数。

一个字符帧的最后部分包含了 1、1.5 或者 2 个总是用负电压表示的停止位。如果没有其他字符传输进来，数据线就停留在负 (MARK) 状态。如果还有字符传输进来，那么下一个字符帧的传输就从正 (SPACE) 电压的起始位开始。

下图为字母 m 的典型字符帧编码。

图 6-13. 字母 m 的字符帧



RS-232 仅使用两个电压状态，分别为 MARK 和 SPACE。在两态编码机制中，波特率与每秒传输的最大位数信息一致（包括控制位信息）。

MARK 为负电压，SPACE 为正电压。上图为理想信号在示波器中的显示状态。RS-232 的真实图表如下：

Signal > +3 V = 0

Signal < -3 V = 1

输出信号电压通常在 +12 V 和 -12 V 间波动，+3 V 和 -3 V 之间的死区设计用于吸收数据线噪声。

起始位标识每个字符帧的开始。该标识为负电压 (MARK) 切换为正电压 (SPACE)。持续时间（秒）是波特率的倒数。如仪器的传输波特率为 9600，则起始位的持续时间及每个后续位时间为 0.104 ms。全部 11 位字符帧的传输时间约为 1.146 ms。

传输数据位的解析为 1101101（二进制）或 6D（十六进制）。ASCII 转换表显示该字母为 m。

传输使用奇校验。在数据位中有五个 1，已经为奇数。因此校验位被设置为 0。

数据传输速率

根据已知的通信设置，将波特率除以每个字符帧的位数，可以计算出每秒最大的字符传输率。

在上一个范例中，每个字符帧具有 11 个数据位。如传输速率设备为 9600 波特率，则每秒传输 $9600/11 = 872$ 个字符。注意，这是最大字符传输速率。终端硬件或另一串口链接可能由于多种原因无法达到上述速率。

串口标准

下列范例是串口通信中最常用的推荐标准：

- RS232 (ANSI/EIA-232 标准) 应用广泛，如连接鼠标、打印机或者调制解调器。它也应用于工业仪器中。由于线驱动程序和电缆的改进，应用程序常常能提高 RS232 的性能，并超越标准中列出的距离和速度。RS232 仅限于实现个人电脑的串口和设备之间的点对点连接。

- RS422 (AIA RS422A 标准) 使用的是一个差分电信号，而不是 RS232 中使用的不平衡（单端）接地信号。差分传输使用了两根线同时传输和接收信号，因此和 RS232 相比有更好的噪声抗扰度和更长的传输距离
- RS485 (EIA-485 标准) 是 RS422 的变体，允许将多达 32 个设备连到一个端口上，并定义了必要的电气特性以确保在最大负载时信号电压足够大。有了这种增强的多点传输功能，就可以创建连到一个 RS485 串口上的设备网络。对于那些需要将许多分布式仪器和一台个人电脑或其它控制器连网，从而完成数据采集和其它操作的工业应用程序来说，噪声抗扰度和多点传输功能使 RS485 成为一个有吸引力的选择。

仪器控制编程

VISA 是调用底层驱动程序的高级 API。VISA 可通过 GPIB、USB、以太网、串口和其他接口控制仪器，它根据使用的仪器类型的不同调用相应的驱动程序。调试 VISA 问题时，请注意 VISA 问题可能是由 VISA 调用的驱动程序的安装引起的。

在 LabVIEW 中，VISA 是一个函数库，用于与仪器提供的不同的 I/O 接口通信。因此无需使用独立的 I/O 选板编程仪器。例如，某些设备在一个仪器上提供多个接口类型（例如，以太网、USB 和 GPIB）。如 LabVIEW 仪器驱动是通过**仪器 I/O>GPIB** 选板上的函数编写的，则仪器驱动 VI 不能用于使用 USB 接口的仪器。VISA 通过提供一些用于不同类型接口的函数集，解决了上述问题。因此，许多 LabVIEW 仪器驱动使用 VISA 用作 I/O 语言。

VISA 编程术语

下列术语与用于仪器驱动 VI 的类似：

- **资源**—系统中仪器，包括串口和并口。
- **会话句柄**—必须打开一个 VISA 会话句柄，与资源通信。类似于信道。如打开资源的会话句柄，LabVIEW 将返回 VISA 的句柄编号，该编号为仪器的唯一引用句柄。在所有后续 VISA 函数中，必须使用该句柄编号。
- **仪器描述符**—资源的名称。描述符指定了接口的类型（GPIB、USB、TCP/IP 和 ASRL）、设备地址（逻辑地址或主地址）和 VISA 会话句柄类型（INSTR 或 Event）。

仪器描述符类似于电话号码，资源类似于要呼叫的用户，会话句柄类似于电话线缆。每个对话均使用专线，发生串扰将导致错误。表 6-1 为用于不同接口的仪器描述符的正规语法。

表 6-1. 用于不同仪器接口的语法

接口	语法
异步串行	ASRL[设备][:INSTR]
GPIB	GPIB[设备]::主地址[:次地址][:INSTR]
嵌入或 MXIbus 控制器的 VXI 仪器	VXI[设备]::VXI 逻辑地址[:INSTR]
GPIB-VXI 控制器	GPIB-VXI[设备][:GPIB-VXI 主地址]:VXI 逻辑地址[:INSTR]

在 MAX 中可使用别名替代设备描述符。

最常用的 VISA 通信函数为“VISA 写入”和“VISA 读取”函数。多数仪器需要用户以命令或查询格式发送信息，然后才能读取来自仪器的信息。因此，“VISA 写入”函数通常后跟“VISA 读取”函数。“VISA 写入”和“VISA 读取”函数适用于任何类型的仪器通信，且执行 GPIB 或串口通信时相同。但由于串行通信需要配置额外的参数，因此串口通信必须由“VISA 配置串口 VI”开始。

使用仪器驱动

假设下列场景。编写一个 LabVIEW VI，其与实验室中的指定示波器通信。但示波器已停止工作。且此型号的示波器已不再生产。目前可购买其他类型的示波器，但 VI 与新型示波器不匹配。用户必须重新编写 VI。

仪器驱动中包含专用于该仪器的代码。更换仪器时，仅需使用新仪器的驱动程序 VI 替换原有 VI，大大缩短了用户的二次研发时间。仪器驱动有助于测试应用程序的维护，因为驱动程序在一个库内包含了用于仪器的全部 I/O，且与其他代码分开。用户升级硬件时，升级应用程序也被简化了。因为仪器驱动包含用于该仪器的全部代码。

仪器驱动

LabVIEW 即插即用仪器驱动为控制可编程仪器的一组 VI。每个 VI 对应一项仪器操作。例如，配置、触发和读取仪器测量结果。仪器驱动帮助用户直接通过计算机与仪器交互。由于无需学习不同仪器的编程协议，大大节省了开发时间和成本。对于开源、具有良好注释的仪器驱动，用户可进行自定义以提供更好的性能。模块化设计简化了驱动程序的自定义。

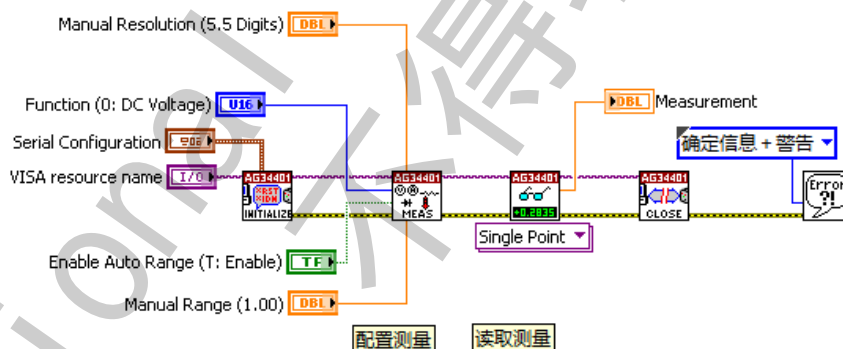
查找仪器驱动

通过仪器驱动查找器可找到绝大部分 LabVIEW 即插即用驱动程序。在 LabVIEW 中选择 **工具 » 仪器 » 查找仪器驱动** 或 **帮助 » 查找仪器驱动**。仪器驱动程序查找器可通过 NI 仪器驱动网 (ni.com\idnet) 下载仪器驱动。安装仪器驱动后，将添加一个使用该驱动的程序范例至 NI 范例查找器。

范例仪器驱动程序 VI

图 6-14 中的程序框图初始化 Agilent 34401 数字万用表 (DMM)，通过配置 VI 选择精度和量程、选择函数及启用或禁用范围自动。通过数据 VI 读取测量值、关闭仪器和检查错误状态。每个使用仪器驱动的应用均带有类似的事件序列：初始化、配置、读取数据和关闭。

图 6-14. Agilent 34401 DMM 仪器驱动范例



这是为 Agilent 34401 DMM 安装 LabVIEW 即插即用仪器驱动时，NI 范例查找器中出现的范例项目。

许多可编程仪器具有大量的函数和模式。由于其复杂性，提供一致的设计模型非常重要。它能帮助仪器驱动开发者和开发仪器控制应用的终端用户。

仪器驱动中的 VI 被分为 6 个类别。下表为类别的总结。

类别	说明
初始化	初始化 VI 建立与仪器的通信，且为最初调用的仪器驱动 VI。
配置	配置 VI 为配置仪器执行指定操作的程序集合。调用上述 VI 后，仪器已准备就绪执行测量或激励系统。
动作 / 状态	动作 / 状态 VI 命令仪器执行操作（例如，打开触发器）、获取仪器的当前状态或等待操作。
数据	数据 VI 将数据传输至仪器，或传输来自仪器的数据。
工具	工具 VI 执行一系列辅助操作。例如，重置或自测。
关闭	关闭 VI 终止软件至仪器的连接。这是被调用的最后一个仪器驱动。

自测：练习

1. 打开文件后，下列哪个输出为“打开 / 创建 / 替换文件”函数的返回值？
 - a. 文件路径
 - b. 文件名
 - c. 引用句柄输出
 - d. 任务输出

2. 什么是 Measurement & Automation Explorer (MAX)？
 - a. 配置和测试 DAQ 设备的工具
 - b. 测试仪器通信的工具
 - c. 可配置 Express VI
 - d. 查看 LabVIEW 项目文件的窗口

3. 下列哪项或哪几项为仪器控制的优势？
 - a. 过程自动化
 - b. 节省时间
 - c. 在一个平台上执行多个任务
 - d. 仅限于一种仪器类型

4. VISA 是一个调用低层驱动程序的高层 API。
 - a. 对
 - b. 错

International Instruments
不得转载

自测：练习答案

1. 打开文件后，下列哪个输出为“打开 / 创建 / 替换文件”函数的返回值？
 - a. 文件路径
 - b. 文件名
 - c. **引用句柄输出**
 - d. 任务输出

2. 什么是 Measurement & Automation Explorer (MAX)？
 - a. **配置和测试 DAQ 设备的工具**
 - b. **测试仪器通信的工具**
 - c. 可配置 Express VI
 - d. 查看 LabVIEW 项目文件的窗口

3. 下列哪项或哪几项为仪器控制的优势？
 - a. **过程自动化**
 - b. **节省时间**
 - c. **在一个平台上执行多个任务**
 - d. 仅限于一种仪器类型

4. VISA 是一个调用低层驱动程序的高层 API。
 - a. **对**
 - b. 错

笔记

ational Instruments
不得转载

使用顺序和状态机算法

开发 LabVIEW 应用的第一步是了解 LabVIEW 的架构。架构对于成功创建软件设计非常重要。最常见的架构通常被分类为设计模式。

由于设计模式被认可，当使用设计模式时较易被识别。这有助于您和其他开发者读取和修改基于设计模式的 VI。

LabVIEW VI 具有多种设计模式。多数应用使用至少一种设计模式。在本课程中，您将学习状态机设计模式。更多设计模式的信息，见 *LabVIEW 核心教程*（二）。

主题

- A. 顺序编程
- B. 状态编程
- C. 状态机

A. 顺序编程

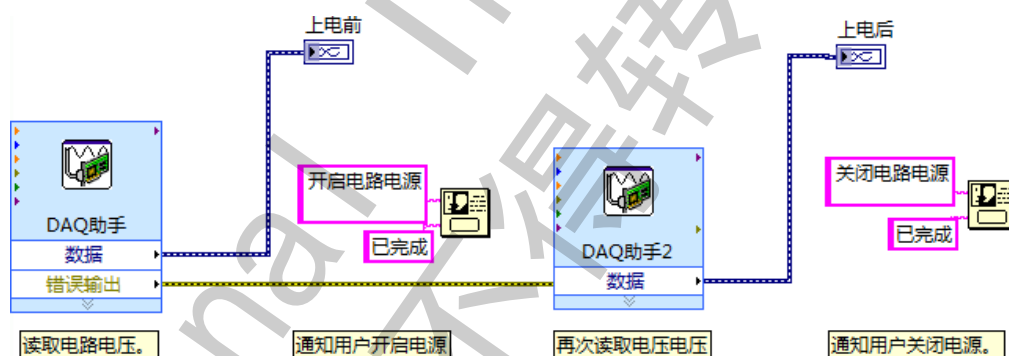
多数用户在 LabVIEW 中编写的 VI 为顺序执行任务。编程顺序任务的方式各不相同。按照图 7-1 创建程序框图。在该程序框图中采集电压信号，弹出对话框提示用户开启电源，然后再次采集电压信号，并提示用户关闭电源。但在本范例中，程序框图中不包含强制上述事件执行的项。上述任何事件均可能最先发生。

图 7-1. 非顺序任务



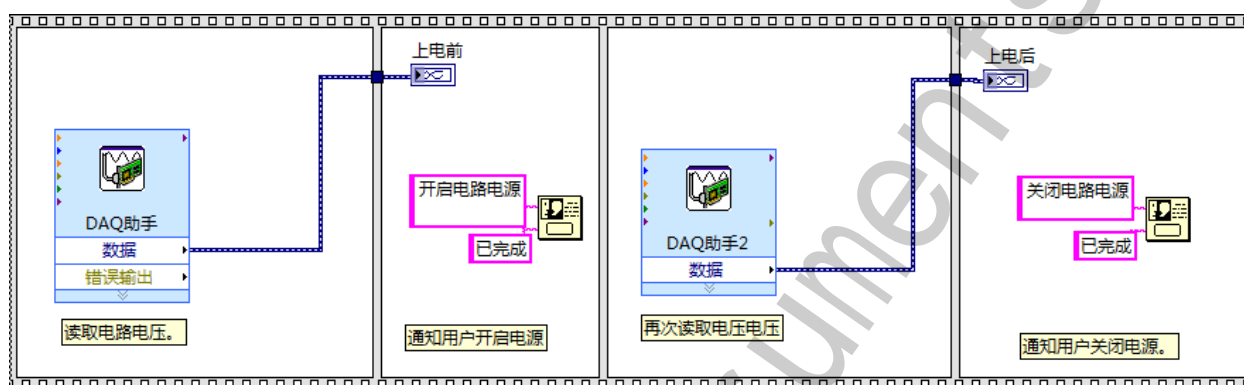
在 LabVIEW 中，可通过将每个任务放置在独立的子 VI 中，并按照所需的顺序连线这些子 VI 并连接错误簇。但在该范例中仅有两个任务带有错误簇。使用错误簇，用户可强制规定两个 DAQ 助手的执行顺序，但不包括“单按钮对话框”函数。如图 7-2 所示。

图 7-2. 部分顺序任务



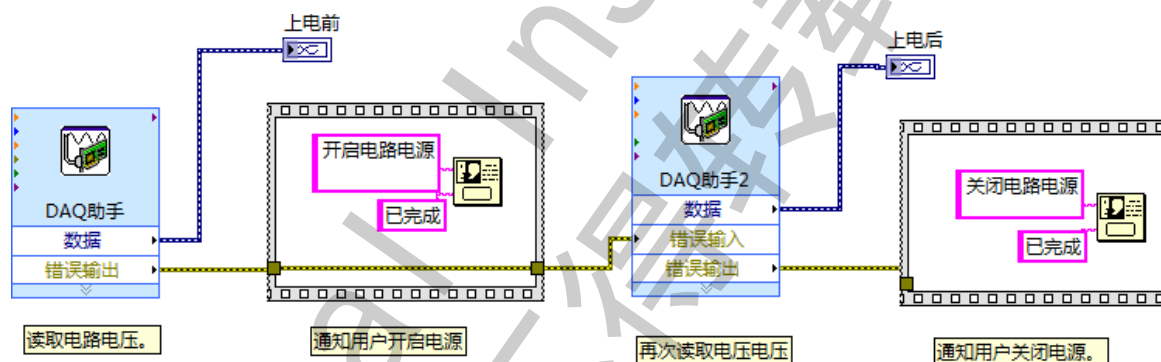
使用顺序结构强制规定程序框图对象的执行顺序。一个顺序结构中包括一个或多个子框图或顺序执行的帧；前一个帧执行结束后，下一个帧才能开始执行。图 7-3 为使用顺序结构强制规定执行顺序的 VI 的范例。

图 7-3. 使用顺序结构的顺序任务



如要充分利用 LabVIEW 固有的**并行机制**，应尽量避免使用过量使用顺序结构。顺序结构保证了执行的顺序，但也禁止了并行操作。使用顺序结构的另一个缺点是，用户不能在顺序执行过程中停止顺序操作。使用该范例的较好范例如图 7-4 所示。

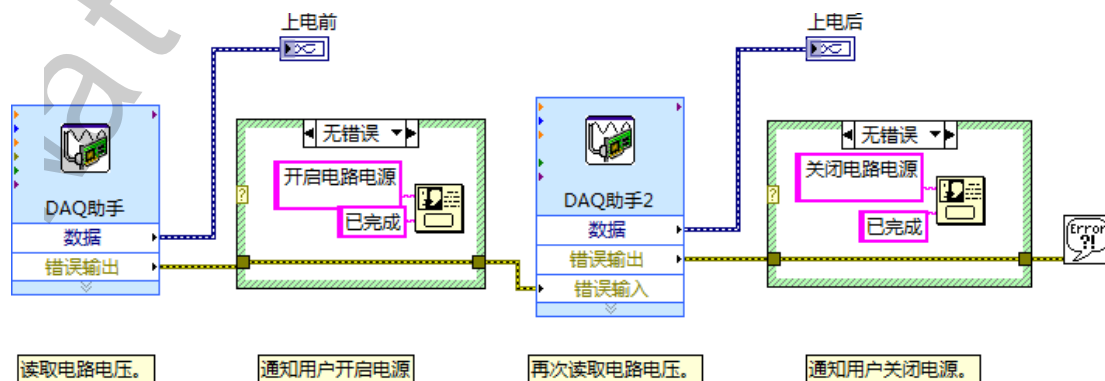
图 7-4. 使用顺序结构和错误簇的顺序任务



编写该 VI 的最佳方法是将“单按钮对话框”函数嵌入在条件结构内，并连线错误簇至该条件分支。

有控制性的使用顺序结构，因为它们不会强制错误检查，且即使检测到错误的情况下，仍会继续执行顺序结构。因此应该使用数据流（而不是顺序结构）控制执行顺序。

图 7-5. 使用错误簇和条件结构的顺序结构



B. 状态编程

尽管顺序结构和顺序连线子 VI 能够实现任务，但 VI 通常需要更复杂的编程：

- 必须更改执行顺序时
- 需重复执行帧中的某一项时
- 需在满足特定条件时才执行某几帧时
- 需立即停止程序，而不是等待最后一个帧执行完毕才结束程序时

尽管当前程序可能无需满足上述要求，但后续存在更改程序的可能性。基于上述原因，即使顺序编程架构足以满足编程需求，状态编程架构仍为较好的选择。

C. 状态机

状态机设计模式是用于 LabVIEW 的常用及非常有用的设计模式。状态机设计模式可用于执行通过状态图或流程表可以显性描述的算法。状态机通常执行适度复杂的决策算法。例如，诊断程序或进程监控。

状态机（更精确的定义是有限状态机）是由状态的集合以及对应状态切换的转换函数构成。有限状态机具有许多变化。两种最常见的有限状态机为米利型 (Mealy) 状态机和摩尔型 (Moore) 状态机。米利型状态机是对每次转换产生响应。而摩尔型状态机是对状态图中的每一个状态产生一个响应。LabVIEW 中的状态机设计模式模板实现由摩尔型状态机描述的任意算法。

应用状态机

在存在显著不同状态的应用中使用状态机。每个状态可触发一个或多个状态或结束进程。状态机依赖于用户输入或状态内的计算判定下一个状态。许多应用需要一个初始化状态，后跟默认状态。默认状态可执行多种不同的操作。执行的操作取决于上一个状态及当前输入和状态。关闭状态通常执行清除操作。

状态机通常用于创建用户界面。在用户界面，不同的用户操作对应不同的用户界面。每个处理过程表现为状态机的一个状态。每个进程可触发另一个状态，状态可为进一步操作或等待用户操作。在用户界面应用中，状态机不断监视用户，以选择下一个要执行的动作。

进程测试是状态机设计模式另一个常见的应用。对于进程测试，每个状态表示一段进程。根据每个状态测试的结果，调用不同的状态。这将连续的发生，进而对测试进程执行深入的分析。

使用状态机的优势在于创建状态转换图后，简化了创建 LabVIEW VI 的过程。

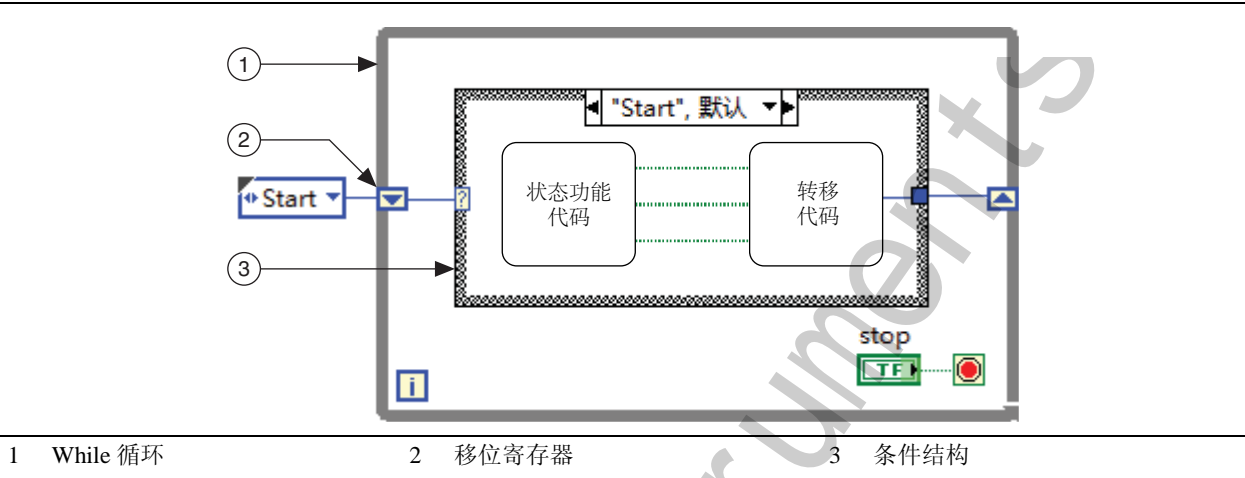
状态机结构

将状态转换图转换为 LabVIEW 程序框图需要下列结构组件：

- **While 循环**—连续执行不同的状态
- **条件结构**—包含用于每个状态的结构分支及状态执行代码
- **移位寄存器**—包含状态转换信息
- **状态功能代码**—实现状态的功能
- **转换代码**—判定下一个状态

图 7-6 为在 LabVIEW 中实现的状态机的基本结构，该状态机用于温度数据采集系统。

图 7-6. LabVIEW 状态机的基本结构



状态转换图的流是通过 While 循环实现的。每个状态是通过条件结构的分支表示的。While 循环中的移位寄存器跟踪当前状态，并将当前状态传输至条件结构输入。

控制状态机

控制状态机初始化和转换的最佳方法是枚举型控件。在状态机中，枚举普遍用于条件选择器。但如用户尝试为枚举型控件添加或删除状态，连接至枚举型控件副本的连线将断开。这是使用枚举型控件实现状态机的常见问题。该问题的解决方案是创建一个自定义类型枚举型控件。此时如添加或移除状态，枚举型控件的副本将进行自动更新。

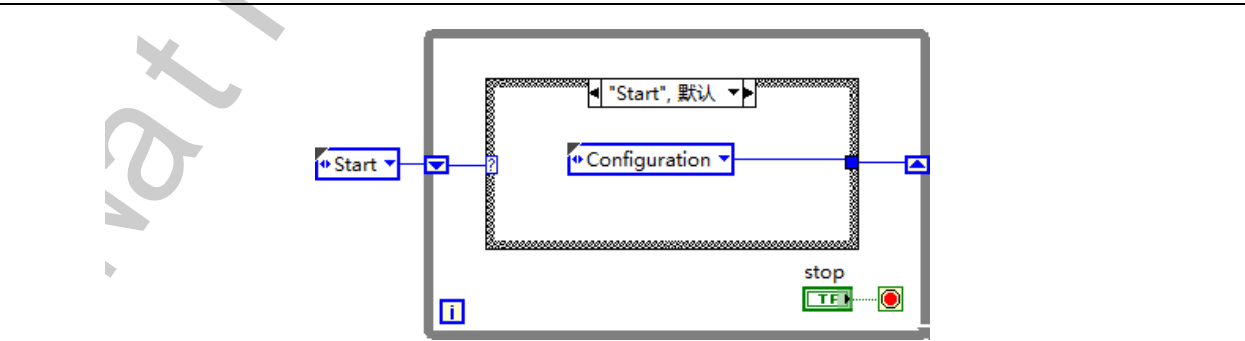
状态机转换

在状态机内可通过多种方法控制条件结构执行的分支。选择最适合函数和状态机复杂度的方法。在实现状态机转换的方法中，最常见和易用的方法是条件结构转换代码。它可用于任意数量状态间的转换。该方法提供了最具扩展性、可读性和可维护性的状态机架构。其他方法适用于指定的情况，熟悉它们也非常重要。

默认转换

对于默认转换，无需代码判定下一个状态。因为接下来仅有一个可能发生的状态。图 7-7 为使用默认转换实现温度数据采集系统的设计模式。

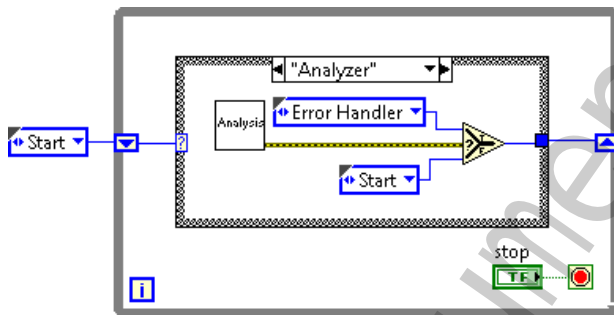
图 7-7. 单个默认转换



状态间转换

下列方法包含在两个状态间转换的选择。通常有几种方式实现上述操作。图 7-8 为用于在两个状态间转换的“选择”函数。

图 7-8. “选择”函数转换代码



本方法适用于已知单个状态在两个状态间的转换趋势。但本方法也限制了应用的可扩展性。如需更改状态使其在多于两个状态间转换，该解决方案不适用且需要对转换代码进行重要更改。

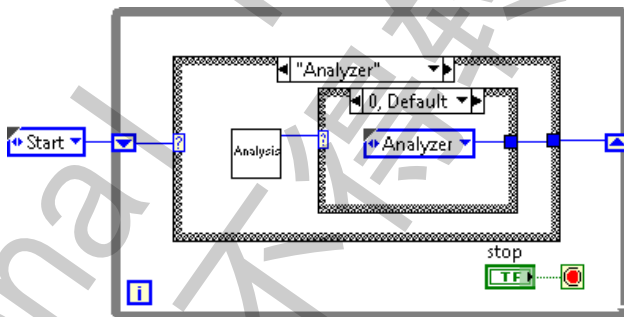
两个或多个状态间的转换

使用下列方法创建一个扩展性更好的架构，以实现在多态间转换。

- **条件结构**—使用条件结构替换转换代码中的“选择”函数。

图 7-9 为用于实现温度数据采集系统的条件结构转换。

图 7-9. 条件结构转换代码

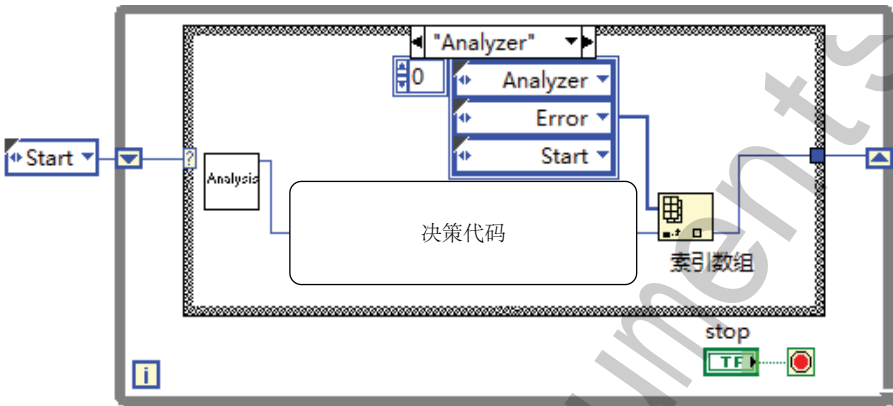


使用条件结构的一个优势是代码更易理解。由于条件结构中的每一个分支对应于枚举的一个项，有助于读取和理解代码。条件结构也是可扩展的。随着应用程序的扩展，为某状态的条件结构添加更多的分支，可为该状态添加更多的转换。使用条件结构的劣势是不能同时显示全部代码。由于条件结构的特性，不能看一眼就了解转换代码的全部功能。

- **转换数组**—如需要显示比条件结构允许更多的代码，可为所有待发生的转换创建一个转换数组。

图 7-10 为用于实现温度数据采集系统的转换数组。

图 7-10. 转换数组转换代码



在本范例中，决策代码提供了描述下一个状态的索引。例如，如代码将进入“错误处理器”状态，决策代码输出数值 1 至“索引数组”函数的索引输入端。该设计模式使转换代码具有可扩展性且方便阅读。该模式的一个劣势是开发转换代码时，用户必须使用警告。因为数组是零索引的。

实例分析：课程项目

课程项目每半秒采集一次温度数据，并对温度值进行分析。当温度值过高或过低时向用户发出中暑警告或冷冻警告。程序在发生警告时记录数据。用户未停止程序的情况下，重复执行上述过程。图 7-11 为课程项目的状态转换图。

图 7-11. 课程项目的状态转换图

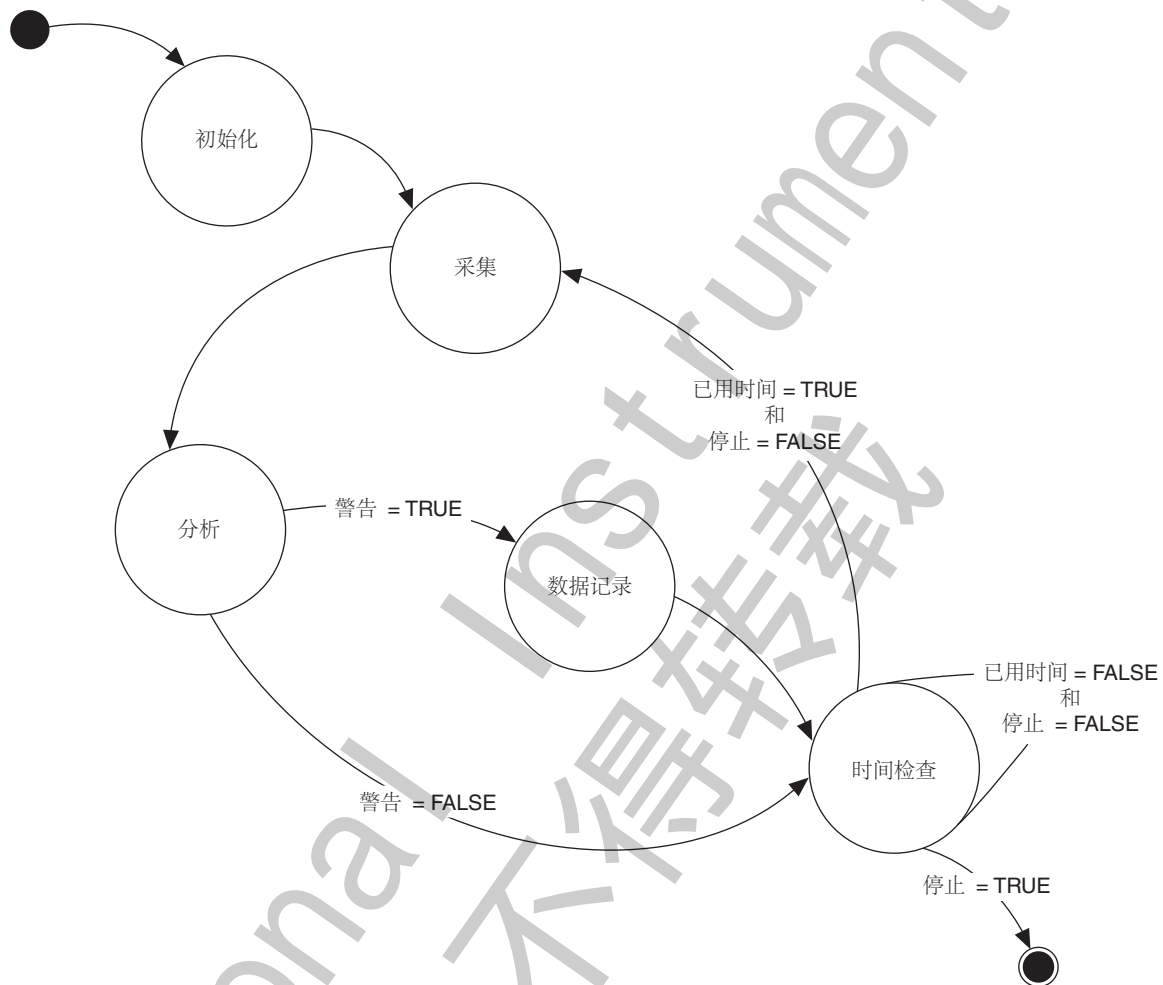


图 7-12 至 7-15 为 7-11 中的状态转换框图的状态机的状态实现。

图 7-12. 采集数据

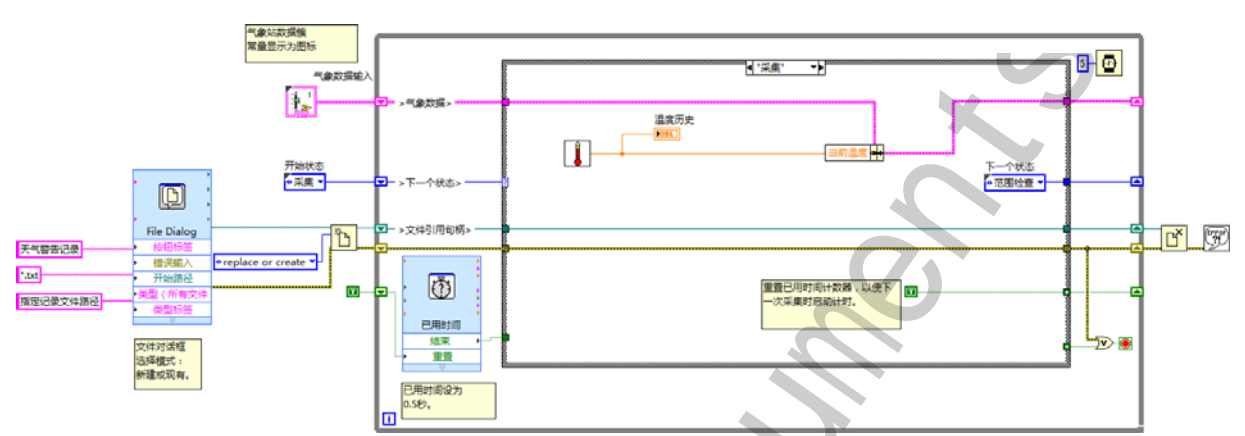


图 7-13. 分析状态

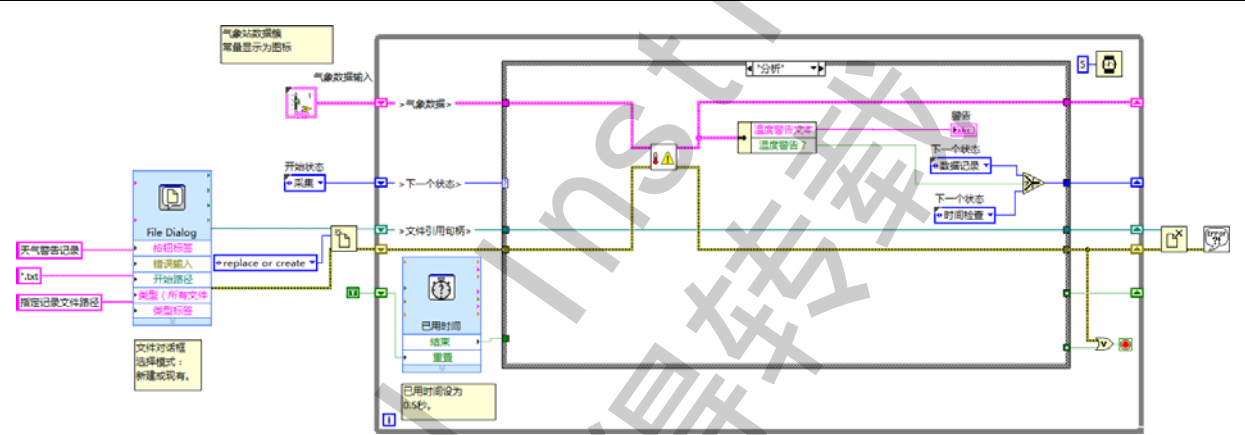


图 7-14. 数据记录状态

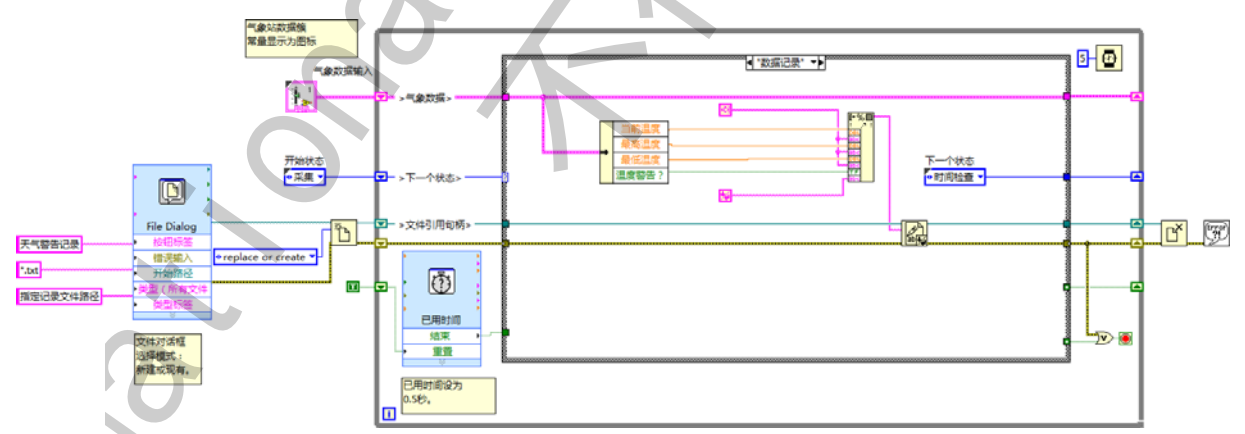
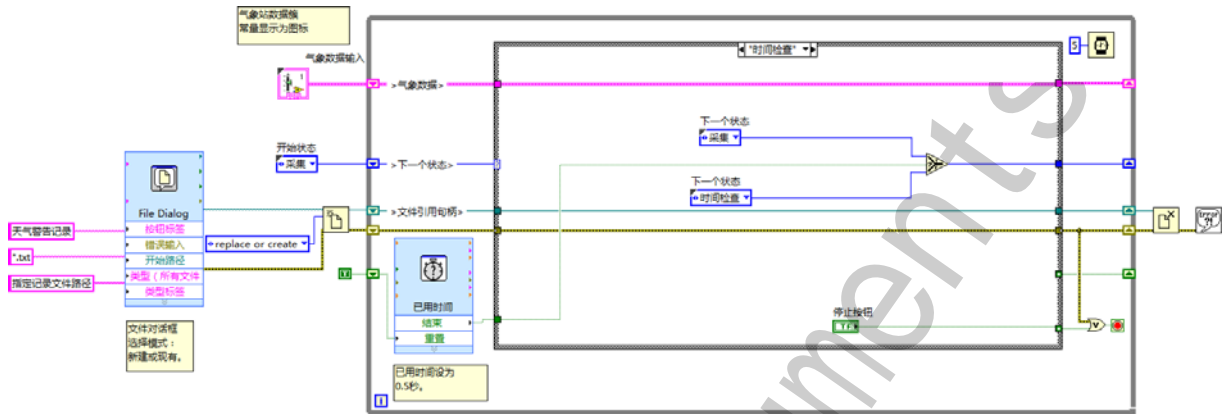


图 7-15. 检查时间状态



自测：练习

1. 使用顺序结构时，可在顺序执行过程中中断执行。
 - a. 对
 - b. 错

2. 下列哪项或哪几项为状态机相对于顺序结构的优势？
 - a. 可更改执行顺序。
 - b. 可重复执行序列中某一项。
 - c. 可设置执行顺序结构中某项的判定条件。
 - d. 可在程序执行的任意阶段中止程序。

International Instruments
不得转载

自测：练习答案

1. 使用顺序结构时，可在顺序执行过程中中断执行。
 - a. 对
 - b. 错**

2. 下列哪项或哪几项为状态机相对于顺序结构的优势？
 - a. 可更改执行顺序。**
 - b. 可重复执行序列中某一项。**
 - c. 可设置执行顺序结构中某项的判定条件。**
 - d. 可在程序执行的任意阶段中止程序。**

笔记

vat i onal I n s t r u m e n t s
不得转载

通过变量解决数据流问题

LabVIEW 是数据流语言。因此通常使用数据流设计原理开发代码。但在几种情况下（例如，在并行循环间通信），必须“打破”数据流执行模型，在不能连线的代码位置传输信息。

本课程将介绍变量。通常，局部变量主要处理数据流执行模型不能处理的特殊分支。您还将学习使用变量编程及解决变量编程的问题。

由于变量不属于 LabVIEW 数据流执行模型中的固有部分，请谨慎使用变量。错误使用变量将增加代码的阅读难度，导致 VI 的未预期行为并降低性能。

主题

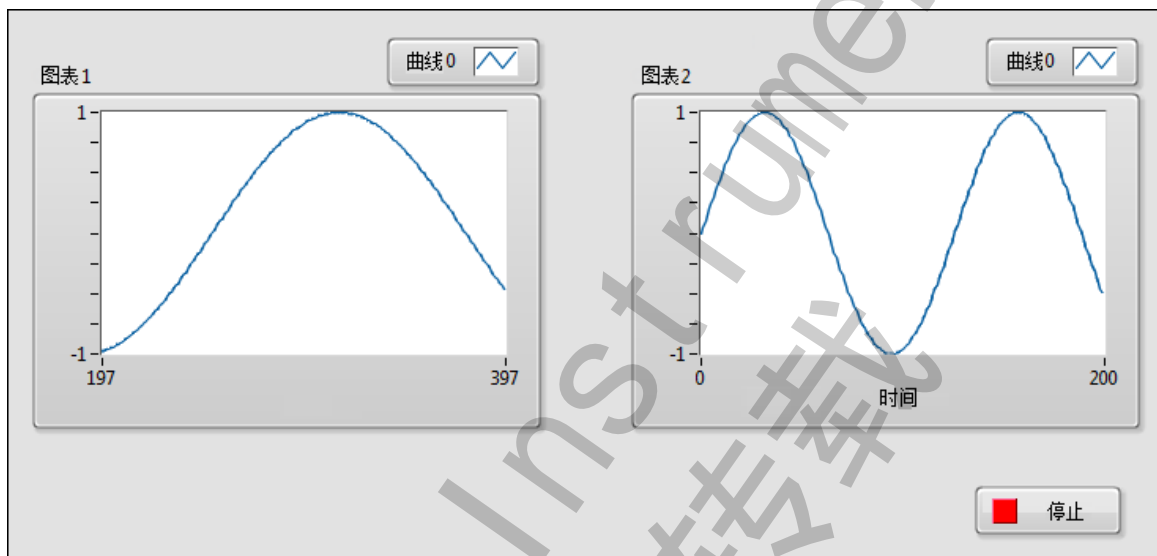
- A. 并行循环间通信
- B. 写入输入控件和读取显示控件
- C. 变量
- D. 竞争状态

A. 并行循环间通信

在本课程中，并行机制是指同时执行多个任务。考虑创建和显示两个不同频率的正弦波的范例。使用并行机制，将两个正弦波分别放置在不同的循环内部。

编程并行任务的一个难点是在不创建数据依赖性的情况下，在多个循环间传递数据。如通过连线传输数据，则循环不再是并行机制。在多个正弦波范例中，可能需要在循环间共享一个停止机制。如图 8-1 所示。

图 8-1. 并行循环的前面板

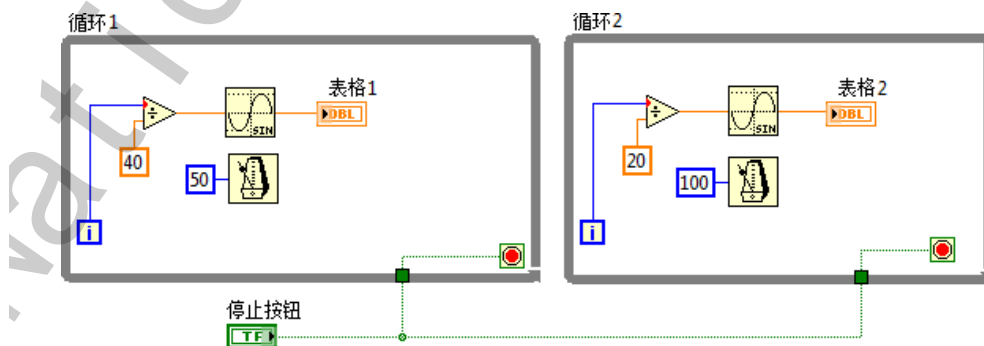


观察使用不同的方法通过连线在并行循环间分享数据的情况。

方法 1（不正确）

在两个循环的外部放置**停止按钮**接线端，并将其连线至条件接线端。如图 8-2 所示。循环控件是两个循环的数据输入，因此**停止按钮**接线端在任意 While 循环开始执行前，仅读取一次。如 False 被传递至循环，While 循环无限运行。单击**停止按钮**不会停止 VI，因为在循环计数期间不会读取单击按钮的操作。

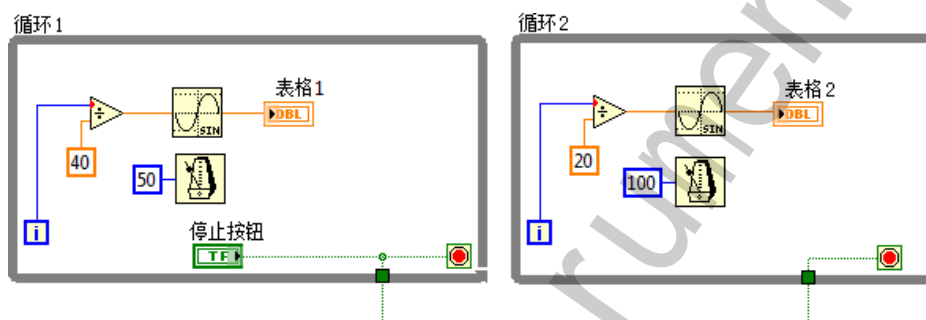
图 8-2. 并行循环方法 1 范例



方法 2（不正确）

移动**停止按钮**接线端至循环 1 的内部，即每次循环计数都会读取该接线端。如下列程序框图所示。尽管循环 1 正确中止，循环 2 不会执行，直至接收到循环 1 的全部数据输入。循环停止前，循环 1 不会将数据传出循环。因此循环 2 必须等待**停止按钮**的最终值，该值仅在循环 1 结束后才能得到。因此循环不是并行执行的。且循环 2 仅执行一个循环计数，因为其条件接线端从循环 1 的**停止按钮**端接收到一个“真”值。

图 8-3. 并行循环方法 2 范例



方法 3（解决方案）

如果能通过文件读取循环控件的值，则循环间无需依赖于数据流。因为每个循环均可独立访问该文件。此外，读取和写入文件将占用时间（至少占用处理器时间）。

另一个完成并行循环通信任务的方法是找到循环控制数据在存储器内的存储位置，并直接读取该存储地址。局部变量允许用户通过读取同一个前面板控件访问程序框图中的不同位置。在本课程的后半部分，您将学习到创建和使用局部变量，并通过局部变量在两个并行循环间通信。

B. 写入输入控件和读取显示控件

除在两个同步运行的循环间通信外，局部变量还可以解决其他数据流问题。

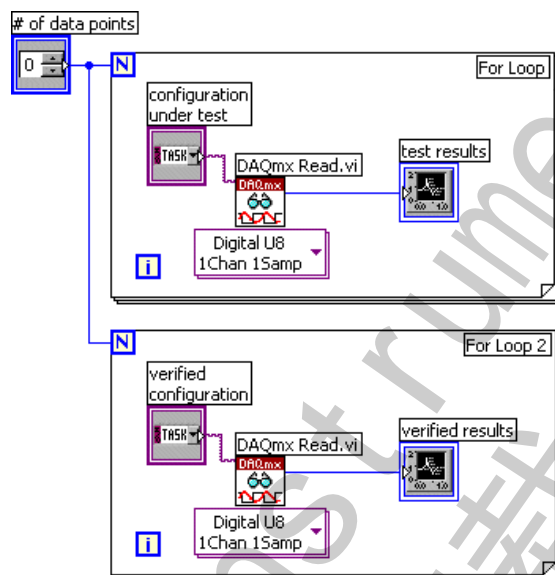
通过局部变量可更新位于不同程序框图位置的前面板显示控件。例如，VI 具有两个 While 循环，更新前面板显示控件可显示当前运行的循环。

通过局部变量，可写入或读取前面板输入控件或显示控件。实际上，使用局部变量可将前面板对象同时用作输入和输出。例如，如用户界面要求用户执行登录操作，每次新用户登录时可清除**登录**和**密码**提示。局部变量在用户登录时读取**登录**和**密码**字符串控件，用户登出时向上述控件写入空白字符串。

C. 变量

在 LabVIEW 中，通过数据流而不是顺序命令判定程序框图元素的执行顺序。因此可创建执行同步操作的程序框图。例如，可同步运行两个 For 循环并在前面板上显示结果。如图 8-4 所示。

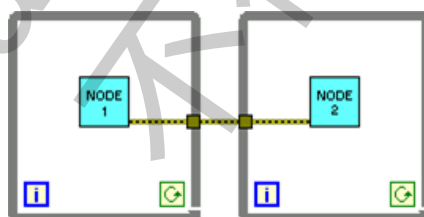
图 8-4. 在前面板上显示两个 For 循环的结果



在 LabVIEW 中，通过数据流而不是顺序命令判定程序框图元素的执行顺序。因此可创建执行同步操作的程序框图。例如，可同步运行两个 For 循环并在前面板上显示结果。如下列程序框图所示。但如果通过连线在并行循环间传递数据，则它们不能执行并行操作。并行循环可以是同一程序框图上不具有数据依赖性的两个并行循环，或同时被调用的独立 VI 中的两个循环。

由于两个子 VI 的连线，图 8-5 中的程序框图不会并行运行两个循环。

图 8-5. 连线的数据依赖性



连线构建了数据依赖性，因为第二个循环在第一个循环执行结束，并通过隧道传递数据前不会开始。如要使两个循环同时运行，需移除连线。如要在子 VI 间传输数据，可使用其他技术。例如，变量。

在 LabVIEW 中，变量为程序框图元素，用户可使用变量在其他位置存储或访问数据。数据的实际位置随变量的类型改变。局部变量在前面板输入控件和显示控件中存储数据。全局变量和单进程共享变量将数据存储在特殊库中，可通过多个 VI 访问。功能全局变量在 While 循环的移位寄存器中存储数据。无论变量在何处存储数据，所有变量均允许用户突破常规数据流的限制，不通过连线在两个不同地址间传递数据。因此，变量在并行架构中非常重要。但同时也具有一些缺陷（例如，竞争状态）。

局部变量—在 VI 中使用变量

局部变量在单个 VI 中传输数据。

在 LabVIEW 中，使用程序框图接线端从前面板对象读取数据或写入数据至前面板对象。但每个前面板对象仅具有一个程序框图接线端，应用可能需要在不同位置访问该接线端数据。

局部变量和全局变量在不能通过连线传递数据的应用程序位置间传输信息。通过局部变量访问位于同一 VI 的不同位置的前面板对象。通过全局变量访问并在不同的 VI 间传输数据。

使用反馈节点存储前一 VI 或循环执行的数据。

创建局部变量

右键单击现有的前面板对象或程序框图接线端，从快捷菜单中选择**创建»局部变量**创建局部变量。该对象的局部变量图标将出现在程序框图上。

在**函数**选板上也可选择局部变量，并将其放置在程序框图上。局部变量节点未连接任何输入控件或显示控件。



如要将局部变量与控件关联，可右键单击局部变量，从快捷菜单中选择**选择项**。展开的快捷菜单中将列出全部具有自带标签的前面板对象。

LabVIEW 通过标签关联前面板对象和局部变量，因此需要给前面板控件命名描述性的自带标签。

读取和写入变量

创建变量后，可通过变量读取数据，或向其写入数据。默认状态下，新的变量接收数据。此类变量就像一个显示控件，同时是一个写入局部变量或写入全局变量。将新数据写入该局部或全局变量时，与之相关联的前面板输入控件或显示控件将由于新数据而被更新。

可将变量配置为数据源。右键单击变量，从快捷菜单中选择**转换为读取**，便可将该变量配置为一个输入控件。节点执行时，VI 将读取相关前面板输入控件或显示控件中的数据。

如需使变量从程序框图接收数据而不是提供数据，可右键单击该变量并从快捷菜单中选择**转换为写入**。

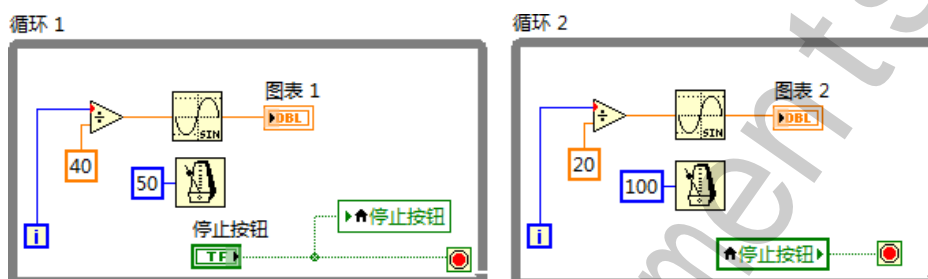
在程序框图上，读取变量与写入变量间的区别相当于输入控件和显示控件间的区别。与输入控件类似，读取变量的边框较粗。与显示控件类似，写入变量的边框较细。

局部变量范例

在本课程的“并行循环间通信”章节，将介绍使用并行循环的 VI 范例。前面板包含一个开关，其用于停止两个图形的数据生成显示。在程序框图上，每个图标数据是在独立的 **While** 循环内生成的，以允许循环独立定时。循环总数接线端将停止两个 **While** 循环。在本范例中，两个循环必须共享一个开关，以同时停止两个循环。

为使两个表格按预期更新，While 循环必须并行操作。在 While 循环间连线，以传递**停止按钮**数据。实现 While 循环顺序操作，而不是并行操作。图 8-6 为 VI 使用局部变量传递**停止按钮**数据的程序框图。

图 8-6. 用于停止并行循环的局部变量



循环 2 读取与**停止按钮**关联的局部变量。如在前面板上设置按钮为 False，循环 1 的接线端将写入 False 值至**停止按钮**局部变量及循环 1 的条件接线端。循环 2 读取**停止按钮**局部变量，并将 False 值写入至循环 2 条件接线端。因此，循环将并行运行，且当用户关闭单个前面板开关时，循环可同时终止。

局部变量可对前面板上的输入控件或显示件进行数据读写。写入局部变量相当于传递数据至其它接线端。但是，局部变量还可向输入控件写入数据和从显示控件读取数据。通过局部变量，前面板对象既可作为输入访问也可作为输出访问。

例如，如果用户界面需要用户登录，可在每次新用户登录时清空**登录**和**密码**提示框中的内容。通过局部变量，当用户登录时从**登录**和**密码**字符串控件中读取数据，当用户离开时向这些控件写入空字符串。

全局和共享变量—在 VI 间使用变量

变量还可用于在同时运行的 VI 间访问和传输数据。局部变量在 VI 间共享数据。全局变量也能够共享数据，但其在多个 VI 间共享数据。例如，假设有两个同步运行的 VI。每个 VI 含有一个 While 循环并将数据点写入一个波形图表。第一个 VI 包含一个用于终止这两个 VI 的布尔控件。使用全局变量可通过一个布尔控件终止两个循环。如这两个循环位于同一 VI 的程序框图上，可用一个局部变量来终止这两个循环。

单进程共享变量的使用方式与全局变量类似。共享变量与局部变量或全局变量类似，但可在网络间共享数据。共享变量可以是单进程的或网络发布的。尽管网络发布的共享变量在本课程中未提及。使用单进程共享变量，后续可将其更改为网络发布共享变量。

使用全局变量在同一计算机的 VI 间共享数据，尤其是在没有使用项目文件的情况下。如后续需要在不同计算机上的 VI 间共享变量信息，可使用单进程共享变量。

创建全局变量

使用全局变量在同时运行的不同 VI 间访问和传输数据。全局变量是内置的 LabVIEW 对象。创建全局变量时，LabVIEW 将自动创建一个有前面板但无程序框图的特殊全局 VI。添加输入控件和显示控件至全局 VI 的前面板，以定义其中包含的全局变量的数据类型。该前面板实际便成为一个可供多个 VI 进行数据访问的容器。

例如，假设现有 2 个同时运行的 VI。每个 VI 含有一个 While 循环并将数据点写入一个波形图表。第一个 VI 含有一个布尔控件来终止这两个 VI。此时须用全局变量通过一个布尔控件将这两个循环终止。如这两个循环位于同一 VI 的程序框图上，可用一个局部变量来终止这两个循环。

从**函数**选板上选择一个全局变量，将其放置在程序框图上。



双击该全局变量节点可显示全局 VI 的前面板。该前面板与标准前面板一样，可放置输入控件和显示控件。

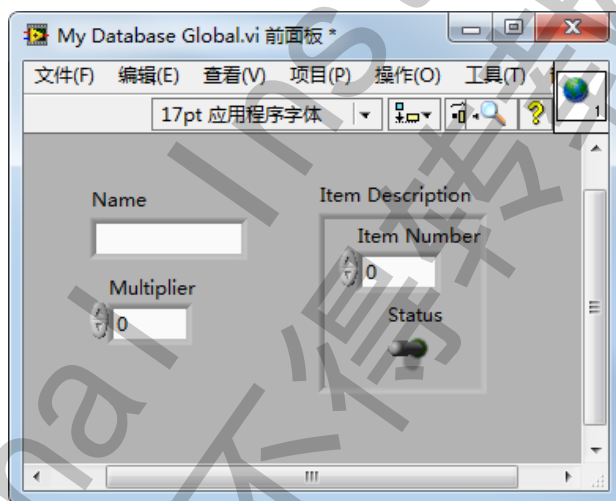
LabVIEW 使用自带标签区分全局变量，因此前面板的输入控件和显示控件的自带标签应带有描述性。

可创建几个仅含有一个前面板对象的全局 VI，也可创建一个含有多个前面板对象的全局 VI，从而将相似的变量归为一组。

带有多个对象的全局 VI 更为高效，因为用户可将相关的变量组合在一起。VI 的程序框图可包含几个全局变量节点，该节点与全局 VI 中的输入控件和显示控件关联。将全局 VI 放置在其他 VI 的方式与在放置子 VI 类似。每次在程序框图上放置一个新的全局变量节点，LabVIEW 将新建一个仅与该全局变量节点关联的全局 VI 及其副本。

图 8-7 为全局 VI 的前面板窗口，其中包括数值、字符串和包含数值和字符串控件的簇。工具栏上不会显示**运行**、**停止**或其他相关按钮。这与常规前面板窗口不同。

图 8-7. 全局 VI 前面板窗口



所有对象在全局 VI 前面板上放置完毕后，保存该全局 VI 并返回到原始 VI 的程序框图。然后必须选择全局 VI 中想要访问的对象。右键单击该全局变量节点并从快捷菜单中选中的一个前面板对象。该快捷菜单列出了全局 VI 中所有自带标签的前面板对象。右键单击该全局变量节点并从**选择项**快捷菜单中选择一个前面板对象。

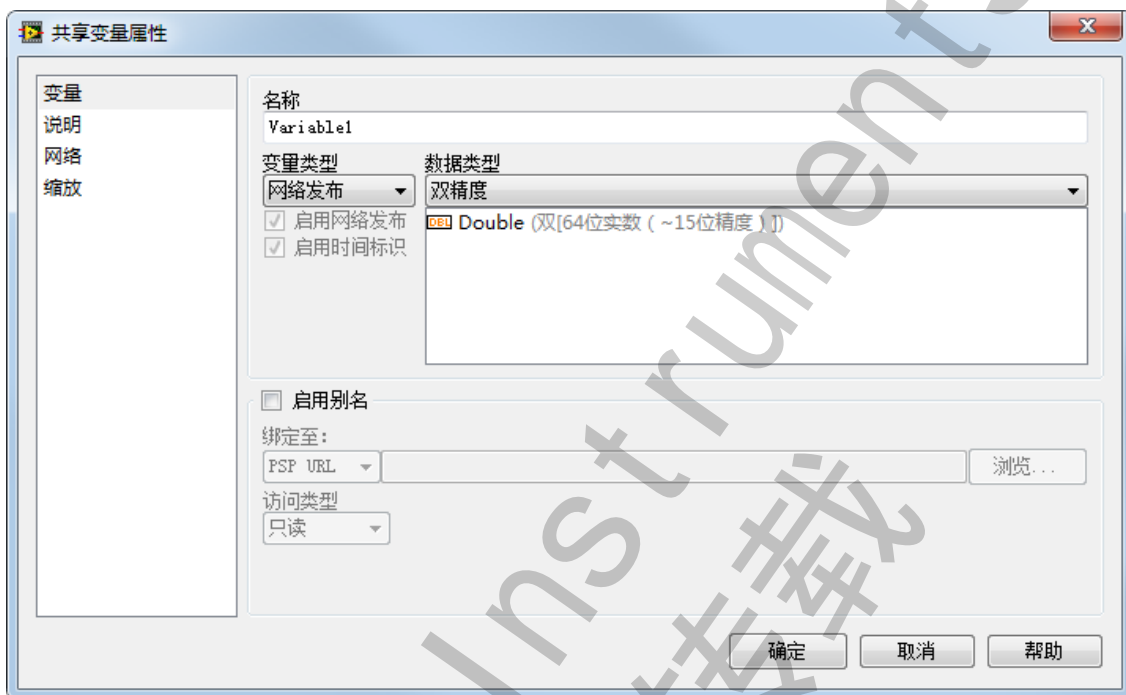
也可用操作工具或标签工具单击全局变量节点，从快捷菜单上选择前面板对象来完成这一步。

如要在其他 VI 中使用该全局变量，请在**函数**选板中选择**选择 VI**选项。默认状态下，全局变量与具有全局 VI 中自带标签的第一个前面板对象相关联。右键单击程序框图上的全局变量节点，从快捷菜单中选择**选择项**，从而将全局变量与另一个前面板对象实现关联。

创建单进程共享变量

创建共享变量之前必须先打开一个项目。如要创建一个单进程共享变量，在**项目浏览器**窗口右键单击**我的电脑**，并选择**新建»变量**。此时出现**共享变量属性**对话框。如图 8-8 所示。

图 8-8. 共享变量属性对话框



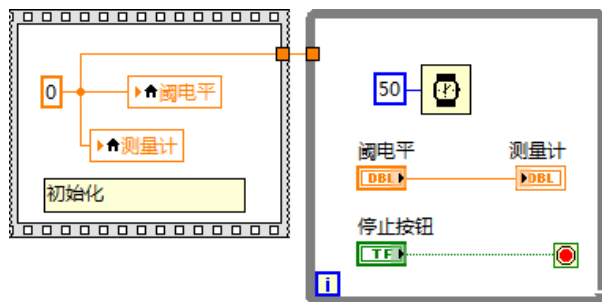
在**变量类型**下选择**单进程**。为变量命名并选择数据类型。创建共享变量后，它将自动出现在项目文件的库内。保存库。需要的情况下，可添加额外的共享变量至该库。可直接拖拽**项目浏览器**窗口中列出的变量至程序框图。使用快捷菜单在写入或读取间切换。使用变量的错误簇限定数据流。

初始化变量

如需对一个本地或全局变量进行初始化，应在 VI 运行前确认变量包含的是已知的数据值。否则变量可能含有导致 VI 发生错误行为的数据。如变量的初始值基于一个计算结果，则应确保 LabVIEW 在读取该变量前先将初始值写入变量。将写入操作与 VI 的其他部分并行可能导致竞争状态。

要使写入操作率先执行，可将写入初始值至变量这部分代码单独放在“顺序结构”的首帧。也可将这部分代码放在一个子 VI 中，通过连线使该子 VI 在程序框图的数据流中第一个执行。如图 8-9 所示。

图 8-9. 初始化变量



如在 VI 第一次读取变量之前，没有将变量初始化，则变量含有的是相应的前面板对象的默认值。

避免变量的固有编程问题

局部和全局变量是高级的 LabVIEW 概念。它们不是 LabVIEW 数据流执行模型中固有的部分。与变量相关的编程问题包含下列：

- 增加了程序框图的阅读难度—由于变量打破了数据流模型，不能通过连线跟踪数据流。
- 导致 VI 未预期的动作—使用变量而不是连线板，或使用变量访问顺序结构中每个帧的值是不良的操作习惯，可导致 VI 未预期的动作。
- 性能降低—过度使用局部和全局变量（例如，使用变量以避免程序框图间的较长连线，或使用变量替代数据流）将降低系统性能。

图 8-10 和图 8-11 解释了编程同一范例的两种方法。图 8-10 为变量使用的较差设计方案，图 8-11 为优化设计方案。编程范例为通过状态机实现的交通灯应用。每个状态更新交通灯序列的下一个信号状态。

在图 8-10 所示的状态中，东西信号灯为绿色，南北信号灯为红色。如“等待 (ms)”函数所示，本阶段等待 4 秒。

图 8-10. 使用了过多的变量

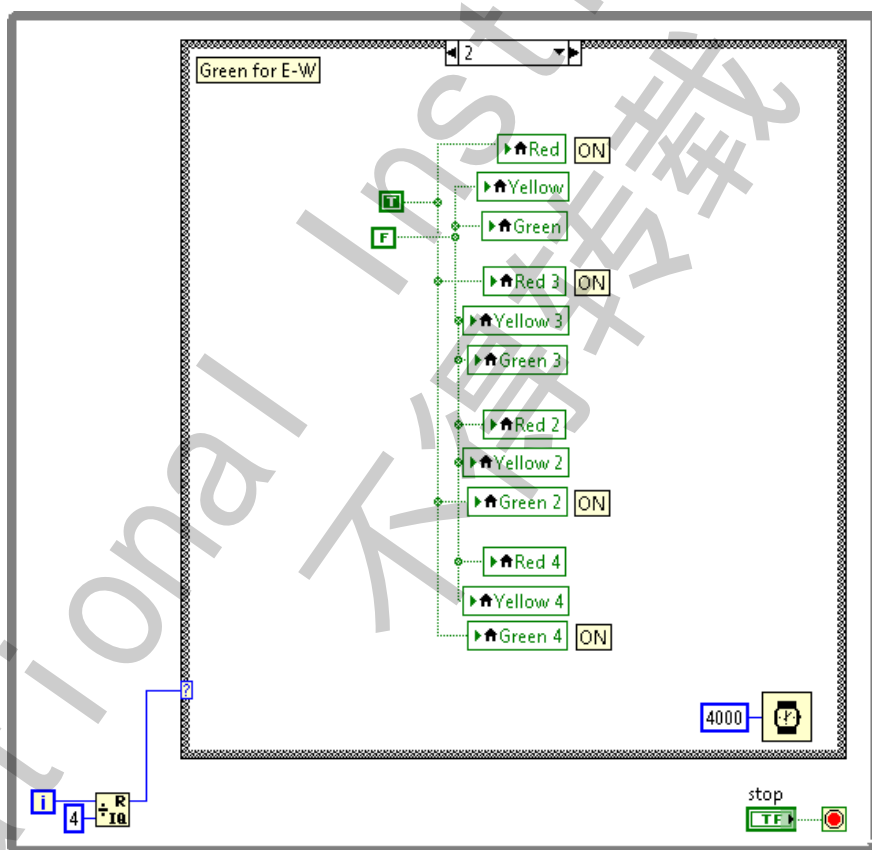
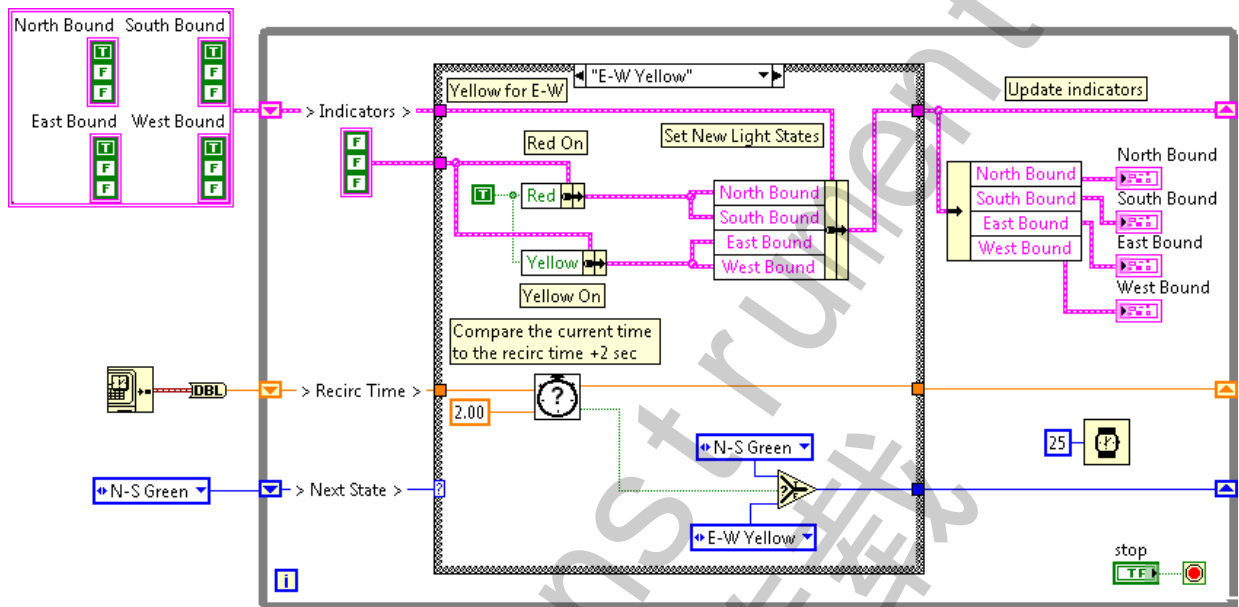


图 8-11 中的范例完成相同的任务，但更加有效且使用较好的设计方案。注意，此范例相对于上一个范例的可读性增强了，主要是减少了变量的使用。在条件结构外部的 While 循环中放置显示控件，无需使用变量，该显示控件可在每个状态后进行更新。如后续要增加功能（例如，添加左转信号灯），修改此框图的难度也有所降低。

图 8-11. 减少变量



D. 竞争状态

事件定时或任务安排无意中影响输出或数据值时，将产生竞争。对于并行执行多个任务并在任务间共享数据的程序来说，竞争状态为常见的问题。

竞争状态不容易识别和调试，因为输出取决于操作执行安排的任务及外部事件定时的顺序。通常带有竞争状态的代码即使在数千次测试中均返回相同的结果，但仍有在某次测试中返回不同的结果的可能性。

避免竞争状态的最佳办法如下列所示：

- 控制和限制共享资源。
- 通过指定执行顺序，正确排序操作。
- 减少变量的使用

控制和限制共享资源

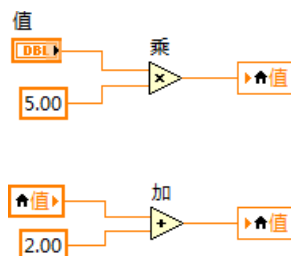
当两个任务同时具备对资源的读取访问权限时，通常会出现竞争状态。资源是指进程间共享的实体。处理竞争状态时，最常见的共享资源为数据存储（例如，变量）。其他资源的范例包括对硬件资源的文件和引用。

更改多个位置的资源时，可能引入竞争状态。因此，避免竞争状态的理想方式是降低共享资源和现有共享资源的写入方数量。通常共享资源可具有多个读取方或监视方。但请尽量为共享资源使用一个写入方或控制器，以避免竞争状态。

合理安排指令的先后顺序

未正确安排执行顺序的数据流代码可能导致竞争状态。未建立数据依赖性的情况下，LabVIEW 可以任意顺序安排任务。如任务间存在依赖性，则可能导致竞争状态。参照图 8-12 中的范例。

图 8-12. 简单竞争状态



范例中的代码具有 4 个可能的输出，具体取决于执行操作的顺序。

输出 1: 值 = (值 × 5) + 2

1. 接线端读取值。
2. 值 × 5 被存储在“值”中。
3. 局部变量读取值 × 5。
4. (值 × 5) 被存储在“值”中。

输出 2: 值 = (值 + 2) × 5

1. 局部变量读取值。
2. 值 + 2 被存储在“值”中。
3. 接线端读取值 + 2。
4. (值 + 2) × 5 被存储在“值”中。

输出 3: 值 = 值 × 5

1. 接线端读取值。
2. 局部变量读取值。
3. 值 + 2 被存储在“值”中。
4. 值 × 5 被存储在“值”中。

输出 4: 值 = 值 + 2

1. 接线端读取值。
2. 局部变量读取值。
3. 值 × 5 被存储在“值”中。
4. 值 + 2 被存储在“值”中。

尽管本代码存在竞争状态，但相对于第一个竞争状态，代码表现出的随机性小一些。因为 LabVIEW 总是按照固定的顺序执行操作。但仍应避免上述状态，因为 VI 的顺序和动作可能发生改变。例如，在不同的条件下或升级 VI 至 LabVIEW 更新版本时，顺序可能发生改变。此类竞争状态可通过控制数据流更改。

Evolutional Instruments
不得转载

自测：练习

1. 应在 VI 中尽可能多的使用变量。
 - a. 对
 - b. 错

2. 控制资源时，下列哪种写入方和读取方的组合可降低出现竞争状态的可能？
 - a. 一个写入方，一个读取方
 - b. 一个写入方，多个读取方
 - c. 多个写入方，一个读取方
 - d. 多个写入方，多个读取方

International Instruments
不得转载

自测：练习答案

1. 应在 VI 中尽可能多的使用变量。
 - a. 对
 - b. 错**
仅在需要时使用变量。尽可能使用连线传输数据。

2. 控制资源时，下列哪种写入方和读取方的组合可降低出现竞争状态的可能？
 - a. 一个写入方，一个读取方**
 - b. 一个写入方，多个读取方**
 - c. 多个写入方，一个读取方
 - d. 多个写入方，多个读取方

笔记

ational Instruments
不得转载

更多信息和资源

附录包含关于 NI 技术支持和 LabVIEW 资源的补充信息。

NI 技术支持

登录 ni.com 的 National Instruments 用户账号，自定义所需的技术支持服务。请访问 ni.com 的下列页面，获取技术支持和专业服务：

- **支持** — 技术支持 ni.com/support/zhs/ 的在线技术支持资源包含以下内容：
 - **自助资源** — 请访问 ni.com/support，查阅有关软件驱动及更新、可搜索的知识库、产品手册、疑难解答向导、数千个范例程序、产品教程、应用手册、仪器驱动程序等相关信息。注册用户还可访问 NI 技术论坛 ni.com/forums。NI 的应用工程师可确保所有您在线提交的问题得以解答。
 - **标准服务项目成员** — 成员可通过电话或 email 直接获取 NI 应用工程师的一对一技术支持，以及 ni.com/self-paced-training 提供的专享自选在线培训模块。购买 NI 开发者套件等大多数软件产品或套件后，您都将自动获取一年的标准服务项目成员 (SSP) 资格。NI 还为您提供了各种续期服务，保证您在会员资格到期后能继续享受 SSP 的各种权益。详细信息请访问 ni.com/ssp。
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。
- **系统集成** — NI 联盟伙伴可帮助解决时间限制、内部技术资源短缺或其他项目问题。NI 联盟伙伴将系统集成者、咨询人员和硬件供应商集合在一起为客户提供全方位的服务和专家咨询。该计划为系统应用和开发提供有力的、专业的帮助。详情请致电当地 NI 办事处或登录网站 ni.com/alliance。

您也可登录 ni.com/niglobal 全球办事处查找最新的办事处联系方式、技术支持电话、电子邮件地址并获取最新消息。

National Instruments 的其他培训课程

National Instruments 向 LabVIEW 用户提供多种培训课程。这些课程在您已学习课程的基础上进一步深入，并且拓展到其他领域。您可访问 ni.com/training/zhs 购买课程资料，也可在全球各地报名参加面授课程。

National Instruments 认证

获得 NI 认证意味着客户已能熟练使用 NI 产品和技术。测量与自动化业界、雇主、客户和同行都将 NI 认证视为使用 NI 产品所获得的技术和知识的一种标志。关于 NI 认证项目的详细信息，请访问 ni.com/training/zhs。

LabVIEW 资源

这部分说明了如何获得更多关于 LabVIEW 的信息。

NI 社区

NI 社区是一个在线社区，其集合了来自全球各地的用户共同讨论、分享和学习 LabVIEW。请访问 NI 社区咨询、了解及分享 LabVIEW 项目代码。详细信息请访问 ni.com/community。

LabVIEW 图书

关于 LabVIEW 编程和应用程序的图书有很多。NI 网站上有 LabVIEW 图书的列表以及这些图书出售处的链接。更多详细信息，请访问 <http://china.ni.com/academic/teach/textbook>。

词汇表

B

标签	命名、描述前面板或程序框图中的对象或区域的文字对象。
布尔控件	前面板对象，用于操作和显示布尔数据（TRUE 或 FALSE）。
波形	以特定采样率采集的多个电压读数的集合。
波形图表	以特定速率绘制数据点的显示控件。

C

菜单栏	用于列出程序中的主菜单名的水平条，在窗口标题栏的下方。虽然部分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。
测量设备	DAQ 设备（例如，E 系列多功能 I/O (MIO) 设备、SCXI 信号调理模块和开关模块）。
传统 NI-DAQ (Legacy)	较早版本的 DAQ 驱动，用于为早期 NI DAQ 设备配置数据采集、仪器控制程序。传统 NI-DAQ (Legacy) 只在某些特定情况下适用。关于何时使用传统 NI-DAQ (Legacy)，及其支持的设备、操作系统、程序软件和语言版本的详细列表，请参阅 <i>NI-DAQ Readme</i> 。

错误列表 窗口	显示 VI 中错误和警告的窗口，有时还显示修改错误的建议。
错误信息	软件或硬件存在故障，或是输入了无法接收的数据时的说明信息。
程序框图	程序或算法的图形化表示。程序框图由可执行图标（称为“节点”）和在节点间传送数据的连线组成。程序框图是 VI 的源代码，在 VI 的程序框图中。
操作值工具	在控件中输入数据或对控件进行操作的工具。

D

DAQ	见“数据采集 (DAQ)”。
DAQ 设备	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 设备包括插入式设备、PCMCIA 卡和 DAQPad 设备，与计算机的 USB 或 IEEE1394 端口连接。SCXI 模块也属于 DAQ 设备。
DAQ 助手	配置测量任务、通道和换算的图形化界面。
断开的 VI	发生错误导致无法运行的 VI；在断开的 运行 按钮中用断开的箭头表示。
断开的 运行 按钮	发生错误导致 VI 无法运行时， 运行 按钮处于断开状态。
当前 VI	前面板、程序框图或图标编辑器处于活动状态的 VI。

动态数据类型	Express VI 使用的数据类型包括与信号相关的数据和说明信号相关信息的属性（例如，信号名称或数据采集的日期和时间）。属性指定了信号在图形或图表上的显示方式。
定位工具	移动对象、改变对象大小的工具。
对象	前面板和程序框图上各项的统称（包括输入控件、显示控件、节点、连线以及导入的图片）。

E

Express VI	用于进行常规测量任务的子 VI。Express VI 可在配置对话框中进行配置。
------------	--

F

For 循环	执行指定次数的迭代循环结构。等效于下列文本代码：For i = 0 to n - 1, do...。
复选框	对话框中的小型方框，可勾选或取消勾选。复选框一般用于在多个选项中进行选择。可同时选中多个复选框。

G

工具	特定的光标工具，可用于实现特定操作。
工具栏	工具栏包含各种命令按钮，可用于运行和调试 VI。

H

函数	内置执行元素，相当于文本编程语言中的操作符、函数或语句。
函数选板	包含 VI、函数、程序框图结构和常量的选板。

I

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

J

节点	程序执行单元。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	程序控制元素（例如，平铺式和层叠式顺序结构、条件结构、For 循环或 While 循环）。

即时帮助窗口

移动光标移动至 LabVIEW 对象时，该窗口可显示对象的基本信息。可在即时帮助窗口显示说明信息的对象包括：VI、函数、常量、结构、选板、属性、方法、事件和对话框的各组成部分。

接线端

节点上用于传输数据的端口。

K**控件**

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。

快捷菜单

右键单击某个对象时出现的下拉菜单。快捷菜单只作用于其所属的对象。

控件选板

包含前面板中输入控件、显示控件和修饰型对象的选板。

L**LabVIEW**

Laboratory Virtual Instrument Engineering Workbench 的简称，LabVIEW 是图形化编程语言，通过图标替代文本行创建程序。

连线

节点间的数据路径。

连线工具

定义接线端之间数据路径的工具。

M**MAX**

见“Measurement & Automation Explorer”。

Measurement & Automation Explorer

Windows 平台上的标准 NI 硬件配置和分析环境。

默认

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。

N**NI-DAQ**

该驱动程序软件包含所有的 NI-DAQ 设备和信号调理组件。NI-DAQ 是可通过应用程序开发环境 (ADE)（例如，LabVIEW）调用的扩展程序库（包含 VI 和 ANSI C 函数），用于配置所有 NI 测量设备（例如，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，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：DAQ 助手可配置设备的通道和测量任务，可用于 LabVIEW、LabWindows™/CVI™ 和 Measurement Studio；NI-DAQmx 的仿真功能支持绝大多数设备，无须插入硬件即可测试和修改程序；创建 DAQ 程序的 API 使用更少的函数和 VI。

P

PXI PCI 仪器扩展接口，基于计算机的模块化仪器平台。

Q

驱动程序 用于控制硬件设备（如 DAQ 设备）的软件。

前面板 VI 的交互式用户界面。前面板外观类似于真实的物理仪器（如示波器和万用表）。

曲线 数据阵列在图形或图表上的图形表示。

R

任务 在 NI-DAQmx 中一个或多个通道、定时、触发等属性的集合。任务表示要执行的信号测量或信号发生任务。

S

设备 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信网络连接到主机。见“DAQ 设备”和“测量设备”。

数据采集 (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。

缩放 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。

数据流 由可执行节点组成的编程系统，节点仅在接收到所有必需的输入数据后才开始运行。节点在执行时可自动生成输出数据。LabVIEW 是基于数据流的系统。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。

数据类型 信息的格式。LabVIEW 中绝大多数 VI 和函数接收的数据类型包括数值、数组、字符串、布尔、路径、引用句柄、枚举、波形和簇。

数值输入控件和显示控件 用于管理和显示数值数据的前面板对象。

属性对话框 通过控件的快捷菜单访问的对话框，可用来配置前面板显示的控件。

T

图标

程序框图上节点的图形化表示。

通道

1. 物理通道—用于测量和生成模拟信号或数字信号的接线端或引脚。一条物理通道可以包括多个端口，如一条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条物理通道，但计数器与其对应的接线端可采用不同的命名。
2. 虚拟通道—包括名称、物理通道、输入终端连接、测量或生成信号类型，以及缩放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 在 MAX 中配置虚拟通道。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，可以将通道配置作为任务的一部分也可以独立配置通道。
3. 开关通道—开关通道表示开关上任意的连接点。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。

条件接线端

While 循环的接线端，包含一个决定 VI 是否再执行下一次循环的布尔值。

图例

图形或图表的示例，用于显示在该图形或图表上的名称和样式。

提示框

小型黄色文本框，显示接线端名称以确定需连接的接线端。

图形

一条或多条曲线的二维显示。图形接收并绘制数据块。

通用接口总线

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，因为它根据 ANSI/IEEE 488-1978、488.1-1987 和 488.2-1992 标准进行定义的。

拖曳

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”。

VI 模板

含有常用控件的 VI，可基于模板创建多个 VI，以执行相似的功能。从**新建**对话框中访问 VI 模板。

W

While 循环

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。

X

选板	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。
显示控件	显示输出的前面板对象，例如图形或指示灯。
下拉菜单	菜单栏上的菜单。通常，下拉菜单项属于同一类。
项目	一组 LabVIEW 文件和非 LabVIEW 特有的文件，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。
项目浏览器 窗口	创建和编辑 LabVIEW 项目的窗口。
虚拟仪器	使用 LabVIEW 编写的程序，具有真实的物理仪器的外观和功能。

Y

样本	单个模拟或数字输入或输出的数据点。
仪器 I/O 助手	从“仪器 I/O 助手”Express VI 启动的附加程序，与仪器进行基于消息的通信，并以图形方式解析响应。
仪器驱动	在系统中控制仪器硬件并与之通信的一套高层函数。

Z

自动调整标尺	标尺根据标绘数值的范围进行自动调整。在图形标尺中，自动标尺调整将确定刻度的最大和最小值。
字符串	值的文本表示。
指示灯	发光二极管，指示灯。
子选板	通过上级选板向下访问的选板。
子 VI	在其它程序框图中使用的 VI，类似于子程序。