第1章　绪　　论

1.1　虚拟仪器与 LabVIEW

21 世纪以来，电子技术、计算机技术和信息技术突飞猛进，各种测试、测量和自动化系统也因此得到了长足发展。作为提高生产力的重要手段之一，这些系统在科研与生产中充当着非同寻常的重要角色。然而，随着商业社会竞争日益加剧以及经济飞速发展，人们对自动化系统的功能和开发周期提出了更高的要求。一方面，要求系统开发者在非常短的工期内完成项目成果交付；另一方面，又要求所设计的自动化系统尽可能成本低、性能高、扩展性强并可实现无缝集成。这就迫使工程人员不断寻求新的技术和理念，通过创新来进一步提高开发效率，虚拟仪器（Virtual Instruments）应运而生。

虚拟仪器是指利用计算机把高性能模块化硬件（如 A/D 转换器、D/A 转换器、数字输入 / 输出、定时和信号处理）和灵活可定制的软件（如数据分析软件、数学计算软件、通信软件及仪器界面等）结合起来完成各种测试、测量和自动化的应用（图 1-1）。传统的、基于硬件的仪器或自动化系统一般由硬件电路实现自动化系统中的数据采集、信号处理、结果显示及仪器控制等功能。而虚拟仪器则一般基于计算机，综合灵活可定制的软件、传感器和运动控制模块来取代传统仪器或自动化系统的功能。

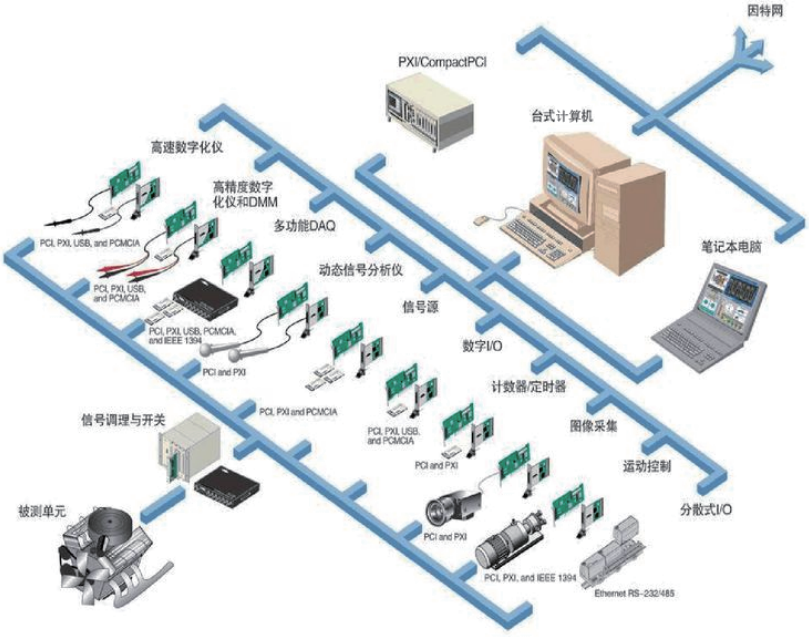


图 1-1　虚拟仪器系统

虚拟仪器是一种全新的仪器概念，通常由硬件设备与接口、设备驱动、数据处理软件和虚拟仪器面板组成。其中，硬件设备与接口可以是各种以计算机为基础的内置功能插卡、通用接口总线接口卡、串行口、VXI 总线仪器接口等设备，或者是其他各种可程控的外置测试、运动控制设备。设备驱动软件是直接控制各种硬件接口的驱动程序，虚拟仪器通过底层设备

驱动软件与真实的仪器系统进行数据交换，经数据处理软件处理后的数据以虚拟仪器面板的形式在计算机屏幕上显示为类似真实仪器面板的显示元件。用户用鼠标操作虚拟仪器的面板上的各种控件来控制外部硬件设备，就如同操作真实仪器一样真实、方便。

软件代替大量硬件是虚拟仪器系统最主要的特色，这一突破不仅允许基于虚拟仪器的系统可使用计算机取代以往昂贵的硬件模块来降低系统成本、提高系统性能，还可以使用户通过对软件的定制或更新来快速、灵活地对系统进行变更或扩展。计算机可以通过不同的接口硬件（ISA、PCI、 PXI、PCMCIA、USB、GPIB、IEEE 1394，串口和并口等）将多种设备无缝集成至系统。由于虚拟仪器的工作过程完全受控于软件，仪器功能的实现在很大程度上取决于软件的功能设计，因此用户可以自由地定义仪器的功能，用一套虚拟仪器硬件来实现多种不同仪器的功能。当然，为了快速可靠地开发各种虚拟仪器，一套优秀的虚拟仪器软件开发平台是必不可少的。

进行虚拟仪器开发的语言和工具很多。传统基于文本的开发语言有 C/C++、Python、Java、C#、Basic 等；基于文本语言编程的开发工具有 Visual Studio（VC++/VB/VC# 等）、Qt、Eclipse、Borland C++/Delphi、NetBeans 以及 Linux 平台下的 GCC 和 Gnome/GTK 库等。但是这些软件多为通用的开发软件，面向多个行业，并不提供规范的虚拟仪器开发元件库和仪器设备驱动集。如果开发任务集中在自动化测控领域，使用它们往往需要从头开发各种虚拟仪器软件模块和硬件驱动，这就如同从每个螺丝钉开始造汽车一样，开发周期长，成本也高。

在实际开发虚拟仪器系统时，我们往往选择提供规范的虚拟仪器开发元件库和仪器设备驱动集的开发工具。比较典型的有美国国家仪器公司（National Instruments，2020 年更名为 NI）的 LabVIEW、LabWindows CVI 和安捷伦科技公司（Agilent Technologies）的 HP VEE（Visual Engineering Environment）等。 其中 LabWindows CVI 为支持 C 语言的虚拟仪器开发工具。LabVIEW 和 HP VEE 虽然均为图形化编程语言（Graphical Programming Language，简称 G 语言）的开发工具，但是 HP VEE 却主要支持安捷伦公司自己的仪器，相对而言 LabVIEW 支持的仪器更广泛。此外根据 2015 年 EE Times 和 EDN 杂志对测试、测量和仪器控制领域各类开发环境的使用情况统计（图 1-2）来看，LabVIEW 的使用最为广泛，它所支持的图形化编程语言也因此成为虚拟仪器项目开发的首选编程语言。 此外，近年来，随着各种官方功能模块和第三方扩展模块的不断发布，LabVIEW 不仅继续保持虚拟仪器开发平台的首要位置，也逐渐向完善的集成开发环境编程方向发展。

LabVIEW（Laboratory Virtual Instrumentation Engineering Workbench）是 NI 公司为测试、测量和控制应用而设计的可视化、跨平台（可在 Windows、Linux、macOS 上运行）系统工程软件。它使用图形化的程序设计语言代替文本行代码来创建应用程序，并可以快速访问和控制硬件和数据信息。传统文本编程语言根据语句和指令的先后顺序决定程序执行顺序，而 LabVIEW 程序的执行顺序则由程序框图中节点之间的数据流向决定，即数据流驱动（Data Flow Driven）。开发人员使用图标化的函数和数据连线，依据数据流逻辑来开发复杂的测量、测试和控制系统。LabVIEW 广泛支持各种仪器硬件的驱动，包含大量内置和扩展的函数库（如数据采集、信号处理、数学计算、统计分析、图像处理、机器视觉、运动控制、数据通信、数据库、报表生成、移动开发、嵌入式开发等），并且都形象地表现为图形化编程语言函数。图 1-3 显示了 LabVIEW 图形化设计平台结构。

利用 LabVIEW 既能集成数千款硬件设备，也能通过直接使用大量内置库实现高级分析和数据处理。这就使开发人员不必从头开发各种处理函数，甚至不必关心处理函数内部的具体细节，从而大大缩减了系统开发时间，并使开发人员能专注于整个系统功能的研究，而不是在可重用的函数开发上浪费时间。此外，由于 LabVIEW 为编译型（并非解释型）开发工具，

其生成的程序执行效率并不输于一般的文本编程开发工具，因此自 1986 年推出以来，就被广泛地应用于数据采集、仪器控制、工业自动化、航空航天和科学研究等领域。

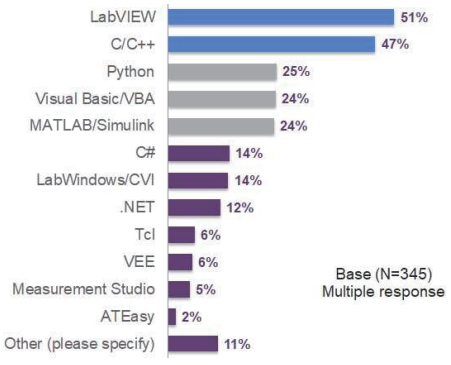


图 1-2 测试、测量和仪器控制领域常用软件的统计结果

图 1-3 LabVIEW 图形化设计平台结构

狭义来说，LabVIEW 程序可以被称为虚拟仪器，因此通常我们把 LabVIEW 程序称作 VI（Virtual Instruments）。每个 VI 由三部分组成（图 1-4）：

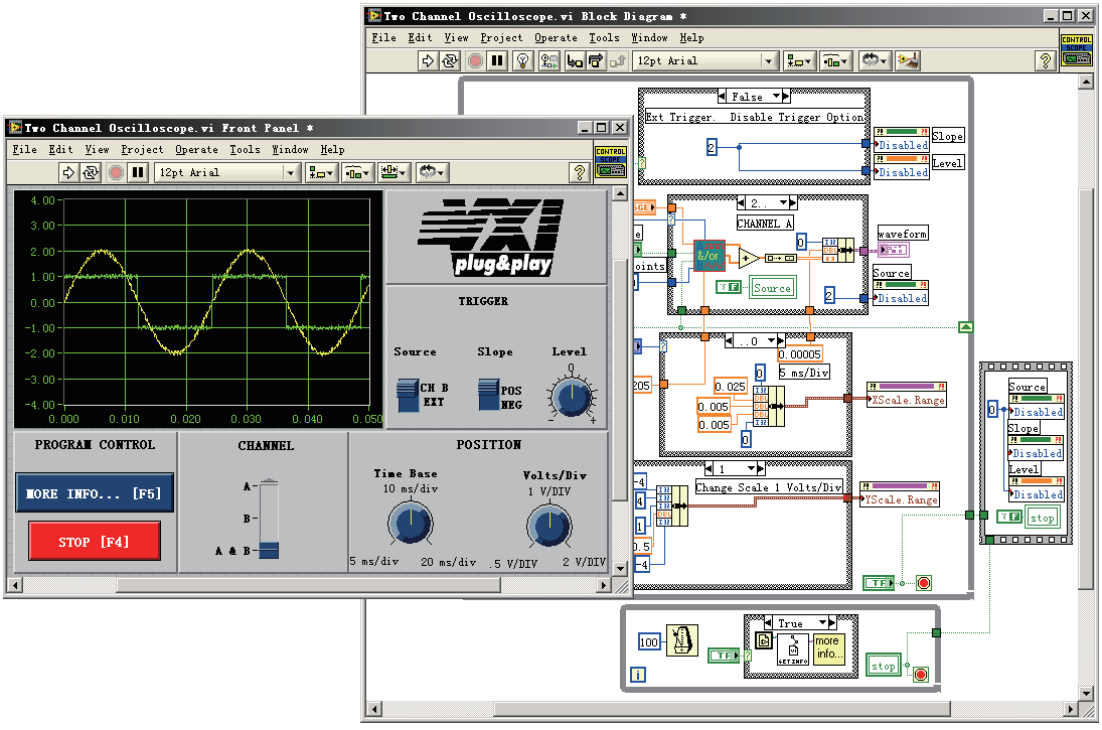


图 1-4 由前面板、程序图和输入 / 输出端子组成的虚拟仪器

（1）前面板（Front Panel）：即用户界面。

（2）程序框图（Diagram Block，又称后面板）：定义 VI 功能的图形化源代码。

（3）图标和输入 / 输出接口端子（Icon & Connectors）：用来定义 VI 的输入 / 输出参数（接口），以便 VI 可以被其他 VI 当作子 VI（SubVI，相当于文本编程语言中的子程序）调用。

前面板由输入控件（Controls）和显示控件（Indicator）组成。这些控件是 VI 的输入 / 输出接口。输入控件有旋钮、按钮、转盘等。显示控件有图表、指示灯等。输入控件模拟仪器的输入装置，为 VI 的程序框图提供数据。显示控件模拟仪器的输出装置，用以显示程序框图获取或生成的数据。

前面板创建完毕，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。前面板上的对象在程序框图中显示为接线端（Terminal），接线端表示输入控件和显示控件的数据类型。开发人员将对数据操作的函数图标添加到程序框图上，并使用循环和条件结构控制程序的执行方式，使用连线控制程序框图中对象的数据传输来实现虚拟仪器的功能。这种基于图标和连线的开发语言称为图形化开发语言（G 语言）。在 G 语言代码中，每个功能块通过连接到输入 / 输出端子的连线连接，只有当每个功能块需要的输入数据（变量）全部到达后，该功能块才能被执行，因而程序中各功能块的执行顺序受数据驱动。

当 VI 作为子 VI（SubVI，类似子程序）被其他 VI 调用时，将以图标形式显示在主 VI 的后面板中，子 VI 的输入 / 输出通过其前面板中的部分输入控件（对应输入参数）和显示控件（对应输出参数）与其连接端子的一一对应来确定。也就是说每个子 VI 均可在单独进行调试后再被嵌入主 VI。

此外，由于程序中的数据可能同时到达多个并行放置的功能块，因此数据驱动原理与生俱来就支持并行运行。当然多进程和多线程执行还是要受操作系统的调度。总的来说，使用 LabVIEW 可以非常快速地构建虚拟仪器系统，本章后续章节，先介绍如何基于 LabVIEW 搭建虚拟仪器项目的开发环境，然后以一个简单的实例，简要介绍虚拟仪器的开发和调试过程，使读者对基于 LabVIEW 的虚拟仪器项目开发过程有一个比较初步的概念。

1.2 虚拟仪器开发环境的搭建

基于 NI LabVIEW 搭建虚拟仪器开发环境时，通常根据开发项目需求的不同，涉及以下三方面全部或部分工作。

（1）LabVIEW 软件自身的安装。

（2）LabVIEW 附加工具模块（Add-Ons、Toolkits）的安装，包括：

- 安装 NI 提供的附加工具模块；
- 安装第三方开源或商用工具模块。

（3）硬件设备驱动（Device Drivers）安装。

在本书编写过程中，LabVIEW 2020 和全新一代的 LabVIEW NXG 以及它们对应的免费社区版（LabVIEW Community Edition）均已发布，一些新的功能还在开发和完善中。LabVIEW NXG 主要增强了对更多硬件和 Web 的支持，允许工程人员在其集成开发环境中一站式完成虚拟仪器项目的配置、测控及测试结果的可视化，如图 1-5 所示。NI 官方建议开发人员使用 LabVIEW NXG 5.0.0 进行产品测试及物理测量类型的虚拟仪器项目开发。由于 LabVIEW NXG 5.0.0 仅仅支持 LabVIEW 2020 及其附加工具的部分功能，且对诸多硬件的支持还不完善，因此对于诸如大型分布式测控系统、智能仪器或工业设备等虚拟仪器项目开发，建议选择 LabVIEW 2020 更为稳妥。此外，从论坛上用户的反馈来看，不少人对迁移至 LabVIEW NXG 意见很大，因此未来 NI 是继续完善 LabVIEW NXG 还是将其中功能与 LabVIEW 未来版本整合还是未知数。

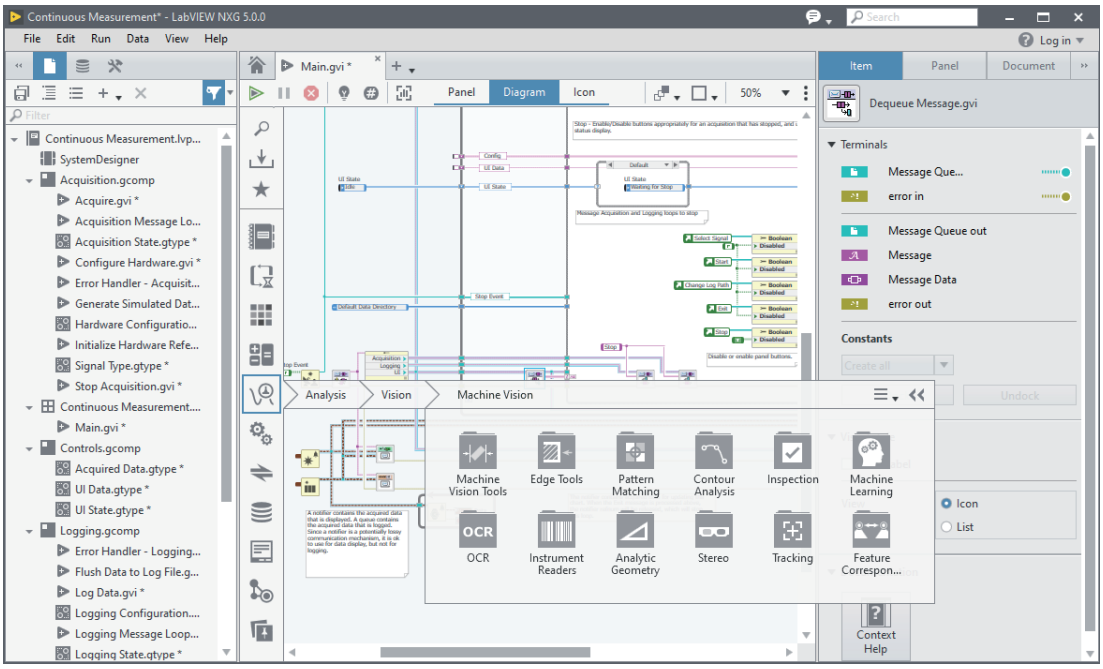


图 1-5 LabVIEW NXG 集成开发环境

LabVIEW 2020 和 LabVIEW NXG 社区版主要针对非商业用途设计。若用户仅将 LabVIEW 用于个人而非商业用途，就可以免费从 NI 的官网下载安装并使用。LabVIEW 社区版不仅包含 LabVIEW 专业版的所有功能，还包含针对 Raspberry Pi、BeagleBoard 和 Arduino 平台的 Linux 工具包，并能访问 LabVIEW NXG Web 模块，以创建基于 Web 的应用程序。

从 LabVIEW 2009 版本起，NI 开始在 Windows 平台上发布 LabVIEW 32 位和 64 位两个版本。自 LabVIEW 2014 版本起，NI 开始在 Linux 和 macOS X 平台上同时发布 LabVIEW 32 位和 64 位版本。相较于 32 位版本，64 位版本的运行速度并未进行提升，并且不能支持 32 位版本所支持的全部附加工具模块。但是基于 64 位版本开发的程序，可以一次性访问更大的内存空间。应用程序可访问的最大内存空间由操作系统决定，32 位 LabVIEW 在 64 位 Windows 平台上运行时，可以访问 4GB 的地址空间，64 位 LabVIEW 在 64 位 Windows 平台上运行时，理论上可支持 16TB 的内存。大多数情况下，使用 32 位版本的 LabVIEW 就可以满足虚拟仪器项目的开发要求。仅在部分特殊情况下，如大数组操作或大图像处理，才会用到 64 位版本。

LabVIEW 32 位和 64 位版本开发的 VI 源文件可以相互兼容，但是用某版本编写的源代码若要在另一版本上运行，则需要重新进行编译。若用某版本编写的代码包含的功能在另一版本上不存在，则会编译失败。用 64 位 LabVIEW 开发的 VI 编译成可执行文件后，将不能在 32 位的操作系统上执行。而用 32 位 LabVIEW 开发的 VI 编译成可执行文件后，要在 64 位的机器上运行，还可能需要要根据操作系统不同，稍作处理。若操作系统为 64 位 Windows，32 位 LabVIEW 编译的可执行文件可以直接运行。若操作系统为 64 位 macOS X，则需要将操作系统的内核切换到 32 位才能执行 32 位 LabVIEW 编译的可执行文件。需要注意部分版本的 macOS X 操作系统并不支持内核切换。若操作系统为 64 位 Linux，则需要安装 32 位的运行库才能运行 32 位 LabVIEW 编译的可执行文件。需要注意这些 32 位库并不能确保所有 32 位应用程序都能在 64 位机器上运行。

综上所述，LabVIEW 32 位版本目前对硬件和附加工具模块有更好的支持，能满足大多

数虚拟仪器项目的开发需求，因此一般基于 LabVIEW 32 位版本进行虚拟仪器开发平台的搭建，仅在程序需要进行大内存操作时才切换到 LabVIEW 64 位版本上。此外，目前 NI 公司每年会分两次发布 LabVIEW，春季版本为当年新版本，秋季版本通常为补丁（Service Pack），用于解决春季版本中发现的问题。每年春季发布新版本后，NI 公司会在接下来的 4 年中对其进行支持（表 1-1）。由此，在开始一个新的虚拟仪器项目时，建议选用秋季发布的最新 32 位版本进行工程应用开发。

表 1-1　NI 公司对各版本 LabVIEW 的支持情况

支持的版本	发 布 时 间	计划结束支持时间	状　　态
LabVIEW NXG 5.1	2021 年 1 月	2025 年 1 月	支持中
LabVIEW NXG 5.0	2020 年 5 月	2024 年 5 月	支持中
LabVIEW 2020	2020 年 5 月	2024 年 5 月	支持中
LabVIEW NXG 4.0	2019 年 11 月	2023 年 11 月	支持中
LabVIEW 2019	2019 年 5 月	2023 年 5 月	支持中
LabVIEW NXG 3.1	2019 年 5 月	2023 年 5 月	支持中
LabVIEW NXG 3.0	2018 年 11 月	2022 年 11 月	支持中
LabVIEW 2018	2018 年 5 月	2022 年 5 月	支持结束
LabVIEW NXG 2.1	2018 年 3 月	2022 年 3 月	支持结束
LabVIEW NXG 2.0	2018 年 1 月	2022 年 1 月	支持结束
LabVIEW NXG 1.0	2017 年 5 月	2021 年 5 月	支持结束
LabVIEW 2017	2017 年 5 月	2021 年 5 月	支持结束

注：表中未列出的 LabVIEW 版本，计划支持时间已结束，如需延长支持，需联系 NI 公司。

工程开发过程中若需要打开旧版本 LabVIEW 创建的 VI，可参阅图 1-6，选择合适的版本或转换工具来完成。其中 LabVIEW 3.x 以及之前版本创建的 VI，需要借助 LabVIEW VI Conversion Kit 免费工具才能转换为较高版本支持的 VI（图 1-6 注解 C）。LabVIEW 8.5 以及后续版本要打开 LabVIEW 6.0 之前版本创建的 VI，需要先将其转换为 LabVIEW 6 ～ 8.2.x，然后再由 LabVIEW 8.5.x 或者后续版本打开（图 1-6 注解 I）。这意味着若要在 LabVIEW 8.5 以及后续版本中打开 LabVIEW 3.x 之前版本的 VI，就需要先借助 LabVIEW VI Conversion Kit 将其转换为 LabVIEW 6 ～ 8.2.x 版本，再由 LabVIEW 8.5.x 或者后续版本打开（图 1-6 注解 C+I）。

反过来，LabVIEW 8.0 及以上版本可以将 VI 直接另存为 LabVIEW 8.0 或高于 8.0 但低于自己的版本。但是若要将 VI 从高版本转换为低于 7.1.x 的版本，则需要借助中间版本来实现转化过程。例如，若要将 LabVIEW 8.2.x 版本的 VI 另存为 LabVIEW 7.1.x 版本，则需要先将 VI 另存为 LabVIEW 8.0.x 版本，再由 LabVIEW 8.0.x 另存为 7.1.x 版本（图 1-6 注解 M）。

新旧 LabVIEW VI 版本转换过程中应注意以下几点。首先，使用新版本 LabVIEW 创建的 VI，若其中包括了在该新版本中首次引入的功能，则在向较低版本转换时，该功能将不可用。其次，如果 VI 中包括第三方工具或 Toolkit 中的 VI，则转换过程中该第三方 VI 的版本转换应单独考虑，一般需参考该第三方工具版本的演化来单独处理。最后，在将 VI 转换为旧版本时，在“.lib”中的 VI 不会参与转换过程，而是保留对其中 VI 的引用。这样旧版本 VI 在调用“.lib”中 VI 时，照样可以正常工作。

保存VI的版本	打开VI的版本																					
	5.0.x	5.1.x	6.0.x	6.1	7.0	7.1.x	8.0.x	8.2.x	8.5.x	8.6.x	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020
2.x - 3.x	C	C	C	C	C	C	C	C	C+I	C+I	C+I	C+I	C+I	C+I	C+I	C+I	C+I	C+I	C+I	C+I	C+I	C+I
4.x - 5.0.x	✓	✓	✓	✓	✓	✓	✓	✓	I	I	I	I	I	I	I	I	I	I	I	I	I	I
5.1.x	S	✓	✓	✓	✓	✓	✓	✓	I	I	I	I	I	I	I	I	I	I	I	I	I	I
6.0.x	M	S	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
6.1	M	M	S	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
7.0	M	M	M	S	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
7.1.x	M	M	M	M	S	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
8.0.x	M	M	M	M	M	S	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
8.2.x	M	M	M	M	M	M	S	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
8.5.x	M	M	M	M	M	M	S	S	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
8.6.x	M	M	M	M	M	M	S	S	S	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
2009	M	M	M	M	M	M	S	S	S	S	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
2010	M	M	M	M	M	M	S	S	S	S	S	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
2011	M	M	M	M	M	M	S	S	S	S	S	S	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
2012	M	M	M	M	M	M	S	S	S	S	S	S	S	✓	✓	✓	✓	✓	✓	✓	✓	✓
2013	M	M	M	M	M	M	S	S	S	S	S	S	S	S	✓	✓	✓	✓	✓	✓	✓	✓
2014	M	M	M	M	M	M	S	S	S	S	S	S	S	S	S	✓	✓	✓	✓	✓	✓	✓
2015	M	M	M	M	M	M	S	S	S	S	S	S	S	S	S	S	✓	✓	✓	✓	✓	✓
2016	M	M	M	M	M	M	S	S	S	S	S	S	S	S	S	S	S	✓	✓	✓	✓	✓
2017	M	M	M	M	M	M	S	S	S	S	S	S	S	S	S	S	S	S	✓	✓	✓	✓
2018	M	M	M	M	M	M	S	S	S	S	S	S	S	S	S	S	S	S	S	✓	✓	✓
2019	M	M	M	M	M	M	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	✓
2020	M	M	M	M	M	M	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	✓

C 需要LabVIEW Convertion Kit支持完成转换
I 需要中间版本软件支持才能完成转换
C+I 需要LabVIEW Convertion Kit和中间版本软件同时支持完成转换
✓ 可直接打开VI
S 可保存为旧版格式的VI
M 需多个版本支持才能完成转换

图 1-6 LabVIEW 各版本 VI 的读写

LabVIEW 及其附加工具模块的安装包均可从 NI 官网的下载通道获得。这些安装包可以分开逐个下载安装，也可以下载“NI 开发者套件”（NI Developer Suite）或“NI 软件平台合集”（NI Software Platform Bundle）进行一次性安装。NI 开发者套件和 NI 软件平台合集中包含了所有 NI 提供的虚拟仪器开发工具及附加工具模块，包括 LabVIEW、TestStand、LabWindows CVI 以及信号处理、数字滤波、图像处理和机器视觉、声音振动分析、数据库、通信，测控等模块，在安装时可以按需选用。图 1-7 显示了基于 NI 开发者套件搭建虚拟仪器开发平台时，选择安装 LabVIEW 版本和附加工具模块的情况。

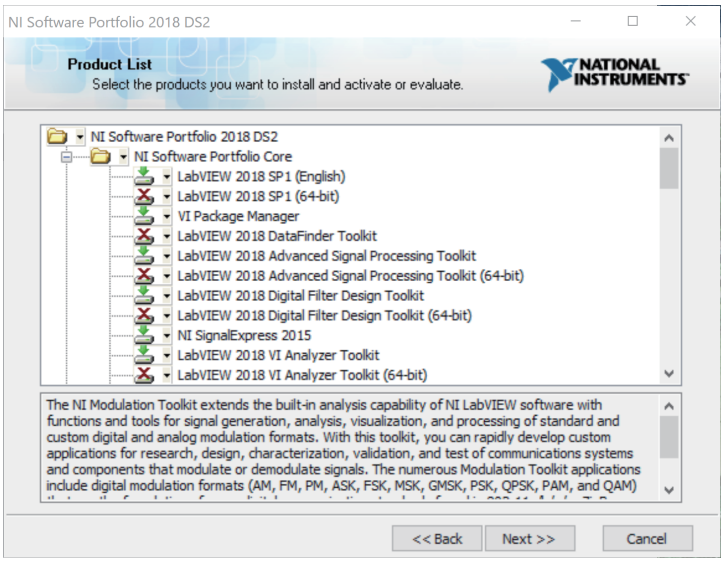


图 1-7 基于 NI 开发者套件选择安装 LabVIEW 和附加工具模块

选择确定需要安装的软件版本模块后，只需指定安装路径（图 1-8）并接受软件许可协议（图 1-9），即可继续完成所选软件的安装。

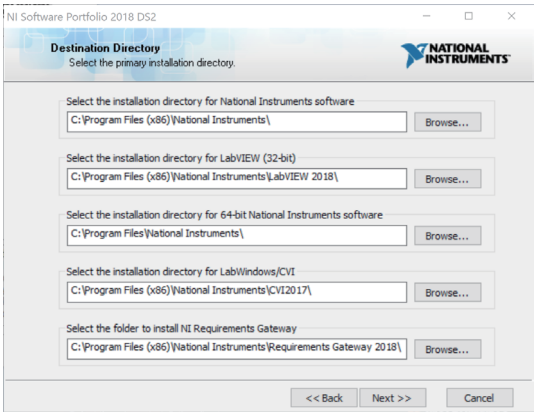


图 1-8　指定安装路径

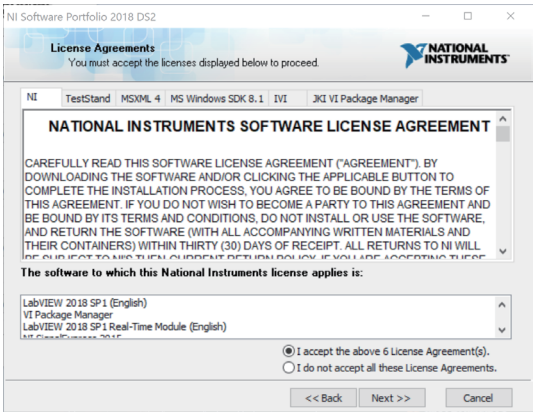


图 1-9　接受软件许可协议

在购买 LabVIEW 及相关附加工具模块软件之前，用户可以对其进行试用评估。试用版软件在试用期功能与正式版软件无异。试用期结束后，软件将被停止使用，直到用户购买软件授权并激活软件为止。值得一提的是，目前很多开发人员使用网络上的破解包对软件进行破解，虽然破解后的软件功能可以完全正常使用，但这毫无疑问是侵犯知识产权的行为，并且通过这种方式使用软件，不会得到 NI 公司的售后服务支持，建议广大读者购买正版软件进行开发。

除了 NI 公司官方提供的附加工具和模块外，也有一些第三方的工具模块值得安装使用。这些工具通常都依据开源 BSD 协议通过 VI Package Manager（VIPM）进行分发。VI Package Manager 是 JKI（http://jki.net/）开发的一个附加工具模块打包安装工具，可以在 https://vipm.jki.net/ 下载安装。安装运行 VI Package Manager 后，就可以浏览并选择所需的第三方附加工具模块（图 1-10），通过右键菜单中的选项来下载安装。

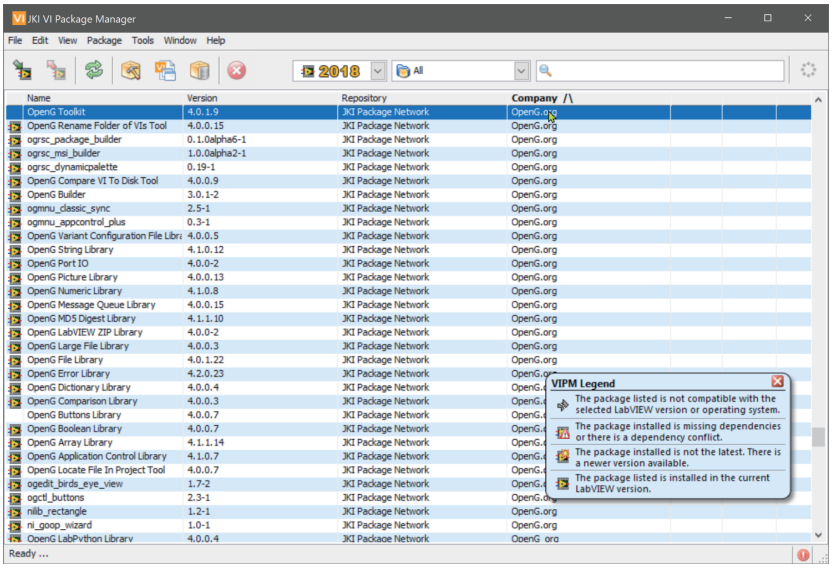


图 1-10　通过 VIPM 安装第三方附加工具模块

较受欢迎的第三方工具模块有 JKI 的开发框架和工具包、Delacor 的队列消息处理框架、

GPower 的工具包，以及 LAVA、MGI 和 OpenG.org 提供的开源开发库等。这些第三方工具模块，通常都根据工程开发过程中的实际需求，对 LabVIEW 的 VI 进行封装或优化，有较强的易用性和可靠性。

硬件设备驱动程序均打包在“NI 设备驱动包”（Device Drivers）中定期发布。这些驱动程序可以从 NI 公司官网免费下载。虽然新版本的设备驱动一般可兼容旧版本的开发平台，但多数情况下都会选择安装和开发平台相同版本的设备驱动程序。在搭建虚拟仪器开发平台时，一般在最后才安装设备驱动程序。若在安装 LabVIEW 和附加工具模块之前已经安装了设备驱动，就需要在安装这些软件后对设备驱动重新再进行安装或修复，确保设备驱动与所安装的 LabVIEW 和工具模块版本兼容。图 1-11 显示了通过 NI 设备驱动包选择安装设备驱动的情况。

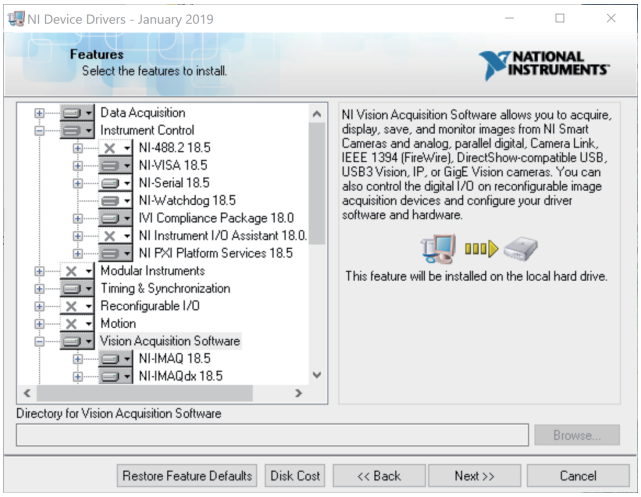


图 1-11 安装设备驱动程序

本书所附各种实例代码，基于图 1-12 所列版本的 NI 产品创建并调试通过。建议读者在学习过程中，安装相同或与之兼容的版本进行学习试验。

软 件 名	版 本	评估版下载地址
LabVIEW Professional Development System	2018 及以上	http://www.ni.com/downloads/products/
NI Device Driver	2018 及以上	http://www.ni.com/downloads/products/

图 1-12 本书代码所用版本

1.3 VI 的开发与调试步骤

虚拟仪器开发平台搭建完成后，即可运行 LabVIEW 来开发实现各种功能的 VI。本节将通过开发一个“判断数字量是否在某一范围内”的简单实例来说明 VI 的开发调试过程。包括需求分析、前 / 后面板设计、定义子 VI、运行调试及错误捕获等。由于重在说明开发流程，就省略了较为易上手的 LabVIEW 基本操作讲解。若读者是第一次接触 LabVIEW，可以先花 1 ～ 2 小时学习 LabVIEW 的入门课程，相关教学视频可在 NI 官方网站或本书的支持网站 https://www.mviacademy.com 免费获得。

假定我们要完成一个 VI，用于判断某种外部输入的实数量（如某种产品的输出电压、电流）是否在一个预先给定标称值的 ±10% 范围内。

1. 需求分析

从数据结构和算法两方面分析以上需求：

（1）数据类型和数据操作：

VI 要求输入的实数量和标称值均应为浮点数值量，需要对这些数据进行算术运算操作；

VI 输出结果只有两种可能（在范围内和不在范围内），因此为布尔量，需要进行逻辑比较运算操作。

（2）算法：

- 通过用户界面获得输入的数字量和标称值；
- 通过计算“标称值 ± 标称值 ×10%”获得范围的上下限；
- 如果“输入的数字量≥下限”并且“输入的数字量≤下限”，则表示输入数字量在范围内；
- 显示给用户输入量是否在范围内的判断结果。

2. 前面板设计

前面板是 VI 的人机界面，创建 VI 时，通常应先设计前面板，然后再设计程序框图。用户可以在前面板设置输入数值并观察 VI 的输出。输入量可以通过操作输入控件（Control）完成，程序的输出可以通过显示控件（Indicator）显示给用户。输入控件和显示控件以各种图形出现在前面板上，如旋钮、开关、按钮、图表、图形等，这使得前面板非常直观、易懂。LabVIEW 控件面板中提供了各种丰富的控件（在前面板右击会弹出控件面板，如图 1-13 所示），开发者可以直接从控件面板中选择需要的控件放入前面板来构建用户界面。选择的控件可以通过右键菜单选项在输入控件和显示控件属性之间转换，以改变组件的输入 / 输出属性。

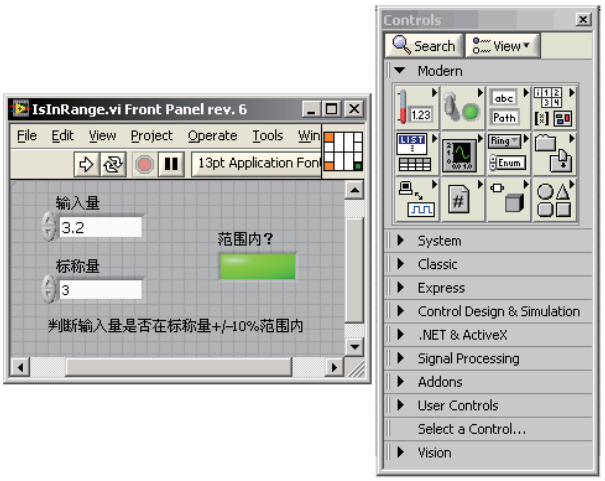


图 1-13　VI 前面板和 LabVIEW 控件面板

根据需求分析，用户前面板上应包含 2 个输入控件（输入量和标称量）和一个显示控件（显示输入量是否在范围内）。通过从控件面板中选择两个双精度类型的数字控件（Double Numeric Control）和一个方形 LED（Square LED）作为显示控件，并修改其标签（Label）后，得到如图 1-13 所示前面板。

3. 后面板设计

每一个 VI 前面板都对应一个编辑实现程序功能的 G 语言面板（通常称后面板）。从控

件选择面板上选择一个控件并放置在前面板上，LabVIEW 会自动在后面板上生成与之对应的图标，显示其对应的数据类型。前面板上用户的输入数据经过输入控件对应的图标传送至 LabVIEW 后面板。这样开发人员就可以在后面板中设计图形化的程序，实现各种功能，然后再将处理结果由输出控件对应的图标返回到前面板中。后面板中的图形化语言程序框图可理解为传统程序的源代码，图 1-14 显示了一个 VI 前、后面板之间的数据传递示意图。

G语言程序由程序结构框图（Structure）、节点（Node）、输入 / 输出端子（Terminal）和连线（Wiring）构成。其中程序结构框图（如循环、分支等）用来实现结构化程序控制命令，节点被用来实现各种功能函数和子 VI调用，输入 / 输出端子被用来同前面板的控件和显示控件传递数据，而连线则代表程序执行过程中的数据流，定义了框图内的数据流动方向。

LabVIEW 含有大量丰富的内置函数库（在后面板右击弹出函数库，见图 1-15），开发者可以直接从函数库中选择需要的函数放入后面板，这些函数以各种直观的图标节点形式出现在后面板上。随后通过工具盒中的连线工具连接各功能节点的输入 / 输出端子，并在程序结构框图的控制下以数据流驱动的方式实现各种复杂的功能。根据前述需求分析中的算法，选择相关函数并连线后创建的程序如图 1-15 所示。

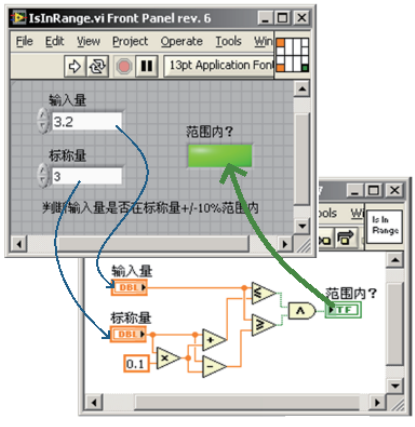


图 1-14 VI 前、后面板之间的数据传递

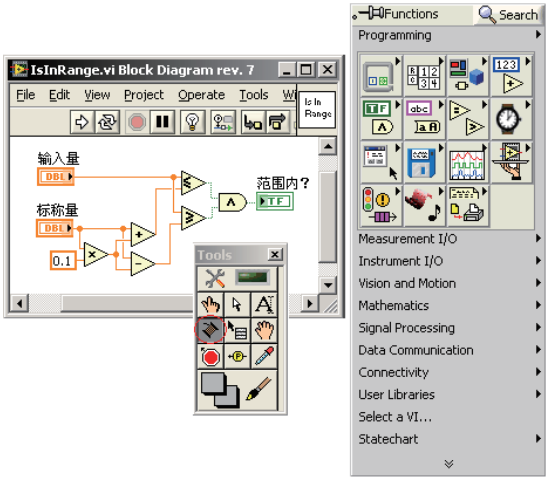


图 1-15 VI 后面板和 LabVIEW 函数库

4. 定义 VI 为子 VI（subVI）

从功能角度来看，前面板和后面板代码设计完成后，VI 设计就已经完成。然而在实际工作中，VI 不仅可作为独立的应用程序，还可以被定义为程序中的一项常用操作，当作子 VI（一个 VI 被其他 VI在程序框图中调用，则称该 VI 为子 VI）来使用。被定义为子 VI 的程序在运行时通常并不显示其前面板，而往往是被当作函数在主 VI 中调用。子 VI 既可有效提高程序的模块化程度和代码重用率，还可使 VI 后面板的图形代码更整洁易读。

定义子 VI 一般可通过三步完成：

（1）定义 VI 输入 / 输出参数。

（2）为 VI 创建图标。

（3）为 VI 添加文档说明。

要将 VI 定义为子 VI，首先需要使用连线工具为 VI 定义输入 / 输出端子。在 VI 前面板右上角图标的右键菜单中选择显示“连线板”选项，再通过选择连线的模式确定输入 / 输出端子数量（图 1-4 中 VI 前面板右上角图标），最后使用连线工具将前面板相应的输入控件与显示控件分别分配至连线板上的输入 / 输出连线端子，即可完成子 VI 输入 / 输出参数定义。

LabVIEW 连线板中最多可设置 28 个接线端。笔者在选择连线板时通常遵循以下两个原则：

（1）选择既能满足当前端子需求数量，又能保留部分裕量的组合模式，但一般选择接线端总数不超过 16 个的接线板。

（2）如果连接的参数较多，尽量使用簇对参数打包，然后将该簇分配至连线板上的一个接线端。

除了定义端子，还应为 VI 定义一个形象的图标（在 VI 前面板右上角图标的右键菜单中选择图标编辑器）。当设计的 VI 作为子 VI 被主 VI 调用时，定义好的图标就会作为代表该子 VI 的功能节点，出现在主 VI 的后面板中，为子 VI 定义的输入 / 输出端子也将整合在图标周围，以便与其他功能模块或数据进行连接。和 G 语言一样，一个形象的图标可以有效增强程序的可读性。图 1-16 是调用子 VI 的例子。

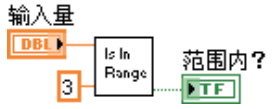


图 1-16　调用子 VI 的程序框图

在实际工作中，为每个 VI 从头设计图标往往要花费大量时间，较为有效的做法是将常用的一些图标进行整理，或者从网上下载常用的图标汇总在一起作为开发资源。这些资源不仅能缩短开发时间，同时在团队工作时，还能规范并提高开发的标准化程度。

最后，也是最为重要的一步，就是为 VI 添加文档说明。很多开发人员在项目之初并不重视文档编写，往往在项目完成后才从头为开发的 VI 或 subVI 添加说明，优秀的开发人员应避免这种陋习。试想一下，若开发一个大型、复杂、工期较长的项目，等到项目交付时，可能根本记不起来为什么要这样设计 VI。另外一种可能就是项目工期非常紧张，到最后迫于客户的压力，可能根本没时间为之前设计的 VI 创建说明，就要将设计交付给客户。当客户验收时发现缺少 VI 文档，可能会拒绝付款。最糟糕的情况是若干年后接到客户升级系统的订单时，由于缺少文档，可能连开发人员自己都已经读不懂程序了。因此，从整个项目生命周期（包含后期升级和维护）来看，养成从一开始就为 VI 添加说明文档的好习惯，总体上反而能有效缩短项目的开发和升级维护工期。另外，若一开始就为每个 VI 添加说明文档，可利用 LabVIEW 的文档生成工具，自动生成整个项目的说明文档及帮助文档。

要为 VI 添加文档，可在 VI 前面板右上角图标处右击，在弹出的菜单中选择 VI 属性选项，再从分类中选择文档，就可以在 VI 描述文本框中为其添加说明（图 1-17）。

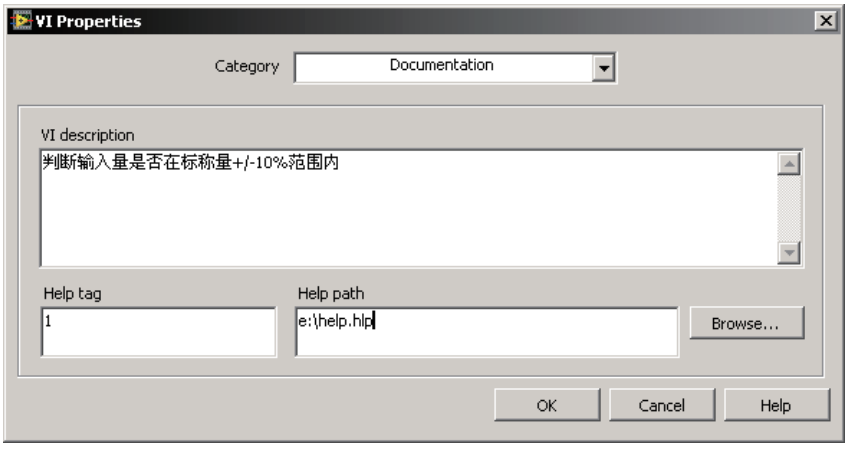


图 1-17　为 VI 添加说明

5. 运行调试

完成以上工作后，就可以对 VI 进行调试和运行了（选择前面板或后面板工具栏的按

钮即可）。

如果 VI 存在语法错误，则面板工具条上的运行按钮将会变成一个折断的箭头，表示程序不能被执行。这时该按钮被称作错误列表，单击则 LabVIEW 弹出错误清单窗口，单击其中任何一个列出的错误，选用 Find 功能，则出错的对象或端口会变成高亮。

在 LabVIEW 的工具条上有一个“高亮执行”（Highlight Execution）按钮，使其变成高亮形式，运行程序，VI 代码就以较慢的、可见的速度运行，没有被执行的代码灰色显示，执行后的代码高亮显示，并显示数据流向和连线上的数据值。这样就可以根据数据流动状态跟踪程序的执行。

为了查找程序中的逻辑错误，有时候希望框图程序逐节点执行。使用断点工具可以在程序的某一点中止程序执行，用探针或者单步方式查看数据。使用断点工具时，单击希望设置或者清除断点的位置。断点的显示对于节点或者图框表示为红框，对于连线表示为红点。当 VI 程序运行到断点设置处，程序暂停在将要执行的节点处，以闪烁表示。按下“单步执行”按钮，闪烁的节点被执行，下一个将要执行的节点变为闪烁，表示将要被执行。用户也可以单击“暂停”按钮，这样程序将连续执行直到下一个断点。

也可以用探针工具查看当框图程序流经某一根连接线时的数据值。从工具盒选择探针工具（Probe），再单击希望放置探针的连接线（或在连线上右击，在弹出的菜单中选择设置断点）。这时显示器上会出现一个探针显示窗口。该窗口总是被显示在前面板窗口或框图窗口的上面。图 1-18 给出了设置了断点、探针并打开高亮执行时的 VI 后面板示例。

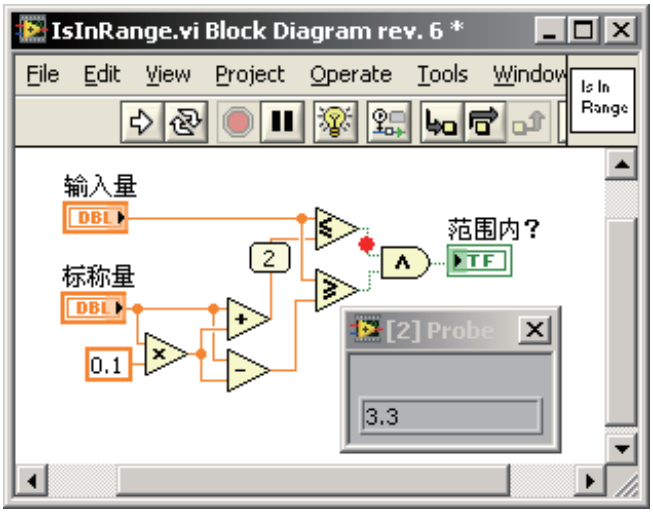


图 1-18 设置了断点、探针并打开高亮执行时的 VI 后面板

6. 错误处理

单独从这个例子来说，完成前面几步已经似乎已经完成了 VI 的开发。然而现实世界非如此简单，无论开发人员如何细心，测试验证过程如何仔细，程序还是难免有 bug。另外，一个程序在运行时总会有这样那样与设计时不一致的情况发生。例如程序要求打开的串口已经被其他进程占用，需要与之通信的设备由于某种原因死机等。如果没有在程序中采取有效的错误捕获、处理措施，就会导致程序异常或者至少降低程序的响应。因此在设计过程中，必须采取有效的错误处理措施，保证程序健壮、可用。

考虑前面的例子，你可能觉得这个 VI 简单到根本无须增加任何错误处理，但是还是有改进的地方。如果考虑这个 VI 被其他复杂的主 VI调用，那么当主 VI中出现错位时，利用

错误状态的传递，就可以通过出现错误时不执行代码来提高整个应用的响应速度（图 1-19）。

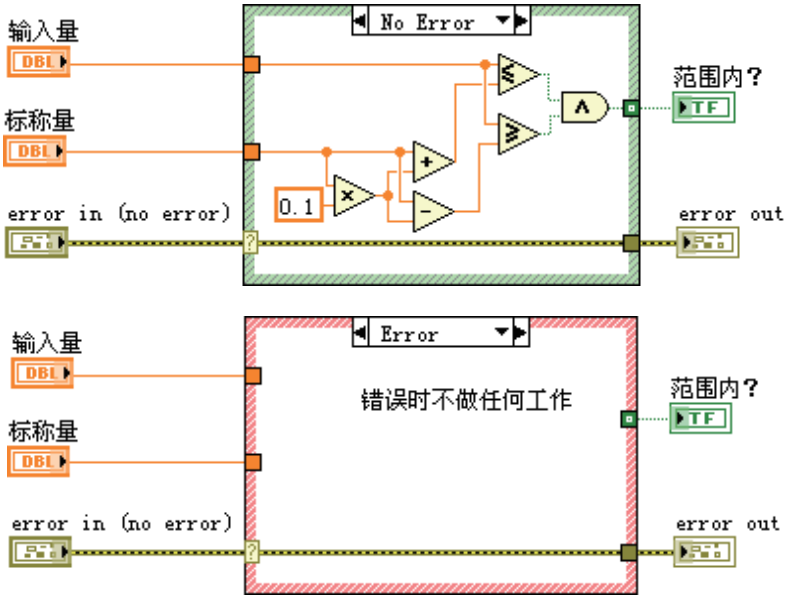


图 1-19　有错误处理能力的 VI 后面板

通过这个简单例子，读者还不能体会这种错误处理的实际意义，随着开发项目的复杂程度越来越高，程序中对错误进行处理的优越性会自然而然体现出来。鉴于错误处理措施可以保证程序健壮性并有效地提高程序可用性，因此将在第 6 章中专门讨论有关错误处理的方法。

1.4　LabVIEW 虚拟仪器项目开发

1.3 节已经介绍了 VI 开发调试的整个过程。由于 LabVIEW 提供的各种基本控件和函数均以直观的图形化形式出现，再加上 LabVIEW 完善的在线帮助文档，初学者已经能设计各种功能的 VI。然而，能快速学会编写 VI 代码与能专业、系统、高效地解决各种实际问题还相去甚远。

首先，为了构建专业的前面板，只了解 LabVIEW 的各种基本控件还不够，有时候还需要使用自定义控件、XControl、ActiveX、COM、.Net 等技术。从用户界面全局来看，还需要使用菜单、多国语言支持、动态调用 VI（Dynamic VI）等技术，以提高灵活性和可维护性。

其次，从程序设计角度来看，不仅需要研究 G 语言的基本数据类型，还要研究各种自定义数据类型，如 Cluster（簇）、Type Def./Strict Type def.（类型定义 / 严格类型定义）、Class（类）等；不仅要研究 Case（分支），Loop（循环）等基本程序结构，还要研究状态机、事件、并行循环等高级程序结构，以提高程序的执行效率和模块化程度。不仅要考虑利用自建的、可复用的代码库扩展 LabVIEW 的能力，还要考虑通过调用动态链接库 DLL、操作系统的 API 和 CIN（Code Interface Node）等扩展应用程序的能力。

再者，从系统来看，不仅要研究如何通过 TCP/IP、UDP、Data Socket 等技术来实现 Client/Server（客户端 / 服务器）结构的分布式系统，还要研究 HTTP、FTP、XML 解析。不仅要研究数据库等数据存储技术，还要研究数据加密、解密、压缩等技术。为了提高程序的可操作性和性能，还要注意前面板布局、后面板代码的风格、整个项目的可维护性和可扩展性，还必须考虑使用面向对象的方式来开发，等等。

最后，若要在诸如机器视觉、振动分析或测量控制等领域开发，还要掌握这些领域的专业知识，熟悉相关附加工具模块。当然，作为保证项目成败的关键，对整个 LabVIEW 虚拟仪器项目的管理，必须贯穿始终。这些极有助于实际工作顺利进行，正是本书各章的讲解重点。

1.5 LabVIEW 虚拟仪器项目管理

所谓项目，就是为了创造某种独特产品、服务或结果而进行的一次性努力。一般来说，LabVIEW 虚拟仪器开发项目是为了创造测试、测量、自动化或其他产品、服务或结果而进行的一次性努力。

衡量一个 LabVIEW 虚拟仪器开发项目最终是否成功的依据如下：

（1）是否满足项目要求和产品要求。

（2）是否满足相关利害关系者的需求、要求和期望。

（3）在相互竞争的项目范围、质量、进度、预算、资源、风险等因素（有时总结为时间、成本和质量三要素）之间作出权衡，最终满足项目在这些方面的目标。

为了能如期完成项目、保证用户需求得到确认和实现，并在控制项目成本基础上保证项目质量，妥善处理用户需求变更、用户的要求和期望，在整个项目执行过程中，项目管理者必须将各种知识、技能、工具和技术应用于项目活动，对项目进行管理。项目管理是快速开发满足用户需求的新产品、新设计的有效手段，也是快速改进已有设计或已经投放市场产品的有效手段。

在实际工作中，制约项目的任何一个因素发生变化，都会影响至少一个其他因素。例如，客户要求缩短开发工期，通常需要通过提高项目预算，以增加额外的资源，从而在较短时间内完成同样的工作量；如果无法提高预算，则只能缩小范围或降低质量，以便在较短时间内以同样的预算交付产品；改变项目范围或要求可能导致额外的成本或为工期带来风险。此外，不同的项目干系人可能对不同制约因素的重要程度有不同的看法。这些问题使得在实际工作中，LabVIEW 虚拟仪器项目开发和管理者不能只埋头研究算法如何完美而不顾项目的工期，不能只考虑所实现系统的性能和质量而不顾项目的成本，不能在答应客户要求的变更后忽视给项目带来的风险，等等。一个技术专家并不一定是项目中处理干系人各种需求、要求和期望、平衡相互竞争项目制约因素的专家，这也是现实生活中很多技术专家参与项目管理后并不成功的主要原因。当然如果能将优秀的专业技术和项目管理知识相结合，将有助于项目成功。

专业的项目管理通常通过将项目划分为启动、规划、执行、监控、收尾五大过程（图 1-20），并通过在各过程中识别、处理干系人的各种需求、要求和期望、平衡相互竞争的项目制约因素，确保项目获得成功。在后续章节中，我们将结合开发，对 LabVIEW 虚拟仪器项目管理知识进行讲解。

图 1-20 项目管理五大过程

从产品生命周期的角度来看，有多种开发模型可供选择。在 LabVIEW 虚拟仪器项目开发中，根据所关心的重点不同及不同模型的优缺点，可以选择一种或多种。当工期比较紧张时，可采用编码修正模型（Code and Fix Model）立即着手编码；当对项目风险比较关心时，可以采用螺旋模型（Spiral Model）不断重复开发过程，以实现不同阶段项目风险的管理；当产品质量相对重要时，可以采用传统的瀑布模型（Traditional Waterfall Model）或修正的瀑布模型（Modified Waterfall Model）等。不同的开发模型对各开发步骤进行了不同限定和定义。例如采用传统瀑布模型的项目生命周期则包含系统需求分析、软件需求分析、系统架构设计、详细设计、编码、测试、升级维护等阶段（图 1-21）。

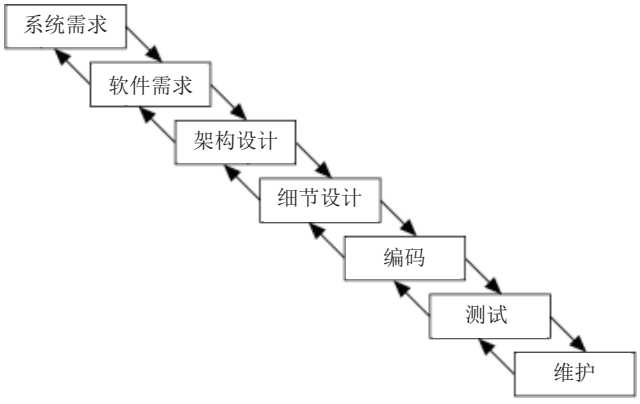


图 1-21　传统瀑布开发模型项目生命周期

NI 公司对整个虚拟仪器项目开发过程均提供相应的工具支持。例如，NI 需求管理工具包（NI Requirements Gateway）可用于管理和跟踪虚拟仪器项目的需求，以确保项目的各种需求均能被响应，它能够较好地与其他 NI 软件开发工具无缝集成。NI LabVIEW 单元测试框架工具包（Unit Test Framework Toolkit）可创建测试框架，自动进行回归测试，以完成对 VI 的功能验证。

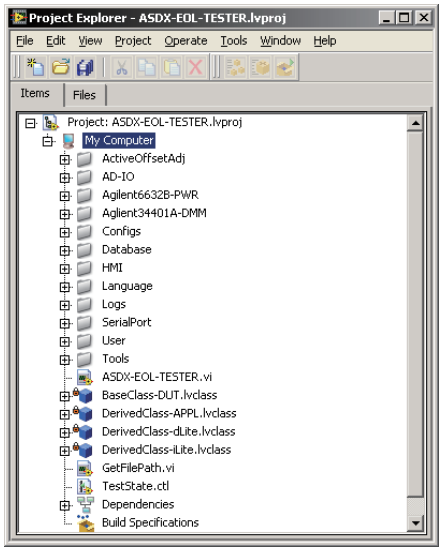


图 1-22　LabVIEW 的项目资源管理器

LabVIEW 自身也在配制管理和版本控制方面下了不少功夫。例如，从 LabVIEW 8.0 开始引入工程资源管理器（Project Explorer，如图 1-22 所示）来有效组织、管理整个项目开发过程中的各种文件。LabVIEW 广泛支持多种版本控制工具，无论是大型的 Rational Clearcase 还是开源跨平台的 CVS（Concurrent Versions System）都有较好的支持。同时，对于每个 VI，还提供 VI 版本号自动更新功能。近年来，Git 作为新一代的版本控制工具受到开发者的欢迎，通过配制，也能基于 Git 来实现对 LabVIEW 程序的版本控制。总之，通过使用各种项目阶段的支持工具包，可以较好地保证高质量的项目交付。

第 2 章　前面板设计

在虚拟仪器项目开发过程中，设计人员必须根据项目的需求，分析应用程序应采用何种方式、数据类型和结构来处理从用户界面接收到的数据，并将数据处理的最终结果通过人机界面展现给用户。虚拟仪器项目中的人机界面通过 VI 前面板展现，因此创建虚拟仪器程序时，通常第一步就是设计其前面板。

LabVIEW 中的输入控件（Control）和显示控件（Indicator）是构建前面板的核心元件，它们分别是 VI 的交互式输入和输出端口。输入控件指旋钮、按钮、转盘等输入装置。显示控件指图形、指示灯等输出装置。输入控件模拟仪器的输入装置，为 VI 的程序框图提供数据。显示控件模拟仪器的输出装置，显示程序框图获取或生成的数据。

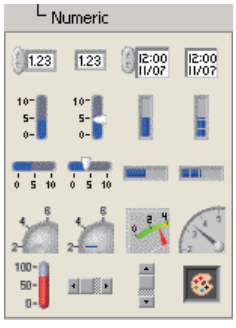
LabVIEW 提供丰富的控件用于构建用户界面，这些控件可分别对应不同精度的数据类型。在进行 VI 前面板设计时，设计人员必须将选用的控件和程序进行数据处理所需的类型、结构相结合，并遵守前面板的布局原则、文本和色彩的相关选用原则，来设计专业美观的用户界面，为后续编码打好基础。根据应用程序所需的数据类型和控件的外观，选取合适的控件，并将其进行合理布局的过程就是构建 VI 前面板的过程。

由此可见，要构建优秀的 VI 前面板，就要先了解 LabVIEW 提供的基本控件和其支持的数据类型。

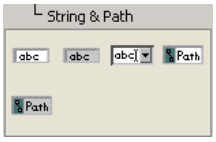
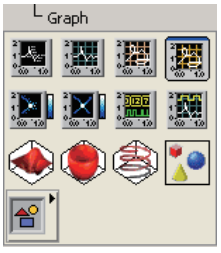
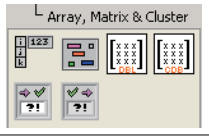
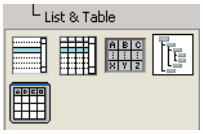
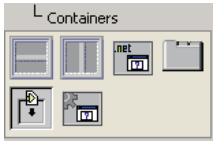
2.1　控件选择

控件是构成前面板的核心元件，LabVIEW 提供的控件位于前面板控件选板上（在前面板上右击即可弹出控件选板）。LabVIEW 支持的基本控件如表 2-1 所示，主要包括数值输入控件（如滑动杆和旋钮）、数值显示控件（如仪表和量表、图表）、布尔控件（如按钮和开关）、字符串、路径、数组、簇、列表框、树形控件、表格、下拉列表控件、枚举控件和容器控件等。

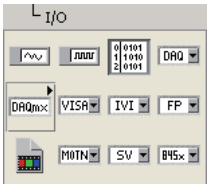
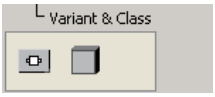
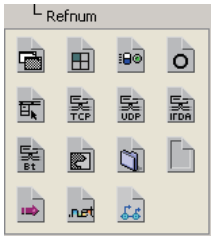
表 2-1　LabVIEW 支持的基本控件和显示控件

控件和名称	分类名称 / 图例	说　明
	+ 数值框（Numeric）	接收用户输入或进行数据显示最便捷的方式
	+ 滚动条和状态条（Slides、Scroll、Progress & Graduate Bars）	（1）一般用于指示工作进度状态。 （2）本质上属于可进行数值调整的数值控件。 （3）可同时接收或显示多个数值
	+ 旋钮和拨号按钮（Knobs & Dials）	（1）本质上属于可进行数值调整的数值控件。 （2）可同时接收或显示多个数值
	+ 时间戳（Time Stamps）	用于程序与界面之间进行日期和时间数据的交互
	+ 色阶（Color Ramp） 	（1）使用颜色代表某个数字或范围。 （2）适合数据超出正常范围时报警
	+ 颜色框（Color Box） 	（1）用于接收用户对颜色的选择或显示颜色给用户。 （2）使用十六进制数 0xRRGGBB 表示红、绿、蓝组成的颜色，RR 代表红色值，GG 代表绿色值，BB 代表蓝色值

续表

控件和名称	分类名称 / 图例	说　明
	+ 按钮（Buttons） + 开关（Switches）	用于创建按钮和开关输入控件
	+ LED	用于创建 LED 等显示控件
	+ 单选框（Radio Box）	（1）从列表中单选。 （2）为枚举类型，可通过 Case 来处理
	+ 复选框（Checkboxes）	（1）可表示 TRUE/FALSE。 （2）可配置为三选框
	+ 文本框 （Text Entry Boxes）	用于创建文本输入控件或字符输出显示控件
	+ 组合框 （Combo Box）	（1）用于创建可选字符串的文本输入和选择框。 （2）数据类型为字符串
	+ 路径 （Path Control & Indicator）	创建用于路径选择的输入控件和路径输出显示控件
	+ 波形图（Graphs）	（1）先缓冲所有数据至数组中，随后一次性绘图。 （2）绘制时，丢弃以前绘制过的数据，显示新数据。 （3）通常用于数据连续采集时曲线快速绘制
	+ 波形表（Charts）	（1）在已绘制数据后添加新数据。 （2）可以实时查看读取的数据和历史数据。 （3）当超出显示范围时，自动滚屏。 （4）通常用于数据量较小、但需实时查看的情况
	+ 数组（Array）	创建类型相同的一组数据
	+ 簇（Cluster）	创建多种类型混合的一组数据
	+ 矩阵（Matrix）	创建用于数学计算的实数或复数矩阵
	+ 列表框（Listboxes）	创建单选或多选列表框
	+ 树形列表框（Tree）	创建分层级的列表框
	+ 表格（Tables）	以表格形式展示数据
	+ 下拉环（Ring）	（1）将图片或字符串映射至数值的列表框。 （2）映射的数值可随意指定。 （3）可在所列互斥条目之间进行单选或多选
	+ 枚举列表（Enumerated）	（1）类似下拉环，但数据类型为枚举型。 （2）映射的数值自动按顺序生成
	+ 分栏 （Horizon/Vertical Splitter）	用于将页面分为不同几栏
	+ 表单（Tab）	用于将多个在同一阶段使用的控件分页显示
	+ 子面板（Subpanel）	用于在前面板显示另一个 VI 的界面

续表

控件和名称	分类名称 / 图例	说　　明
	+ 硬件 I/O 名（I/O Name）	用于与 DAQ、 VISA、IVI 等仪器进行通信的逻辑名称
	+ 变量（Variant） + 类（Class）	用于与变量和类对象进行交互的控件
	+ 对象参考（Objects References） + 程序参考（Applications Reference）	（1）用于操作文件、目录、计算机虚拟设备、网络连接的逻辑控件。 （2）本质上为临时指针。 （3）操作前必须先打开（分配内存），操作结束后需关闭（释放内存）
	+ .NET 控件 + ActiveX 控件	（1）用于操作 .NET 控件和 ActiveX 控件。 （2）可自行加载更多系统支持的 .NET 控件和 ActiveX 控件

开发人员从控件面板中选择需要的控件放入前面板来构建用户界面，并可通过被选控件的右键菜单中的“变更为输入控件”（Change to Control）和“变更为显示控件”（Change to Indicator）选项切换其输入或输出属性。例如，若选择了一个数值输入框放入前面板，则可以通过其右键菜单中的“变更为显示控件”选项，将输入控件的属性变更为显示控件。变更为显示控件后，输入框只能用于进行数据显示，不接受用户的任何输入。

进行前面板设计的第一步就是按照以下原则选择用于构建用户界面的控件：

（1）所选控件构成的用户界面应能兼容目标显示设备和目标操作系统。

（2）结合所开发应用的内容，所选输入控件应容易理解、便于操作。

（3）所选输出显示控件应能直观、有效的向用户表达相关数据。

（4）控件所代表的数据类型应能兼顾 VI 进行数据处理所需的范围、精度和效率。

（5）控件数据类型对应的操作应能满足程序数据处理的要求。

大多数的 LabVIEW 基本控件有传统（Classic）、现代（Modern）和系统（System）三种版本。原则上讲，不同版本的控件只是外观不同，可根据所开发应用使用的场合选用不同版本。若因某种原因（如为了兼容同一台机器上另外一个旧版本的程序），开发的 VI 必须运行在设置为 16 色或 256 色的显示器上，就可以选择传统版本的控件。现代版本的控件相对于传统版本控件对显示器的色彩设置有较高的要求（至少 16 位彩色），使用现代版本控件设计的 VI 前面板在较低配置的显示器上显示会比较难看。反过来说，如果必须兼容不同色彩的显示设置，那么选择传统版本的控件较为稳妥。另外，如果用户对传统风格的界面相对比较熟悉，或认为传统风格的界面比较美观，也可以选用传统版本的控件。

系统版本的控件按照用户对操作系统的色彩设置来显示，一般用来专门设计对话框。系统版本的控件会自动匹配其父窗体的颜色。例如，父窗体的色彩为白色，则在该窗体上的系

统版本控件会自动与白色背景匹配，实现较好的显示效果。系统版本控件的外观依赖 VI 所运行的平台。当 VI 在不同的系统上运行时，系统版本控件的显示会自动进行调整，与所运行操作系统的标准对话框进行匹配。

除了外观，控件选用的另一个重要原则就是其所代表的数据类型和该类型对应的操作（LabVIEW 所支持的数据类型和数据操作将在第 3 章详细讲解）。选用控件所代表的数据类型应能兼顾 VI 进行数据处理所需的范围、精度和效率。然而这个看似简单的原则在实际设计过程中并不那么容易执行。没有哪个用户“需要解决电流采集问题”时会说“我需要解决单精度浮点类型电压量的采集问题”这样的话。将用户需求抽象为合适的类型数据，为代表实际事物的数据选择适当精度，并合理使用计算机资源，确定高效的数据类型是设计人员的责任，更是一门艺术。除了控件所代表的数据类型外，关注数据类型对应的操作也非常重要。例如，在 LabVIEW 中选择了数值类型，相应的也就选定了对应数值类型的加、减、乘、除、数据格式转换等操作，对应的数据类型操作手段越丰富，程序设计越快速灵活。

基于需求分析决定了要使用的控件后，就可以从控件箱中将所选控件拖放至前面板中，开始着手前面板设计。几乎所有被拖放至前面板的控件都有标签（Label）、标题（Caption）、可见状态（Visible）、使能状态（Enable State）、大小（Size）和颜色（Color）等通用属性。标签是控件以及与之对应的变量、常量等对象在前、后面板的唯一标识。也就是说，一旦设计人员变更了控件的标签，那么与之对应的常量名、变量名等对象也会随标签的变化而变化；控件的标题则只出现在前面板，它可在控件标签不变的情况下任由设计人员修改。标题的这种特性，可使设计人员在不变更对象名的前提下，对 VI 前面板进行修饰说明或本地化（在后续章节会详细讲解）。此外，若要使程序支持 Unicode 编码方式，则必须在前面板中使用标题来说明控件。因此，建议在设计时采用标题来说明所有前面板中的控件。例如图 2-1 中，控件在前、后面板中均使用了 Progress 作为标签，而在前面板中则使用“测试进度”作为标题。与控件标签对应，在 LabVIEW 中还有一种称为“自由标签”（Free Label）的控件，它用于在前、后面板的任何空白处添加程序的说明或注释（选择文本工具后，在前面板或后面板空白处单击，然后输入注释即可）。控件使能状态或可见状态用于控制控件对用户是否可见或可操作，而通过控件大小和颜色属性可控制控件在前面板的外观。

图 2-1　控件的标签和标题示例

除了通用属性外，不同控件还有千姿百态的“个性”。以按钮控件为例，在设计时不仅可以通过按钮的属性对话框（通过控件的右键菜单的“属性”选项打开“属性”对话框）对通用的属性进行设置，还可以设置仅属于按钮的按下或弹出文本（Button On/Off Text）、按钮行为（Button Behavior）方式等属性，如图 2-2 所示。

控件的属性不仅可以在设计时通过属性对话框进行修改，还可以在运行时通过控件的属性节点（Property Node，控件的右键菜单中选择“创建”→“属性”→“节点”选项）进行修改（这些内容将在后续章节详细介绍）。

如表 2-1 所示，LabVIEW 提供多种控件，正是这些控件的通用属性和不同特性构成了 LabVIEW 可快速开发的基础。鉴于 LabVIEW 在线帮助对这些控件使用有详细说明，这里不再一一说明，我们将在后续章节中结合例子，对部分重要控件的使用进行讲解。对读者来说，越是了解这些控件，开发速度就越快，熟悉这些控件使用的过程是学习 LabVIEW 必不可少的一个阶段。

一般来说，完成控件选择和属性设置后，还需要把逻辑上相近的控件组织在一起，并对

它们在前面板上的布局进行调整，必要时还需配以辅助修饰线、图片等，来完成 VI前面板的设计。

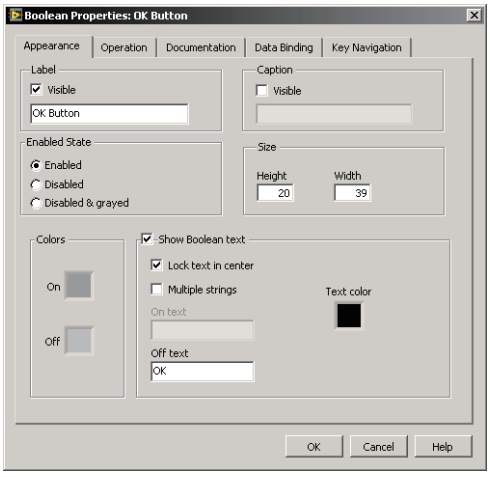


图 2-2　通过属性对话框设置控件属性

2.2　前面板设计总原则

无论对于何种 VI前面板，其主要目的是用于与用户或开发者进行交互，以最终实现用户要求的功能。也就是说，对界面布局的调整、美化、修饰都应以提高交互效率、实现用户任务为目的。任何牺牲交互性的界面设计都是失败的。评判前面板设计的好坏是个比较主观的问题，不同的使用环境、不同的用户群可能有不同的评价标准。虽然很难有统一创建 VI前面板的标准，但还是有一些常规经验供设计时参考。

表 2-2 列出了 VI前面板设计总原则，这些原则贯穿整个前面板设计过程。

表 2-2　前面板设计总原则

分类	设 计 原 则	工具 / 技巧
总原则	功能为主，表示为辅	实现用户要求的功能是任何界面设计的基础
	区分使用场合和目标客户	● 桌面应用使用标准对话框和系统控件； ● 工业应用使用自定义对话框和流行（3D）控件
	只传递用户需要的信息	● 程序的问题程序内部解决； ● 使用下拉环或枚举框将数据与文本图片等信息进行映射，替换难理解的数字
	限定单个页面信息的数量，必要时提供总览或导航页	● 使用 Tab 对页面进行分页； ● 使用动态 VI 加载

设计人员必须时刻牢记设计 VI 的目的是实现用户要求的功能。换句话说，如果用户要求的功能尚未实现，无论花多少时间和精力，设计出的漂亮界面都是华而不实。由于不同场合和用户群的使用习惯各不相同，因此在设计前应根据这些因素决定采用何种设计风格。例如由于桌面应用一般在办公室或实验室等较好环境下，由熟悉计算机的人群使用，因此可使用标准对话框和系统风格控件进行设计；而对工业应用来说，一般使用在生产线、户外等较差环境下，并由操作工使用，则可选用自定义对话框和流行的 3D 控件来增强显示效果。

很多设计人员习惯“从内到外”进行思考，从自己进行编码的容易程度来主观决定哪些信息应当显示给客户，然而客户并不知道程序内部如何工作，他们只关心如何利用软件快速高效地完成手头工作，当不能很快上手时，就会不断抱怨。因此，设计前面板时要改变思路“由外到内”进行设计，只把用户需要的信息按照用户的语言显示出来，把内部处理所需的数据与用户需要的信息分离开来，做到程序的问题程序内部解决。另外，使用下拉环或枚举框将数据与文本图片等信息进行映射，用有意义的文本或图片代替晦涩的数字也能较好提高可读性。如图 2-3 所示，要显示某种类型为 31014 的集成电路产品，使用该产品的图片比直接给出一串数字要直观得多，而且还能传递诸如封装形式、针脚数量等更多信息。

图 2-3　用图片代替晦涩的数字

限定显示在界面上的信息数量能很快使用户将注意力集中在重要信息上。在同一页提供大量的信息只能使用户无从下手。如果需要显示的信息的确非常多，可以将逻辑上或功能上相关、但相对其他信息较为独立的部分用 Tab 控件分页显示。若有必要，还可以增加一个“概览”（Overview）页面，将总体情况或重要的信息显示给客户，细节部分可随后逐页查看，如图 2-4 所示。

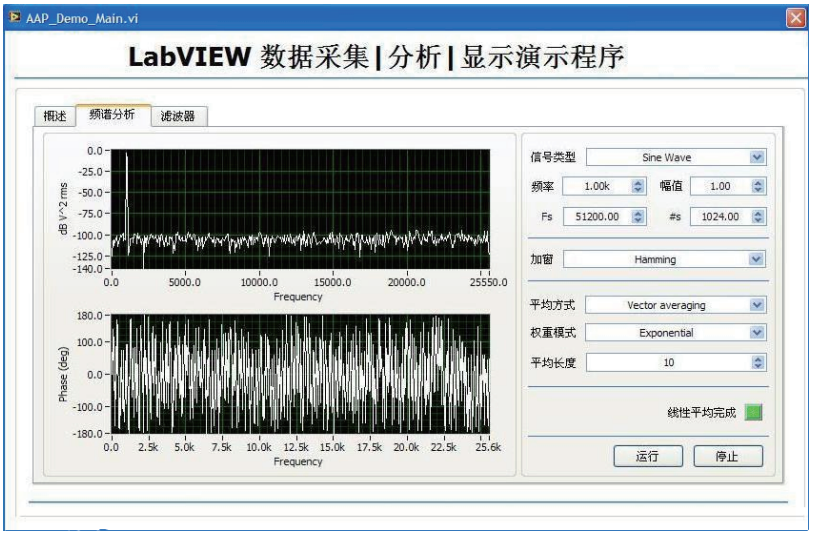


图 2-4　使用 Tab 将信息进行分类，并添加概览页

前面板设计时另一个重要的总体原则是界面的一致性。保持一致的界面设计不仅可以使程序显得专业，更重要的是可以缩短操作人员上手的时间。由于软件界面的一致性，用户可在使用软件的某一部分后，可快速地将操作方式沿用到其他模块。

表 2-3 列出了前面板设计的一致性原则。要做到软件界面一致，可以通过使整个应用程序的风格一致、使功能相近的控件大小、颜色尽量一致以及设计时沿用通用习惯、惯例或行业规范来实现。使整个应用程序的风格一致，意味着在整个应用程序界面中要使用相同风格的控件、相同类型的字体和相同配色方案等。除非要刻意达到某种着重强调的目的，否则使用与整个界面风格不一致的控件会显得很突兀。在保证与整体风格一致的前提下，使功能相近的控件大小、颜色尽量一致，可以有效增强软件的可读性和信息传递的有效性。在设计时可以使用控件尺寸调整工具（Resize Objects）和颜色设置工具（Set Color）来调整尺寸、颜色等。设计时沿用通用习惯或惯例尤为重要。例如，绿色通常代表正常通过，黄色代表告警，红色代表故障，如果在设计中不沿用这些惯例，用户会觉得很迷惑。在实际开发过程中，

行业规范并不像红、绿、蓝那么简单，通常必须针对特定行业，通过研究规范或与客户进行大量沟通来确定软件界面的最终显示。

表 2-3　前面板设计的一致性原则

分类	设 计 原 则	工具 / 技巧
一致性	整个应用程序保持一致风格	• 使用相同风格的控件； • 使用类型字体； • 使用相同配色，方案一致
	功能相近的控件的大小、颜色尽量一致	• 使用尺寸调整工具（Resize Objects）； • 使用颜色设置工具（Set Color）
	沿用通用惯例或行业规范	沿袭红、绿、蓝通常代表的意义等

2.3　前面板布局

用户界面是进行信息交互的重要场所。考虑程序整体以何种途径收集和向用户显示各种信息是界面设计的一项重要工作，也是程序界面框架设计的主要步骤。通常在程序设计之初确定程序界面的总体框架和布局。

大多数虚拟仪器项目以对话框的形式来组织信息交互。当然也可以选择 LabVIEW 提供的其他方式（图 2-5）来搭建用户界面框架。例如，可以使用纵向分隔条（Vert Splitter Bar）、横向分隔条（Hor Splitter Bar）创建分栏形式的主界面；可以使用选项卡（Tab Control）创建分页形式的主界面；还可以使用子面板（SubPanel）在项目窗口中调用不同的 VI，等等。

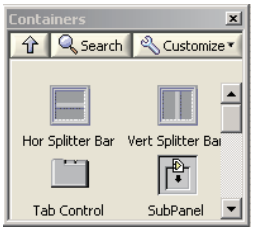


图 2-5　LabVIEW 提供的界面框架工具

确定界面的整体框架后，就可以选取控件并安排其在前面板上的位置。前面板上控件或控件分组的布局是前面板设计的关键和重要工作之一。

当按照需求确定了要使用的控件后，可以首先使用一些装饰部件（如组合框、矩形框等）或簇将功能或逻辑相关的控件进行分组，随后使用对齐工具（Align Objects）和排列工具（Distribute Objects）来摆放这些控件或控件的分组。表 2-4 列出了前面板布局的一些基本原则。

表 2-4　前面板布局原则

分类	设 计 原 则	工具 / 技巧
布局	对称排列控件，并留有足够、等距间隔	• 使用对齐工具（Align Objects）； • 使用排列工具（Distribute Objects）；
	将功能或逻辑相关的控件进行分组	• 使用框架（Frame）； • 使用簇
	按照常用度（频度）或自然顺序摆放控件或控件分组	• 按照使用频率、重要性摆放控件； • 按照使用习惯、自然顺序摆放控件； • 子 VI 输入 / 输出控件摆放与 I/O 端子定义一致
	使用辅助控件修饰、美化界面	• 使用公司图标； • 用背景图片作装饰

在摆放过程中一般遵循“控件对称排列”的原则，并使各控件或控件分组之间留有足够、等距离的间隔。控件和控件分组在前面板上总体排放位置则由常用度（频度）或自然顺序来定。例如可以将使用频率较高、较重要的控件或控件分组放在显眼的位置；可以按照人们习惯的、从左到右的阅读习惯来摆放逻辑分组（某些西方国家习惯从右至左）；可以将子 VI 前面板上用于输入 / 输出的控件与 I/O 端子定义的布局对应一致等。在整个过程中可以使用分隔线等装饰控件进一步区分不同逻辑显示单元，最后还可以在前面板上添加公司的图标或使用背景图片来美化界面。

图 2-6 给出了一个简单的例子。在这个例子中，首先采用一致风格的字体和配色；其次，使用从左到右的习惯来安排总体布局，在左边放置采集（Acquire）、分析（Analysis）和演示（Present）操作选择按钮，在右边安排信息显示和调整按钮。另外，由于显示的信息较多，使用了 Tab 控件将信息进行分类，并使用一个“概述”页对信息进行汇总。“基本控件”页左下方的按钮用于采集控制，因此将其放在一起，使用相同大小、相等间隔进行排列；右边使用组合框架，将相关的控件进行分类后对称摆放，既美观又清晰。最后，在页面上添加公司图标和“关于”按钮，不仅美化了界面，还增加了关于软件更详细的信息索引。整个界面布局合理，条理清楚，是一个值得参考的例子。

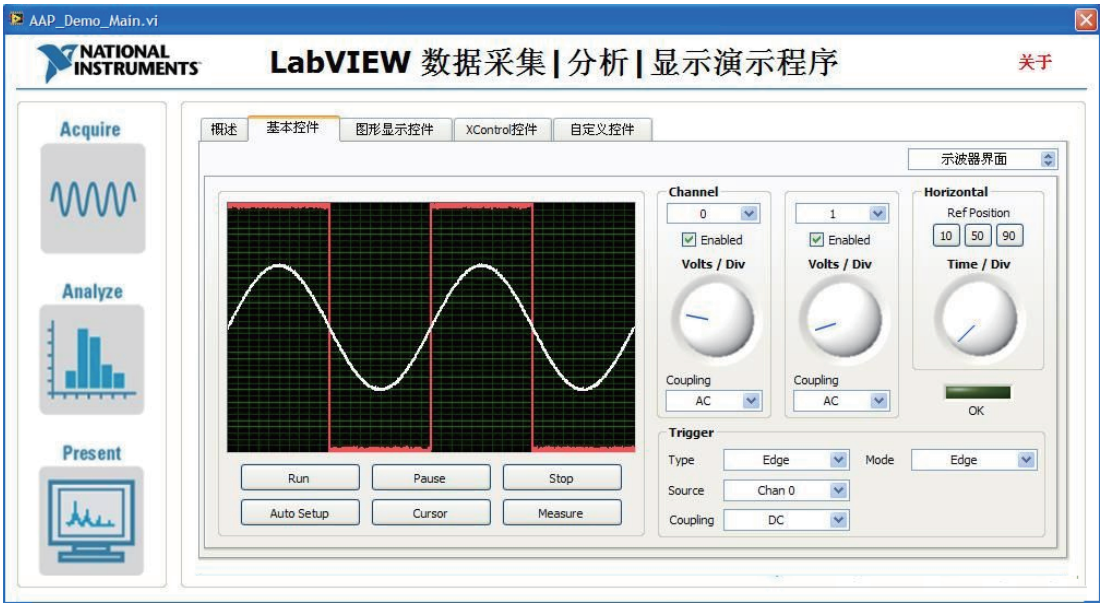


图 2-6　页面布局的简单例子

进行 VI 前面板设计时，按照其用途可将所设计的 VI 前面板分为和用户进行交互的用户界面（GUI）VI 和子 VI 两大类。由于子 VI 通常只是被主 VI 载入内存，运行后实现某种模块化的功能，因此其前面板一般不呈现给用户，而是被开发人员用来定义 VI 的输入 / 输出参数或进行调试诊断。基于这种考虑，笔者一般将子 VI 中的控件按照输入 / 输出的功能逻辑进行分类组织，并将其前面板背景色设置为与用户界面 VI 不同的较深的颜色，以示区别。

图 2-7 给出了一个子 VI 前面板的简单例子。其中虽然只包含一个错误簇输入控件和错误簇显示控件，但是还是使用了分类框将输入 / 输出控件分成两类，并将输入控件放置在左边，输出控件放置在右边，以便和子 VI 连线板的布局尽量保持一致。另外，子 VI 的背景色也被设置为和用户界面 VI 不同的深灰色。采用以上方式组织子 VI 中的控件，可以使设计人员对子 VI 的输入 / 输出参数一目了然。

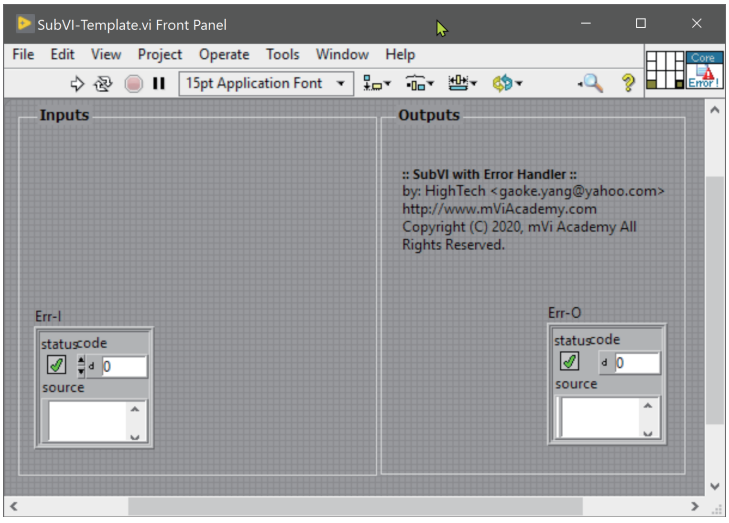


图 2-7　子 VI前面板布局

2.4　前面板文本和色彩

前面板上的文本是向用户传递信息的基本途径，其重要性不言而喻。一般来说 LabVIEW 中的文本通过以下几种方式展现出来。

1. 标签

标签（Label）是前面板和程序框图对象的唯一标识。LabVIEW 有两种标签：自带标签和自由标签。自带标签属于某一特定对象，相当于文本语言中的变量名称，随对象移动。自带标签可单独移动，但移动该标签的对象时，标签将随对象移动。可对自带标签进行隐藏，但无法在不影响自带标签所属对象的前提下，单独复制或删除自带标签。

2. 自由标签

自由标签（Free Label）不附属于任何对象，用户可独立创建、移动、旋转或删除自由标签。自由标签用于在前面板和程序框图上添加注释，也可用于注释程序框图上的代码以及在前面板上列出用户指令。双击空白区域或使用标注工具可创建自由标签，或编辑任何类型的标签。

3. 标题

标题（Caption）用于在前面板上对对象进行描述，它不像标签那样影响程序代码。标题与标签的不同之处在于标题不会影响对象的名称，因此常用于在前面板上为对象添加表述，支持 Unicode 文本，实现程序的多语言。标题仅在前面板上出现，而且在创建应用程序的本地化版本时必须用到它。

4. 布尔文本

布尔控件除了有标签和标题外，还具有布尔文本（Boolean Text）标题。布尔文本标题随输入控件或显示控件的值而改变。初始状态下，布尔控件在 TRUE 状态下标注为“开”，FALSE 状态下标注为“关”。

在前面板上使用文本时应注意使用准确、精简、友好的语言。实际中该如何理解“准确、精简、友好”呢？所谓准确是指从用户角度来看，只要使用者能清楚地知道文本的意思，就说明文本使用是准确的。因此要习惯在设计界面时使用用户的语言。精简意味着界面上不能大段地放置文本说明，那样只能干扰用户抓住重点信息，可以使用一些代表性的词语，然后使用提示（Tips）工具进一步显示较详细的信息给客户。如果需要显示的说明信息成段成篇，

则可以使用在线帮助的方式提供给用户。

表 2-5 列出了一些前面板文本设计原则。根据使用场合的不同，使用规范的字体大小、颜色和大小写格式的文本是较为常用的做法。例如在工业场合下，一般使用较大的粗体字，以便观察。值得一提的是，在控件标签的末尾显示控件的默认值，也是一种对用户友好的做法。

表 2-5　前面板文本设计原则

分类	设 计 原 则	工具 / 技巧
文本	使用准确、精简、友好的语言	● 使用用户的语言； ● 使用代表性的词语； ● 使用提示（Tips）； ● 将详细说明汇总到帮助文件； ● 控件标签末尾显示默认值
	采用规范的字体大小、颜色和大小写格式	● 使用字体设置（Text Setting）工具； ● 为应用于工业场合的程序使用大字体、高对比度的背景和字体色彩

好的配色方案可以有效传递信息。前面板设计时使用的色彩方案应尽量简单，同时整个应用程序的配色应统一。表 2-6 列出了一些前面板色彩设计的原则。在实际设计中，同一套配色方案一般使用不超过 3 ～ 4 种互补色彩，根据使用场合的不同，前景和背景色彩对比度也不同。对子 VI 设计来说，由于一般不显示前面板给用户，因此一般统一使用一种与主 VI 区别开来的背景色彩；在工业 VI 中，使用高对比度的背景色和前景色配色方案，可使用户在有干扰的情况下清楚地看到显示的信息，而桌面 VI 则以美观为主。

表 2-6　前面板色彩设计原则

分类	设 计 原 则	工具 / 技巧
色彩	使用简单、统一的配色方案	● 整个应用采用同一种配色方案； ● 同一套配色方案使用不超过 3 ～ 4 种互补色彩； ● 为工业 VI 选用高对比度的背景色和前景色配色方案； ● 子 VI 面板背景色彩统一并与主 VI 区别开

图 2-8 列出了笔者常使用的一些颜色（分别以十六进制数 0xRRGGBB 的形式列出红、绿、蓝三色值，RR 代表红色值，GG 代表绿色值，BB 代表蓝色值），供读者参考。

红、绿、蓝三色值十六进制编码

第 1 行从左至右颜色值：

0x9DC338	0x9ACEF7	0x9DC338	0xFF9700	0x900000
0xFF9700	0x016A99	0xA566AD	0x74B46A	0xF7582E
0xEEF2F5	0x6BD6FA	0xAFDF5A	0xFCDE59	0xCA0005
0x4B7B22	0x648DB6	0xB70000	0x800000	0xf5B55A
0x99CCFF	0x009999	0xFCA13D	0xC4240C	0x894611

图 2-8　颜色与相应的 RGB 值

第 2 行从左至右颜色值：

0xE1EDC3	0xE1F0FD	0xFFE0B2	0xE1F0FD	0xDDB2B2
0xFFE0B2	0xB2D2E0	0xE4D1E6	0xD5E8D2	0xFDCDC0
0xEEF2F5	0x8BDFFB	0xD9FC9A	0xFEEFA0	0xF1F1F1
0xA9CD75	0xF1F1F1	0x999999	0xE6E6E6	0xCCCCCC
0x999999	0xFFFFFF	0xFDB95C	0xF2F2F2	0xDBC7B7

图 2-8（续）

2.5　可见性和健壮性

表 2-7 列出了一些前面板控件可见性设计的原则。

表 2-7　前面板设计的可见性原则

分类	设 计 原 则	工具 / 技巧
可见性	保证显示给用户的信息可见	避免对象重叠
	慎重使用隐藏功能	尽量避免使用动态菜单
	合适的窗口初始位置和状态	为工业应用最大化界面
	隐藏 LabVIEW 的工具条和菜单	如有必要，为应用单独定制菜单

设计 VI 界面的目的是与用户交互，如果由于某种原因，设计的成果不能呈现给用户，那么再漂亮的设计也是枉然，必须通过各种技术确保最终设计对客户可见。实际中不把界面呈现给用户的错误大家都很少犯，但是 VI 上的对象重叠却经常能见到。例如为 VI 选择了“窗口缩放时自动缩放前面板对象尺寸”（Scale all objects on front panel as the window resizes，见图 2-9）选项，但是却混用了不同类型的控件，就很容易导致对象重叠。

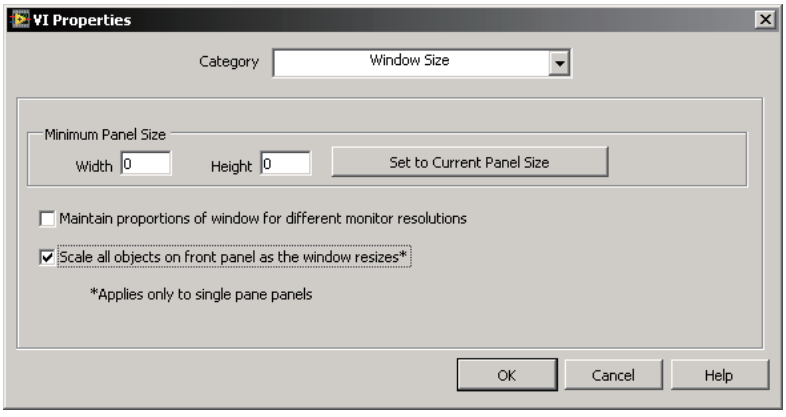


图 2-9　“窗口缩放时自动缩放前面板对象尺寸”选项

慎重使用隐藏功能对于可见性也很重要。笔者曾经参与一个项目，在设计时，供应商的设计团队提议在主界面上使用“动态菜单”功能。使用这种菜单的菜单选项会随用户使用软件的环境或选择对象的不同而变化。起初大家都觉得这是个很不错的主意，但是软件交付后，很多用户在刚使用时就抱怨他们之前能选择的菜单项不见了。当供应商告诉客户要花些时间学习软件如何运作时，他们很失望，在学完之前，客户已经对软件失去了耐心。在程序中过多地使用隐藏功能只能使用户对软件的使用更迷惑。好的处理办法是使用对象的“禁用”（Disable）功能，

在用户不具备使用权限时，对象将变灰且不接受用户的输入，这样可以保证对象始终可见。

保证 VI 运行时窗口的初始位置极为重要。例如在调试时，基于某种显示分辨率，窗口可以正常显示，但是，当最终用户调整分辨率时，测试时正确显示的窗口可能已经不在显示器范围内了。可以通过以下方法来解决这个问题。

（1）对于工业化的 VI 来说，通常会将 VI 初始窗口最大化，并禁用窗口缩放或关闭功能。这样可以避免用户空闲时在计算机上进行其他不相关的操作（有时候甚至会禁用操作系统的开始菜单）。

（2）设计时就考虑运行 VI 的目标计算机配置，并设置开发 VI 的计算机的分辨率与目标计算机的分辨率相同。

（3）使用代码确保相同分辨率。在程序开始运行时，使用代码更改目标计算机的分辨率与开发代码的计算机的分辨率一致，程序退出前，恢复目标计算机的原有分辨率设置。

值得一提的是，在应用程序中隐藏 LabVIEW 的菜单栏和工具栏是较为流行的做法，留着它们只能使用户迷惑。如果真的需要使用菜单，可以为应用程序定制属于自己的菜单栏。

与可见性相同，用户界面的健壮性也是前面板设计时需要考虑的重要因素。表 2-8 列出了一些前面板健壮性设计的原则。

表 2-8　前面板设计的健壮性原则

分类	设 计 原 则	工具 / 技巧
健壮性	减少输入错误或误操作	● 规定控件数据范围； ● 采用下拉环或者枚举框减少错误输入； ● 使用禁用（Disable）功能

保证应用程序的健壮性是设计人员的责任。设计人员必须千方百计减少用户的错误输入或误操作的可能性。有很多方法可以提高软件的健壮性，例如规定控件的输入范围、用下拉环或者枚举框代替一般的数值输入或使用组件的“禁用”功能，等等。

以上对 VI 前面板的设计原则进行了分类讨论。关于界面的设计方法不仅限于以上内容，读者可以这些内容为基础在实践中逐步进行汇总和扩充，表 2-9 是这些原则的汇总。

表 2-9　前面板设计的原则和技巧

分类	设 计 原 则	工具 / 技巧
总原则	功能为主，表示为辅	实现用户要求的功能是任何界面设计的基础
	区分使用场合和目标客户	● 桌面应用使用标准对话框和系统控件； ● 工业应用使用自定义对话框和流行（3D）控件
	只传递用户需要的信息	● 程序的问题程序内部解决； ● 使用下拉环或枚举框将数据与文本图片等信息进行映射，替换难理解的数字
	限定单个页面信息的数量，必要时提供总览或导航页	● 使用 Tab 对页面进行分页； ● 使用动态 VI 加载
一致性	整个应用程序保持一致风格	● 使用相同风格的控件； ● 使用类型字体； ● 使用相同配色方案
	功能相近的控件的大小、颜色尽量一致	● 使用尺寸调整工具（Resize Objects ）  ； ● 使用颜色设置工具（Set Color ） 
	沿用通用惯例或行业规范	沿袭红、绿、蓝通常代表的意义

续表

分类	设 计 原 则	工具 / 技巧
布局	对称排列控件，并留有足够、等距间隔	● 使用对齐工具（Align Objects）； ● 使用分布工具（Distribute Objects）
	将功能或逻辑相关的控件进行分组	● 使用框架（Frame）； ● 使用簇
	按照常用度（频度）或自然顺序摆放控件或控件分组	● 按照使用频率、重要性摆放控件； ● 按照使用习惯、自然顺序摆放控件； ● 子 VI 控件摆放与 I/O 端子定义一致
	使用辅助控件修饰、美化界面	● 使用公司 Logo； ● 用背景图片作装饰
文本	使用准确、精简、友好的文本	● 使用用户的语言； ● 使用代表性的词语； ● 使用提示（Tips）； ● 将详细说明汇总到帮助文件； ● 子 VI 标签末尾显示默认值
	采用规范的字号大小、颜色和大小写格式	● 使用字体设置（Text Setting）工具； ● 为工业应用大字体、背景和字体色彩高对比度
色彩	使用简单、统一的配色方案	● 整个应用采用同一种配色方案； ● 同一套配色方案使用不超过 3 ～ 4 种互补色彩； ● 为工业 VI 选用高对比度的背景色和前景色配色方案； ● 子 VI 面板背景色彩统一并与主 VI 区别开
可见性	保证显示给用户的信息可见	避免对象重叠
	慎重使用隐藏功能	尽量避免使用动态菜单
	合适的窗口初始位置和状态	为工业应用最大化界面
	隐藏 LabVIEW 的工具条和菜单	如有必要，为应用单独定制菜单
健壮性	减少输入错误或误操作	● 规定控件数据范围； ● 采用下拉环或者枚举框减少错误输入； ● 使用禁用（Disable）功能

2.6　对前面板进行装饰

如果把界面设计比作盖房子，那么前面几节讲解的界面设计原则就如同是盖房子的主体结构的必需工具。完成“主体结构”后，还需要对房子进行装修，才能更舒适、漂亮。LabVIEW 提供了一些基本的用户界面装饰控件，如线条、边框、箭头等（图 2-10）。

设计人员可以使用这些控件对用户界面进行装饰，使用户界面更直观易懂。例如，可以使用箭头连接按顺序操作的各步骤，引导用户完成整个测试流程。除了基本的修饰控件外，还可以使用图片对用户界面进行装饰。值得注意的是，此处所说的“装饰”，并不仅局限于使用 LabVIEW 提供的装饰控件对界面的局部进行修饰，也可以使用 Photoshop、Paint.NET、PowerPoint 和 Visio 等软件，将外部图片或矢量图导入前面板中，来完成各种信息的组织和前面板装饰，以创建专业、个性的用户界面。

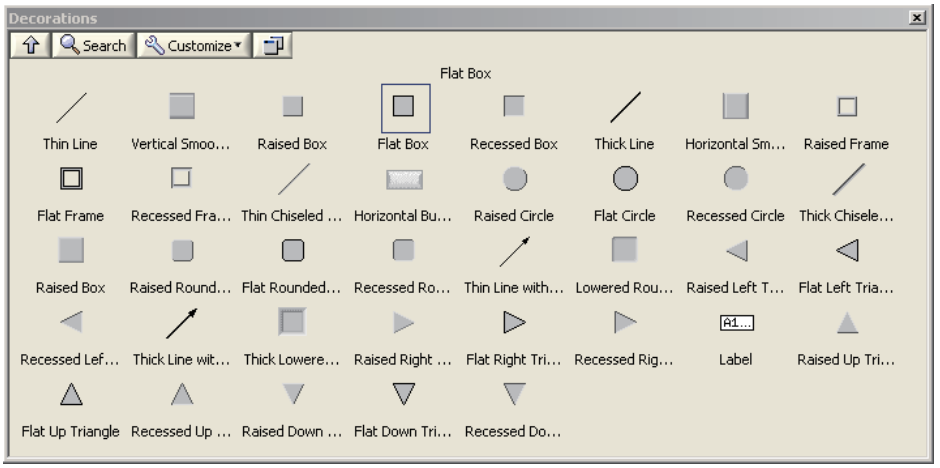
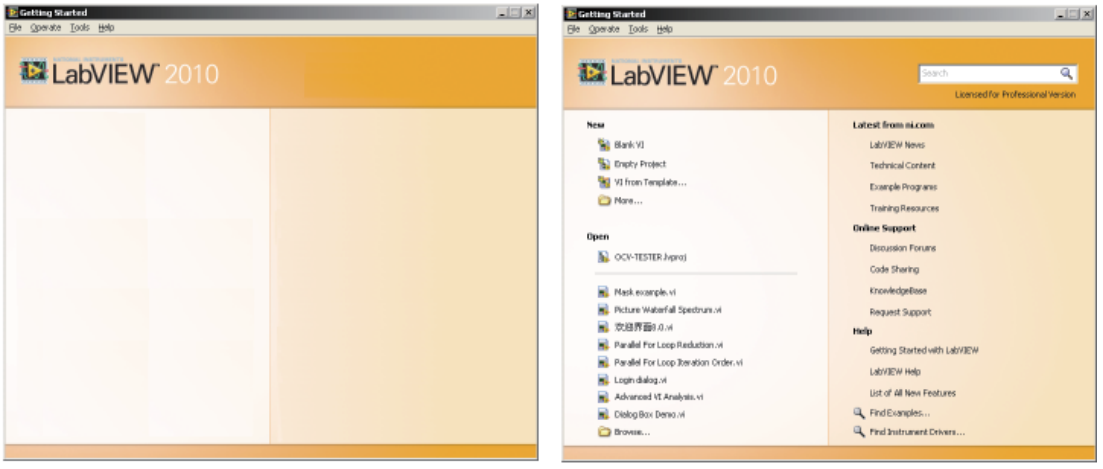


图 2-10　LabVIEW 提供的基本装饰控件

以 LabVIEW 的开始对话框为例（图 2-11），可以按照以下步骤创建其前面板：



（a）　　（b）

图 2-11　使用图片装饰用户界面

（1）使用界面绘图工具（Photoshop、 Visio、PowerPoint 等软件）绘制用户界面背景。

（2）将绘制好的图片复制、粘贴到 VI前面板中，如图 2-11（a）所示。

（3）选择各种控件。必要时使用工具栏上的“纵向排序”（Reorder）工具（图 2-12）对控件和图片的层叠顺序进行排列，确保各控件位于背景图片之上。

（4）将控件设置为透明，使控件和背景图片融为一体，如图 2-11（b）所示。创建此类界面时，一般选用传统风格的控件，因为传统风格的控件可以完全被设置为透明。这一点经常被作为构建独特风格界面的有效手段，例如可以将图片作为背景，将传统风格的控件设置为透明，放置在图片之上，则用户看到的是背景图片，实际上操作的却是控件。

大多数个性化、专业的 VI 前面板都可以使用这种方法创建。甚至可以结合这种方法创建类似图 2-13 所示的 iPhone 界面风格的 VI（本书第 2 章的源码）。

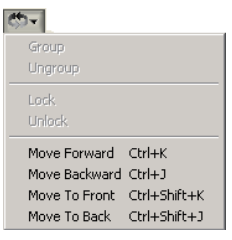


图 2-12　控件纵向排序工具

图 2-13　使用 LabVIEW 创建的 iPhone 风格的 VI

2.7　本章小结

创建用户界面是虚拟仪器项目设计的第一步。一般来说，可以通过以下步骤搭建用户界面：

（1）根据信息的收集和呈现方式，决定是使用对话框、分页、分栏还是使用子窗口形式的应用程序界面整体框架。必要时使用背景图片对主界面整体进行装饰。

（2）根据数据类型和外观选择需要的控件。

（3）根据逻辑调整控件在前面板上的布局。

（4）对控件的文本、色彩按照一致性等原则进行调整。

（5）对界面局部进行装饰，使信息的呈现一目了然。

用户界面设计没有统一标准，对界面的任何优化都应以提高交互效率、实现用户任务为目的。设计时可以围绕这个核心，充分发挥想象，创建美观、专业的界面。