第 1 章　LabVIEW 简介

LabVIEW（Laboratory Virtual Instrument Engineering Workbench）是美国国家仪器公司（National Instruments，NI）开发的应用于工业测试测量的集成开发环境，也是工业上广泛使用的一种功能强大的图形化系统设计编程软件。本章将概括性地介绍 LabVIEW 的基本功能、属性设置、新特性及 LabVIEW 的编程风格。

- ► LabVIEW 概述
- ► 软件环境
- ► VI 属性设置
- ► 其他工具
- ► LabVIEW 2011 新特性

1.1　LabVIEW 概述

本书中涉及的 LabVIEW 技术是笔者使用 LabVIEW 以来的一些切身感受，并不针对固定的某个 LabVIEW 开发版本，为体现出 LabVIEW 的更多新特性，并且针对读者的使用习惯，本书截图实例均来自于 LabVIEW 2009 中文版。图 1-1 所示为 LabVIEW2009 的启动界面。

图 1-1　LabVIEW 2009 启动界面

1.1.1　虚拟仪器技术

虚拟仪器技术首先由 NI（National Instruments）公司提出，它是以计算机软、硬件技术为核心，以自动控制技术、传感器技术、现代信号处理技术、现代网络技术、数值分析技术为

电子工业出版社版权所有 盗版必究

支撑，以各专业学科为应用背景的现代测试技术。它利用高性能的模块化集成概念和方法，结合软件设计平台高效、简便的程序编译功能，依据用户各类特殊需求创建出人机对话界面，实现并取代各类特殊、昂贵的测试仪器的功能，目前已成为测试理论和应用实验研究的重要支撑。

20 世纪 70 年代，因为个人电脑技术的出现，人们开始考虑用电脑来处理传统仪器测试的数据，同时 GPIB 技术也发展起来，促进了 IEEE 488.2 标准的诞生；20 世纪 80 年代，随着计算机技术进一步发展，计算机主板上有了多个扩展槽，并出现了插在计算机里的数据采集卡，这样的系统已经可以进行一些简单的数据采集工作，将采集到的数据直接由计算机软件进行处理，这就是虚拟仪器技术的雏形；20 世纪 90 年代，计算机总线速度进一步提高，PCI 总线的数据传输速率达到了 132Mbps，1996 年底，NI 公司在 PCI 数据总线的基础上提出了第一代 PXI（PCI eXtensions for Instrumentation）系统的技术规范。

到 21 世纪初，全球已有超过 25000 用户在使用虚拟仪器技术，其中不乏国际知名的大公司，像 Nokia、Siemens、Tektronix 等。在世界财富 500 强中的制造业厂商，有 95%都采用了虚拟仪器技术。据专家预测，到 2015 年我国将有 70%的仪器为虚拟仪器。虚拟仪器将在航空、航天、通信、医疗、电力、石油、铁路等行业普及应用。表 1-1 列出了虚拟仪器和传统仪器各自的特点。

表 1-1　虚拟仪器与传统仪器的比较

虚 拟 仪 器	传 统 仪 器
用户自定义	生产厂家定义
开发和维护费用低	开发和维护费用高
技术更新周期短（0.5～1 年）	技术更新周期长（5～10 年）
软件是关键	硬件是关键
价格低廉	价格昂贵
开放灵活、与计算机同步，可重复用和重配置	固定
可利用网络与各远程仪器设备进行通信	只可连接有限设备
自动、智能化、远距离传输	功能单一，操作不便

虚拟仪器最直观的特点就是能够提供和传统仪器一样的图形化操作，为了更形象地体现出虚拟仪器的优势，美国国家仪器公司最近又赋予了虚拟仪器技术一个新的称谓——图形化系统设计（Graphical System Design，GSD）。

1.1.2　LabVIEW 开发系统

图 1-2 所示为 LabVIEW 2009 的启动窗口，窗口的左边栏用来创建新的 vi 或者打开最近打开的 vi 程序或项目。右边栏列出了用户需求的相关链接，包括 NI 公司的最新消息、在线支持、帮助以及范例查找等。

自 1986 年 NI 公司正式发布 LabVIEW 1.0 以来的 20 多年时间里，LabVIEW 不断完善，融合最前沿的技术，已经从最初简单的图形编程、支持单一平台的开发软件，发展到现在以 LabVIEW 为核心，支持多核处理器、FPGA、无线传感器等最新技术并能运行于主流平台的工业软件开发环境。如今的 LabVIEW 通过对计算模型的无缝整合，逐渐解决了在较大规模工程应用中因为工程师之间“语言不通”而造成的瓶颈。

LabVIEW 以其完善的架构，将工业通用的 GPIB，PXI，VXI 等总线的仪器整合到一起，为用户提供了统一的开发标准，使用户在开发过程中不再需要考虑具体的硬件平台。NI 公司

电子工业出版社版权所有 盗版必究

还提供了业界最大的仪器驱动库，同时支持 TCP/IP 协议和 ActiveX、SQL 等技术，通过网络实现实时数据共享，使整个测试测量不再局限于特定的时间和地点。

图 1-2　LabVIEW 2009 启动窗口

在以 LabVIEW 为核心的软件架构上，NI 公司同时提供了机器视觉与运动、信号处理、嵌入式开发、FPGA、模拟仿真等开发包和全面而强大的硬件，为过程控制和工业自动化提供了简单易用的解决方案。基于“软件就是仪器”的理念，LabVIEW 为科研提供了强大的支持。随着软件和硬件功能的不断完善，NI 公司提供了整套的产品研发解决方案，极大地缩短了产品原型研发时间。LabVIEW 不仅仅只是一个软件平台，学习 LabVIEW 软件开发技术的同时，也需要开发人员不断熟悉测控行业的相关技术。以 LabVIEW 为核心，NI 公司针对不同的应用提供了各种软件包，构成了强大的软件开发平台，表 1-2 所示为 NI 提供的主要软件包及其用途介绍。

表 1-2　LabVIEW 工具包及其用途

LabVIEW 工具包	用　　途
Control Design Toolkit	控制设计工具包
Database Connectivity Toolset	数据库连接工具
Digital Filter Design Toolkit	数字滤波器设计工具包
DSC Module	数据记录和监控模块
DSP Design Toolkit	DSP 设计工具包
ECU Measurement and Calibration Toolkit	ECU 测量和校准工具包
Embedded Development Module	嵌入式开发模块
Modulation Toolkit	调制解调工具包
PDA Module	PDA 模块
PID Control Toolkit	PID 控制工具包

电子工业出版社版权所有

（续表）

LabVIEW 工具包	用　　途
Real-time Module	实时模块
Report Generation Toolkit	报表生成工具包
Simulation Module	仿真模块
Sound and Vibration Toolkit	声音和震动分析工具包
Spectral Measurements Toolkit	频谱量测工具包
Switch Executive Toolkit	开关管理和应用工具包
System Identification Module	系统认证模组
Touch Panel Module	触控面板模块

1.1.3　LabVIEW 帮助选项

LabVIEW 在拥有强大功能的同时，也提供了同样强大的帮助系统。对于开发环境中的每个函数和节点，LabVIEW 都提供了在线帮助，打开 LabVIEW 的“帮助→显示即时帮助”菜单项，当鼠标指向函数和节点时在帮助窗口中将显示出该函数和节点的简短说明，如图 1-3 所示。单击“详细帮助信息”，可以看到更多关于“加[Add]”节点的帮助，从帮助文档中可以了解到当前节点所有的功能及用法。

图 1-3　LabVIEW 的即时帮助窗口

LabVIEW 也提供了大量的实例，可以让初学者尽快了解整个系统开发。在不断添加 LabVIEW 工具包的同时，这些实例也会不断增加。单击启动窗口右下角的查找范例或者帮助菜单下的查找范例便可打开范例查找器，其中有大量的实例可作为参考。如果计算机连接 Internet，还可以通过 NI 公司的网站获取更多实例。

在学习 LabVIEW 的过程中，很多人往往忽视了 LabVIEW 帮助功能，其实这些帮助文档是非常重要的，从中可以获取到很多有用的信息，希望读者能够重视。

1.1.4　LabVIEW 选板

LabVIEW 程序通常由前面板和背面板（也称作程序框图）组成，前面板为程序界面放置控制控件和显示控件，背面板实现 VI 逻辑功能的图形化源代码。

前面板是 VI 的人机界面，它通过各种控制控件和显示控件实现各类数据的输入和显示，模拟了真实仪器的面板，包括数值输入/输出、旋钮、开关、按钮、图表、图形、指示灯等。它将数据以图形化的方式呈现在用户面前，因此用户可以非常方便地在前面板中执行对仪器的操作，并观察到希望得到的结果。

创建 VI 时，通常应先设计前面板，单击 LabVIEW 前面板或右击 LabVIEW 前面板空白处，即可打开控件选板，如图 1-4 所示，它提供了前面板上所有的控制控件和显示控件，控件选板以树形目录的方式排列，右上角的箭头表示该子选板下面还有相应的子选板。

图 1-4　控件选板

控件选板包含表 1-3 所列的子选板。

表 1-3　控件选板的各个子选板

图 标	说　明
	新式：包含数值型、布尔型、字符串、数组、簇、图形、容器等各种输入/显示控件。
	系统：系统控件与 Windows 系统控件相同，在不同的操作系统下，该类控件将根据当前操作系统的界面风格自动更改其颜色和外观。
	经典：与新式包含的控件基本相同，经典选板下的控件适合创建在 256 色和 16 色显示器上显示的 VI。
	Express：包含最常用的一些控件，大部分控件和普通控件一样，只有 Express 表格和 Express XY 图控件会自动在程序框图中产生代码。
	控制设计与仿真：安装控制设计与仿真模块后，选板下会放置各种与控制设计和仿真相关的控件。
	.NET 与.Active：该选板下可选择.NET 和.Active 容器以及常用的.NET 和.Active 控件。
	信号处理：该选板下放置了各种与信号处理相关控件。
	附加工具包：用于定位 LabVIEW 中安装的其他模块或工具包。
	用户控件：用户库选板用于添加 VI 至函数选板。
	选择控件：打开选择 VI 对话框，将保存的 VI 作为子 VI 放置到前面板。

设计完前面板，然后设计背面板来执行在前面板上创建的输入/输出任务。背面板是程序员编写 LabVIEW 程序的窗口界面。LabVIEW 是一种图形化的编程语言，相对于文本编程语言，程序员不用一行一行地将代码“敲入”计算机，只需要将图标“拖入”程序框图，再用逻辑连线将各个图标连接在一起，便可以很方便地完成程序编写，并实现特定的功能。

单击背面板或在背面板上空白处右击，就可以打开函数选板，如图 1-5 所示。它提供了开发需要的各个功能模块的函数，该选板仅可以在背面板打开。

图 1-5　函数选板

函数选板包含表 1-4 所示的子选板。

表 1-4　函数选板的各个子选板

图标	说　明
	编程：编程 VI 和函数是创建 VI 的基本工具，包括结构、数值、文件 I/O、波形等子选板。
	测量 I/O：测量 I/O VI 和函数可与 NI-DAQ（Legacy）、NI-DAQmx 及其他数据采集设备交互。选板显示了已安装的硬件驱动程序的 VI 和函数。
	仪器 I/O：仪器 I/O VI 和函数可与 GPIB、串行、模块、PXI 及其他类型的仪器进行交互。NI 仪器驱动查找器用于查找并安装仪器驱动程序。
	视觉和运动：安装相应工具包后，该选板下会产生相关的函数，主要是一些图像采集和图像处理的函数。

版权所有

（续表）

图标	说　明
	数学：数学 VI 用于进行多种数学分析，数学算法也可与实际测量任务相结合来实现实际解决方案。该选板下包含初等与特殊函数和 VI、概率与统计 VI、积分与微分 VI 等各种数学工具。
	信号处理：信号处理 VI 用于执行信号生成、数字滤波、数据加窗及频谱分析，包括各种波形信号生成和调理工具。
	数据通信：函数通信 VI 和函数用于在不同的应用程序间交换数据。该选板下包含 DataSocket VI 和函数、队列操作函数、共享变量节点、VI 和函数、同步 VI 和函数、协议 VI 和函数。
	互连接口：互连接口 VI 和函数用于.NET 对象、已启用 ActiveX 的应用程序、输入设备、注册表地址、源代码控制、Web 服务、Windows 注册表项和其他软件。
	控制设计与仿真：需安装相应的控制设计与仿真模块，该选板下放置了与控制设计和仿真相关的各种函数和 VI。
	Express：Express VI 和函数用于创建常规测量，包括输入/输出 Express VI、算术与比较 Express VI、信号操作分析 Express VI、执行过程控制 Express VI 和函数。
	附加工具包：用于定位 LabVIEW 中安装的其他模块或工具包。
	收藏：用于存放常用的函数。用户可将常用的 VI 和函数放在此选板下，以方便查找。
	用户库：用于添加 VI 至函数选板。默认情况下用户库不包含任何对象。
	选择 VI：打开选择 VI 对话框，将保存的 VI 作为子 VI 放置到程序中。

LabVIEW 还有另外一个突出的优势，由于 LabVIEW 的开发环境是基于数据流的，程序中一个模块运行与否取决于数据是否到达该模块，即模块所有的接口都得到数据时模块才会运行，同一个 VI 中两个并行模块的运行是相互独立的。如图 1-6 所示，模块 1 与模块 2 在同一个 VI 程序中，但它们的运行是相互独立的，互不影响。相比文本编程语言的自上而下的顺序结构，这种基于数据流的 LabVIEW 程序本身是一种并行的编程结构。

图 1-7 所示为 LabVIEW 中的工具选板。选择 LabVIEW 菜单栏的“查看→工具”便可打开，在前面板或背面板按住 Shift 键的同时并右击也可以打开工具选板。在开发或调试程序时，用户可以灵活使用各种工具，如用连线工具来连接背面板的各个函数节点。若打开自动选择工具，那么 LabVIEW 将依据当前鼠标的位置在各种工具之间切换。表 1-5 列举了工具选板中各工具的详细功能。

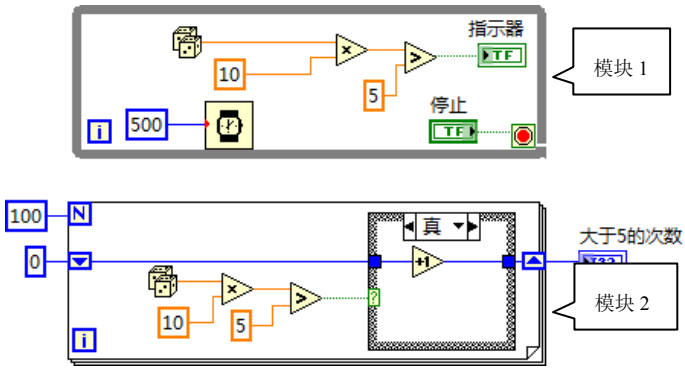


图 1-6　同一程序中的并行模块

图 1-7　工具选板

表 1-5　工具选板的名称和功能

图标	名称和功能
	自动选择工具：根据鼠标相对于控件的位置自动选择合适的工具。
	操作值工具：用于操作前面板对象数据，或选择对象内的文本或数据。
	定位/调整大小/选择工具：用于选择对象、移动对象或缩放对象大小。
	编辑文本工具：用于在对象中输入文本或在窗口创建标注。
	进行连线工具：用于在框图程序中节点端口之间连线，或定义子程序的端口。
	对象快捷菜单工具：用于弹出右键菜单，与单击鼠标右键作用相同。
	滚动窗口工具：同时移动窗口内所有的对象。
	设置/清除断点工具：用于在框图程序内设置或清除断点。
	探针数据工具：用于在框图程序内的数据连线上设置数据探针。
	获取颜色工具：获取对象某点的颜色。
	设置颜色工具：利用在颜色选择对话框中选择的颜色，或用由颜色复制工具获得的颜色给对象上色。

1.2　软 件 环 境

作为代表性的程序开发环境，LabVIEW 提供了许多设定编程软件的配置选项，主要包括系统运行环境、编程窗口环境、VI 服务器和 Web 服务器 4 大部分。用户可以从“工具→选项”菜单中调出软件环境设置对话框，如图 1-8 所示。LabVIEW 程序开发人员可以按照自己的习惯或者需求设定这些功能属性。

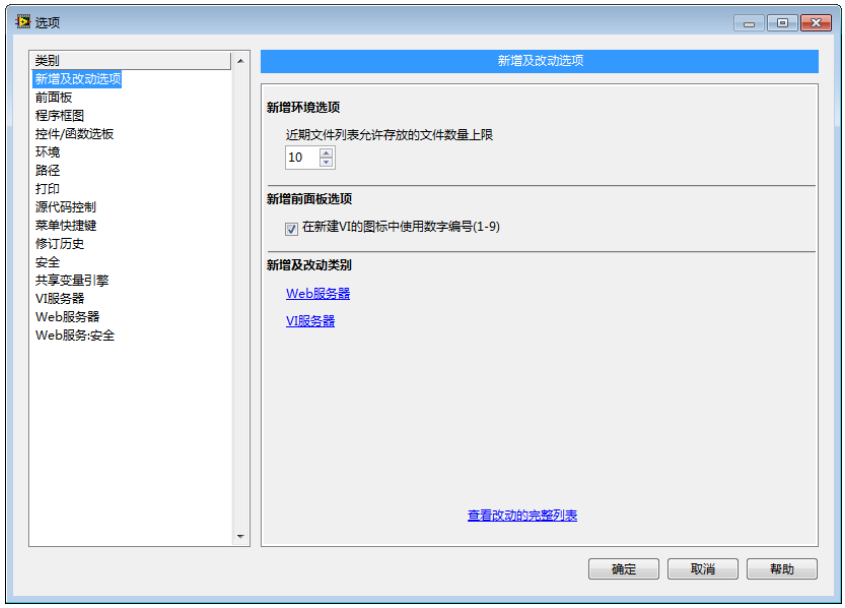


图 1-8　软件环境设置界面

电子工业出版社版权所有 盗版必究

1.2.1 系统环境

LabVIEW 系统环境配置，主要是指那些与操作系统相关的，关系到编程环境本身安装、运行等方面的配置选项。这里主要介绍 Windows 操作系统下的 LabVIEW 系统环境设置。

1．使用环境

LabVIEW 具备 Windows 平台下应用软件的共性，如快捷键、窗口风格、操作方式等。当然它也有自己独特的配置选项，其中很多是和它独特的应用方式相关的。LabVIEW 的熟练使用者，可以通过设置编程环境来使工作更便捷。

① 自动关闭 VISA 会话句柄（Automatically close VISA sessions）：当顶层 VI 空闲时，自动关闭应用程序打开的 VI 会话句柄。该复选框默认为未勾选。

② 锁定自动选择工具（Lock automatic tool selection）：这是设置控件自动切换的功能快捷键。该复选框默认为勾选，按 Tab 键时进行控件自动切换，否则用 Shift+Tab 组合键进行切换。

③ 每个 VI 允许的最大撤销步数（Maximum undo steps per VI）：默认值为 30，最大可以设置为 99。撤销在程序编写过程中是经常用到的，多的可撤销步数将为编程提供方便。需要注意的是，一旦文件保存，任何保存前的操作都不能撤销。

④ 清除密码缓存（Clear password cache）：对于使用了密码保护的 VI，在查看程序框图时 LabVIEW 会将密码保留在缓存中，在同一次会话中即无须重新输入密码。当清除密码缓存时，LabVIEW 将关闭所有受密码保护 VI 的程序框图，再次查看时，必须重新输入密码。

⑤ 锁定只读 VI（Treat read-only Vis as locked）：主要用于源代码的控件，该选项限制了对只读 VI 的修改。如果未勾选则仍可编辑只读 VI，但是不能保存所做的改动；否则只读 VI 将以锁定形式打开，不能进行任何编辑。该复选框默认为勾选。

⑥ 启动用于恢复的自动保存（Enable automatic saving for recovery）：在 LabVIEW 非正常关闭后，再次开启时，会恢复上次没有保存或损坏的文件，以减小意外造成的损失。在 LabVIEW 中还可以设置自动保存的时间，当然及时手动保存的习惯更值得提倡。

⑦ 字体（Fonts）和颜色（Color）：用户可根据自己的习惯配置应用程序、对话框和系统的字体，并在下面的方框中显示所选字体的预览。用户还可配置前面板、程序框图、强制转换点、滚动条等的显示颜色，单击下面的自定义颜色，在颜色选择器中可定义多达 18 种指定名称的用户颜色。

此选项页还包括：“使用回车键结束文本输入”（End text entry with Enter key）、“启动时忽略启动窗口”（Skip Getting Started window on launch）、“显示所创建常量的标签”（Show created constant labels）、“启动时提示调查内部错误”（ Prompt to investigate internal errors on startup）等配置项。

2．路径选项

该选项用于定义 LabVIEW 资源的路径，并指定搜索目录的顺序，包括临时目录、默认目录、默认数据目录及 VI 搜索路径，一般使用默认值即可。

编写大型软件时，在 LabVIEW 编程中一般通过目录结构管理文件，通过设定 VI 搜索路径（VI Search Path）可使 LabVIEW 在确定的范围内操作 VI，既避免错误关联又提高程序编译速度。

在目录编辑中，LabVIEW 提供了“符号路径”（Symbolic Paths），如表 1-6 所示。需要注

意的是这些符号路径只在设置 VI 搜索路径时有效，而不能在程序编写中运用。

表 1-6　路径环境

符 号 路 径	代 表 意 义
<topvi>	指 LabVIEW 打开的顶层 VI 所在的目录
<foundvi>	指先前搜索到子 VI 的全部目录。搜索过程中，LabVIEW 会将搜索到子 VI 的所有目录添加到该列表中。<foundvi>帮助定位已经移动或重命名的 VI 的目录。如调用一个改变路径的 VI，则必须找到该 VI 所在目录并将路径添加至目录列表
<vilib>	指 LabVIEW 目录下的 vi.lib 目录
<userlib>	指 LabVIEW 目录下的 user.lib 目录
<instrlib>	指 LabVIEW 目录下的 instr.lib 目录
<osdatadir>	指操作系统的默认文件目录

该选项可通过路径列表（Path List）显示路径选择器列表中选中的路径。如果列表中的路径不存在，则路径旁将显示一个警告标志。库目录、临时目录和默认目录只能含有一个路径。VI 搜索路径可以包含多条路径，默认为库目录。在 LabVIEW 搜索缺失 VI 时，用户可以手动将缺失的 VI 替换为选中的 VI。

3．安全（Security）

安全选项用于共享变量的管理，在分布测量中采用了可以多人使用的共享变量，因此，当多台计算机协同工作时，为了安全起见需要设立域，访问时要求使用密码登录，安全设置就是对这些参数进行配置。设置主要包括登录方式、用户名、密码管理、域管理。设置前，程序会自动探测网络中存在的域作为设定的必要信息。根据设定，每次启动 LabVIEW 都会自动登录或提示手动登录到特定的域。

一般小规模的工程是不需要设定域的，也就无所谓登录，可以忽略这个选项。

4．打印（Printing）

设置打印模式，是针对使用的打印机的情况，与打印到文件的情况无关，应根据具体打印机来进行设置。和一般的 Windows 应用程序一样，LabVIEW 提供的打印功能，内容包括前面板、程序框图、说明文字等。通过“文件→打印”进入打印向导，LabVIEW 提供了 4 种打印模式，可以根据需要选择打印的模式。

1.2.2　编程环境

LabVIEW 编程环境配置，主要是指那些与程序设计、编码及调试等过程相关选项的配置。

1．前面板（Front Panel）

控件是前面板的基本组成元素，如果选择控件样式不当，可能会使程序在不同的计算机上有不同的显示。读者会留意到前面板控件选板的前 3 个子选板——“新式”、“系统”、“经典”中的控件几乎都一样，只不过它们的控件样式不同。一般我们选择“新式”或“经典”两种控件，控件的样式不随操作系统改变。“系统”直接调用操作系统的控件，风格与 Windows 应用程序相似，这在编写非 LabVIEW 风格的程序时是最好的选择，但要注意这样的程序运行在不同版本的操作系统下控件的样式是不一样的。

“工具→选项→前面板”选项页包含前面板显示相关选项的设置，用户可以用来配置控件的默认属性，主要的配置项目包括以下几项内容。

① 连线板接线端默认为必需（Connector pane terminals default to Required）：设置任何连

电子工业出版社版权所有 盗版必究

接至 VI 连线板的接线端为“必需”，而不是“推荐”。适用于通过连线工具创建的连接，以及通过创建子 VI 创建的子 VI 连接。该复选框默认为未勾选。

② 使用操作系统小数点（Use localized decimal point）：勾选该复选框，可使用操作系统的小数点符号；否则 LabVIEW 在任何情况下都以“.”作为小数点。

③ 双击打开控件编辑器（Open the control editor with double click）：允许双击控件显示控件编辑器窗口。控件器窗口用于自定义前面板对象的外观。一般使用控件右键菜单“高级→自定义…”选项调出控件编辑器窗口。

④ 新 VI 控件样式（Control style for new VI）：设置在程序框图中通过连线端子创建的控件式样。包括通过已存在控件创建新的控件，新建控件与原控件样式相同；通过 VI 的接线端子创建对应控件，新建控件与 VI 中的对应控件相同。

⑤ 前面板网格（Front Panel Grid）：通过配置可选择是否显示前面板网格，以及网格大小，前面板背景对比度，是否启动前面板网格对齐等。

此选项页还包括“闪烁延迟”（Blink delay）、“显示前面板控件提示框”（Show tip strips on front panel controls）、“默认锁定标签”（Labels snap to preset positions on controls）、“默认标签位置”（Default label position）等配置项。

对单个控件来讲，用户可以通过控件的右键快捷菜单设置控件的属性，更多的则采用属性节点的方式设置控件属性。

2．程序框图（Block Diagram）

此选项页包含程序框图的相关设置，主要是程序框图编写的默认属性。可供选择的程序框图配置项目较多，这里主要介绍以下几个。

① 以图标形式放置前面板接线端（Place front panel terminals as icons）：默认情况下，新建前面板对象的接线端显示为图标。取消勾选该复选框，新前面板对象的接线端显示为数据类型。也可右击已有前面板接线端以图标或数据类型形式显示接线端。

② 启动放置结构的自动扩展功能（Place structures with Auto Grow enabled）；调整程序框图的新建结构大小，以便在结构边框放置或移入控件时有足够空间。该复选框默认为勾选。也可为已有结构启用自动调整大小功能。

③ 立即配置 VI（Configure Express Vis immediately）：默认情况下，在程序框图上放置 Express VI 后立即显示配置对话框。若取消，则用户需双击 Express VI 或右击控件在快捷菜单下选择“属性”才能配置 Express VI。

④ 在循环中自动插入反馈节点（Auto-insert Feedback Node in cycles）：在 For 循环和 While 循环中，将子 VI、函数或一组子 VI 及函数的输出连接至其输入时，LabVIEW 将自动插入一个反馈节点。该复选框默认为勾选。

⑤ 在新 VI 中启动自动错误处理（Enable automatic error handling in new VI）和启用自动错误处理对话框（Enable automatic error handling dialogs）：运行 VI 出现错误时，LabVIEW 将挂起执行，高亮显示出现错误的节点并显示错误对话框。建议禁用此项，规范的程序应有自己的错误处理系统。若启用自动错误处理，程序在发生错误时只会中断执行，却不能提供详细的错误信息，更不能针对错误进行适当的处理。

⑥ 启用自动连线路径选择（Enable automatic wire routing）和启用自动连线（Enable auto wiring）：用户连线时由 LabVIEW 选择连线路径，选择路径时 LabVIEW 可自动减少连线转折，同时将仅可能自动连线至输入控件的右侧和显示控件的左侧。用户还可以禁止自动连线，或

电子工业出版社版权所有 盗版必究

设定自动连线的最小距离与最大距离。

此选项页还包括：“显示接线端提示框”（Show tip strips over terminals）、“默认标签位置”（Labels locked by default）、“放置子 VI 时显示名称”（Show SubVInames when dropped）、“在断线上显示红色 X”（Show red Xs on broken wires）、“在连线交叉处显示点标记”（Show dots at wire junction）等，还可以像前面板那样对程序框图网格大小和背景对比度进行设置，以及对程序框图整理设置并在下次通过 VI 菜单选择“编辑→整理程序框图”时生效。

3．控件/函数选板（Controls/Functions Palettes）

在前面板/背面板空白处右击，会弹出控件/函数选板。用户可以设置选板的显示模式和加载方式。该选项页可以设置的配置项主要有以下几种。

① 正在加载选板（Platte Loading）：设置 LabVIEW 如何加载选板的信息，该信息可用于显示和搜索选板。

可选的加载方式有 3 种。“在后台加载选板”（Load palettes in background）：不使用鼠标或键盘时，在后台加载选板信息，选择该方式可能导致 LabVIEW 运行变慢或启动后无响应，尤其是在通过网络运行 LabVIEW 的情况下；“在需要时加载选板”（Load palettes when needed）：浏览选板时加载选板信息，在浏览选板或单击选板工具栏的搜索按钮时，选择该选项可能会导致 LabVIEW 变慢或无响应；“启动时加载选板”（Load palettes during launch）：启动时加载选板信息，LabVIEW 将在启动后到完成加载所有选板信息前阻止任何操作。

② 格式（Format）：设置选板外观和选板项。

格式选板可设置导航按钮的 3 种显示模式以及选板的 6 种显示方式，具体的显示方式从下拉框中可以看到，读者可以依次地选择显示方式再比较导航按钮和选板有什么不同的变化，并且选择自己合适的显示方式，或者从导航按钮的“查看→查看本选板”对选板的显示方式进行更改。

还可以设置选板项排序（Sort palette items）：根据字母顺序对同一级的项进行排序。该选项仅使用于文本和目录树格式。单击控件或函数选板导航按钮的“查看→按字母排序”也可实现该功能。

从菜单“工具→高级→编辑选板…”中可调出选板编辑对话框，如图 1-9 所示，可以添加、删除或者调整选板中 VI 图标的顺序。在相应的 VI 组或者单个 VI 上右击，可以弹出操作菜单。

除了编辑各个分类中的项目，用户还可以单击导航按钮的查看按钮，在弹出的菜单中修改“更改可见类别”选项，勾选常用的项目，这样可进一步缩小函数菜单所占的面积，如图 1-10 所示。

4．共享变量引擎

此选项页在“时间服务器搜索顺序”（Timer Server Search Order）中列出作为时间服务器的计算机。任何运行时间同步服务的计算机都可作为时间服务器使用，主时间服务器是列表中的第一台计算机。如果主服务器因为某种原因离线，被设定为自动同步的计算机将找到同步服务列表上的第二台计算机。在下次同步时，计算机在查找次服务器之前，将首先查找主服务器。如果没有计算机被设置为主时间服务器，计算机将自己进行同步。

在此选项页中，用户可以查找、添加或删除作为时间服务器的计算机；还可以设置“休眠时间（秒）”（Sleep time /seconds）选项，即时间服务器计算机在两个同步的间隔等待的秒数。

电子工业出版社版权所有 盗版必究

图 1-9　控件/函数选板编辑

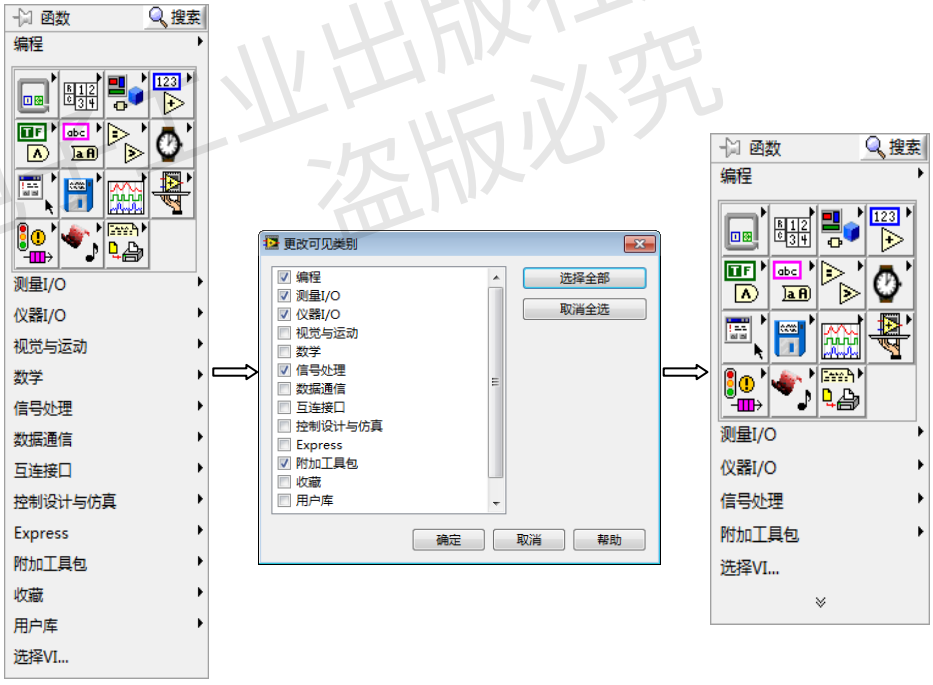


图 1-10　选择“更改可见类别”选项

5．菜单快捷方式（Menu Shortcuts）

该页用于设置 VI 菜单项的键盘快捷方式。VI 窗口之外的其他窗口（如“项目浏览器窗口”或“图标编辑器”对话框）处于活动状态时，键盘快捷方式无效。Ctrl+*快捷键基本都已经定义，一般自定义都是 Ctrl+Shift+* 快捷键。

电子工业出版社版权所有 盗版必究

菜单项（Menu Items and Shortcuts）列表显示了菜单项对应的快捷键。选择某个菜单项后，用户可以在快捷方式/组合键（Shortcut /Press key combination）设置该菜单项的快捷方式。使用 F1～F12 功能键时，无须使用 Ctrl 修饰键。但在按其他键时，LabVIEW 将自动添加 Ctrl 修饰键；如果选择 LabVIEW 的保留组合键，将出现警告，同时显示按钮被禁用；如果选择了另一个菜单项的快捷组合键，将出现一个冲突警告并显示先前指定了该快捷方式的菜单项。

6．修订历史（Revision History）

该页用于设置修订历史的添加规则，与下文中介绍的 VI 属性不同的是，这里主要是选择通用的添加注释的方法及显示的格式。

1.2.3 VI 服务器

通过设置 VI 服务器（VI Server），用户可以将本机上的 VI 发布，这样其他用户可以通过网络访问该 VI，权限为“只读”，即只能运行程序，而不能修改。VI 服务器是 C/S 模式的，一般在工作组中设立一个 VI 服务器，其他计算机可以访问获取信息。

1．协议（Protocols）与可访问的服务器资源（Accessible Server Resources）

这两页用于配置 VI 服务器。默认的 VI 服务器设置为启用 ActiveX，禁用 TCP/IP。

首先应启用 VI 服务器对 TCP/IP 的支持，并且每个应用程序必须拥有唯一的 VI 服务器端口号，默认状态下，端口号为 3363。如果要在机器上运行多个应用程序实例，且每个实例各自的 VI 服务器都处于运行状态，则需重新分配端口号。

然后设置 VI 服务器 TCP 实例的服务名称。服务名称默认为“主应用程序实例/VI 服务器”。

最后通过“可访问的服务器资源”（Accessible Server Resources）选项设置远程应用程序的操作权限。选项包括：“VI 调用”（VI calls）、“VI 属性和方法”（VI properties and methods）、“应用程序方法和属性”（Application methods and properties）、“控件方法和属性”（Control methods and properties）。

2．机器访问选项（Machine Access Options Page）

该选项页用来设置可以访问 VI 服务器的机器终端列表。“机器访问列表”（Machine access list）列出了 VI 服务器允许和拒绝访问的机器的 TCP/IP 地址。用户可以根据需要添加和删除计算机，也可以设置“机器名称/地址”（Machine name/address）指示的计算机为“允许访问”（Allow Access）或“拒绝访问”（Deny Access）。

3．导出 VI 选项页（Exported VIs Options Page）

在此配置页中的“导出 VI 列表”（Exported VI）列出了 VI 服务器可导出的 VI，用户可以添加或移除 VI，并可以设置某个 VI 为“允许访问”（Allow Access）或“拒绝访问”（Deny Access）。

4．用户访问选项页（User Access Options Page）

该页用于控制用户通过 VI 服务器访问 VI 的权限。对项目而言，通过选项对话框即可显示该页，用于控制用户对主要应用程序实例的访问权限。如果需控制用户对终端的访问权限，可从该终端的属性对话框显示该页。

“用户和组访问列表”（User and group access list）列出了可以访问 VI 服务器的用户和组。如果列表中未指明用户或组，则所有在“VI 服务器：机器访问”页具有访问权限的用户和组都可以访问 VI 服务器，可以添加或删除用户和组到列表中，也可以将某个用户和组设置为“允许访问”（Allow Access）或“拒绝访问”（Deny Access）。

电子工业出版社版权所有 盗版必究

1.2.4 Web 服务器

与 VI 服务器不同，Web 服务器（Web Server）主要针对非工作组成员。它允许用户通过浏览器访问和使用 VI，是将程序提供给用户测试的好方式。在使用 Web 服务器时，首先要启动 Web 服务器，默认是未勾选。

1．基本设置（Basic Settings）

在启用 Web 服务器之后便可对 Web 服务器进行配置，主要包括以下几个选项。“根目录”（Root Directory）：表示 Web 服务器 HTML 文件的存放目录，即 VI 发布流程中“保存新网页”对话框中指定的目录，默认路径是 labview2009\www。“HTTP 端口”（HTTP Port）：表示 Web 服务器所使用的 HTTP 端口，默认值为 80。如果使用除 80 以外的其他端口，如 8000，必须在指向服务器的 URL 中指定端口，例如 http://hostname:8000/index.htm。使用高级设置还可对 Web 服务器进行更加详细的配置。

“使用记录文件”（Use Log File）和“记录文件路径”（Log File Path）：记录文件保存的默认路径是 LabVIEW 2009\resource\webserver\logs\access.log。

2．可见 VI 选项（Visible VIs Options Page）

该选项页用于配置并编辑 Web 上可见的 VI。“可见 VI 列表”（Visible VIs）中列出通过 Web 服务器可见的 VI。若允许访问，项的左侧会出现绿色勾选标志；若拒绝访问，则会出现红色的“×”。如果项名称旁边没有绿色的勾选标志或红色的“×”，则该项中存在语法错误。用户可以通过 VI 路径添加 VI 至“可见 VI 列表”，也可以设置列表中的 VI 为“允许访问”（Allow Access）或“拒绝访问”（Deny Access）。

“控制时间限制”（Control time limit）选项用于在多个客户端等待控制 VI 时，对远程客户端控制某个 VI 设定时间限制（以秒为单位），默认为 300s。直到第二个客户端申请控制该 VI 时，LabVIEW 才开始监控该 VI 的控制时间限制。如果没有第二个客户端请求控制 VI，则第一个客户端永远不会失去对 VI 的控制。

3．浏览器访问选项（Browser Access Options Page）

该选项页用来管理可访问 Web 服务器的浏览器地址。“浏览器访问列表”（Browser Access List）列出了允许访问 Web 服务器的浏览器地址。如果允许查看和控制前面板，则项的左边会出现两个绿色的勾选标志；如果只允许查看前面板，则会出现一个绿色勾选标志；如果拒绝访问，则会出现一个红色的“×”。如果项名称旁边没有绿色的勾选标志或红色的“×”，则该项中存在语法错误。用户可以通过 IP 地址添加浏览器至“浏览器访问列表”，也可以设置列表中的浏览器的权限。权限分为 3 种：“允许查看和控制”（Allow Viewing and Controlling）、“允许查看”（Allow Viewing）和“拒绝访问”（Deny Access）。

4．VI 发布

选择菜单栏的“工具→Web 发布工具”选项，可显示如图 1-11 所示的 Web 发布工具对话框，它主要用于创建 HTML 文件和嵌入式 VI 前面板图像。

首先选择要发布的 VI。可以是当前已经载入内存的任何 VI，如果是一组程序则选择它们的主程序。然后定义发布的模式。模式主要包括以下 3 种。

① 嵌入（Embedded）：嵌入 VI 的前面板，客户端可通过浏览器远程查看和控制前面板。只有在服务器计算机内存中的 VI，客户端才能查看和控制前面板。远程前面板连接仅支持嵌入标准 VI，不可嵌入多态 VI。

电子工业出版社版权所有

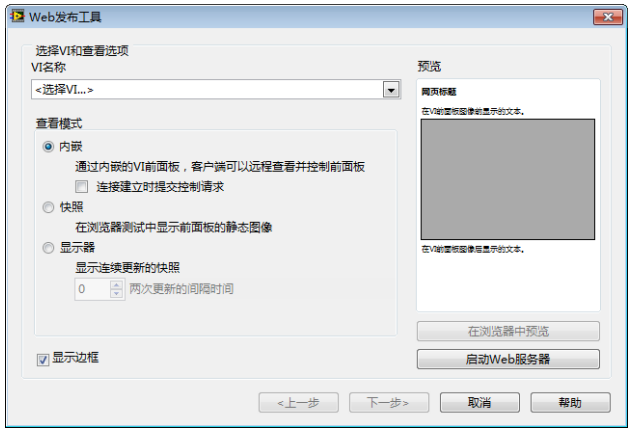


图 1-11　VI 发布工具界面

② 快照（Snapshot）：在浏览器中显示前面板的静态图像，不能在快照中通过浏览器与 VI 控件进行交互。

③ 显示器（Monitor）：显示连续更新的动画快照，不能在快照中通过浏览器与 VI 控件进行交互。所有浏览器均支持显示器选项。

设置完毕后，可以通过“预览”（Preview）或在“浏览器中预览”（Preview in Browser）观看发布效果，确认后单击“启动 Web 服务器”（Start Web Server）。

接着进入“选择 HTML 输出”（Select HTML Output）对话框，设定 HTML 文件的样式，包括“文档标题”（Document title）、“页眉”（Header）、“页脚”（Footer）等。

而后进入“保存新网页”（Save the New Web Page）对话框，设置网页的保存信息。该页主要包括网页的“保存网页的本地目录”（Local Directory to save the Web page）、“文件名”（Filename）及 URL 等项。需要注意的是，Web 发布工具将在打开 VI 的应用程序实例中创建一个 HTML 文件。如果关闭 VI 后再从另一个应用程序实例中打开它，则客户端将无法查看该 VI。

最后单击“保存到磁盘”按钮完成发布，LabVIEW 会自动弹出对话框让用户测试连接。为了正常使用发布的 VI，浏览器不能启用弹出窗口拦截功能，否则可能显示“连接断开”。

1.3　VI 属性设置

VI 属性设置是程序编写的一部分，功能主要是帮助用户管理 VI 程序及控制 VI 运行时的状态和显示方式。通过设置 VI 属性达到某些运行效果也是 LabVIEW 的编程技巧之一。从菜单项“文件→VI 属性”可以调出 VI 属性设置对话框，如图 1-12 所示。

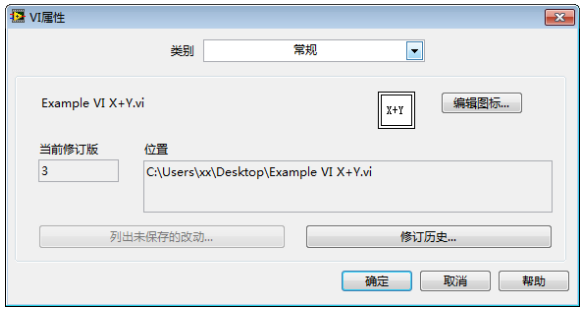


图 1-12　VI 属性设定对话框

1.3.1 基本属性

通过设置基本属性，用户可以更有效地管理 VI 程序。主要配置项有：“名称和图标”（Edit Icon）是 VI 的存储名称和在程序中的显示图标；“当前修订版本”（Current revision）即 VI 的保存次数，可以判定 VI 是否经过修改；“位置”（Location）是 VI 在硬盘上的绝对存储路径；“列出未保存改动”（List Unsaved Changes）列表框显示正在编辑的 VI 所有未保存的修改，以及对改动的简单说明，描述只精确到修改类型；“内存使用”（Memory Usage）显示 VI 在内存和硬盘上占据的空间；“说明信息”（Documentation）是对 VI 的简单说明，也可以添加帮助文件，在编程中调用该 VI 时，可以在“即时帮助”中看到说明和帮助文件的超链接；在“修订历史”（Revision History）中，用户可以添加对该 VI 的修订的描述，所有描述将自动添加修改版本、修改日期、修改人并列表显示，还可以选择提醒功能。

1.3.2 窗口属性

窗口属性主要控制 VI 运行时窗口的显示状态。

1．窗口外观（Window Appearance）

自定义程序运行时窗口中需要显示的项目。可以改变程序运行的窗口上显示的文字，默认是程序的名称，这可以让程序更加专业。

LabVIEW 软件本身提供了几种定义好的风格，但最常用的还是用户自定义模式。适当的选择可能会产生特殊的效果，比如选择“加载时显示前面板”选项就可以作出弹出窗口效果。在 VI 属性“类别”选项框中选择“窗口外观”再单击“自定义”选项，便可打开如图 1-13 所示的“自定义窗口外观”对话框。

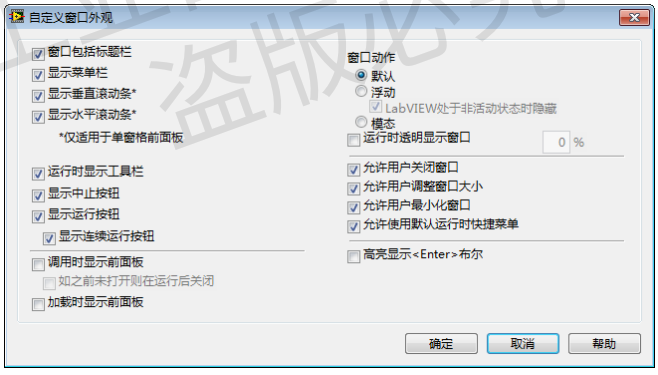


图 1-13　VI 显示属性设定对话框

2．窗口大小（Window Size）

用户可以规定运行时窗口的最小尺寸，单位是像素。

此外还有两个选项：“使用不同分辨率显示器时保持窗口比例”和“调整窗口大小时缩放前面板上的所有对象”。如果忽略显示器的分别，可能会使精心设计的程序界面在用户面前完全走样。

3．窗口运行时位置（Window Run-Time Position）

该选项用于设定程序刚开始运行时窗口的状态，包括窗口位置和窗口大小。

注意：程序运行时的起始位置是它上次保存时的位置，而不是在这里选择的位置；如果两个位置不一致，程序开启后会从保存时的位置“跳”到设定的位置上，所以在程序保存时应当注意窗口的位置和这里设定的位置一致。

电子工业出版社版权所有 盗版必究

4．编辑器选项（Editor Options）

该选项用于设定控件风格和前面板上网格的大小。

1.3.3 执行属性

该选项页可用来设置 VI 的运行属性，主要的配置项包括以下几项。

① 优先级（Priority）：设置程序执行的优先级，是对程序的高级操作，会影响操作系统对程序资源的分配。合理设置程序优先级将提高程序的执行效率。

② 首选执行系统（Preferred Execution System）：设置用户首选的执行系统。LabVIEW 支持多个同步执行系统。在某些平台上，在一个执行系统中运行的 VI 能够在另一个执行系统的 VI 处于运行中途时开始运行。优先级更高的任务（如数据采集循环）可将长操作（如速度较慢的计算）中断。如果有一组 VI 需使用特定的执行系统，可将顶层 VI 设置在该执行系统，这样所有子 VI 也将在该执行系统中运行。父动态分配 VI 与子动态分配 VI 的首选执行系统必须相同。有关执行系统的详细内容请参见本书第 3 章。

③ 重入执行（Reentrant execution）：将 VI 设置为可被一个以上的调用方调用。一般情况下，VI 一次只能被一个调用方调用。如果需两个调用方同时调用同一个 VI，应勾选重入执行复选框。选择此项以后，程序在运行中需要重复调用这个 VI 时，将在内存空间 VI 中开辟新的空间放置数据，而不是直接调用内存中已经有的数据。这样做主要是防止数据流混乱，在多重调用时需要考虑。

④ 打开时运行（Run when opened）：使 VI 进入运行模式并在打开时自动运行。也可使用“打开时运行”属性，通过编程使 VI 在打开时运行。如果需编辑一个已设置为“打开时运行”的 VI，可将其放在新的空 VI 的程序框图上，然后双击该 VI。使用 VI 服务器加载 VI 时，LabVIEW 将忽略该选项，也可通过“运行 VI”的方法运行 VI。

⑤ 调用时挂起（Suspend when called）：子 VI 调用时就被挂起并等待用户的交互。可使用“调用时挂起”属性，通过编程将 VI 挂起。

⑥ 运行时自动处理菜单（Auto handle menus at launch）：使 LabVIEW 在用户打开和运行 VI 时自动处理菜单选项。取消勾选该选项将使运行时菜单栏被禁用，这种情况下可通过“获取所选菜单项”函数进行菜单选择。

⑦ 调用时清除显示控件（Clear indicators when called）：在每次调用含有显示控件（如一个图形）的 VI 时，清除显示控件的内容。

⑧ 启用自动错误处理（Enable automatic error handling）：激活当前 VI 程序框图的自动错误处理。VI 运行时，LabVIEW 会中断执行，高亮显示发生错误的子 VI 或函数并弹出错误对话框。也可用“自动错误处理”属性，通过编程为 VI 启用自动错误处理。

1.3.4 前面板设置

LabVIEW 是针对仪器设计的编程语言，界面和控件都类似于仪器的面板。通过设置标签的字体、控件的颜色式样，再配合使用装饰控件（Decorations），用户可以设计出合适的程序面板，增强程序性能。

为了方便控件摆放，LabVIEW 还提供了各种对齐工具，包括位置调整工具、间距调整工具、尺寸调整工具和图层管理工具。

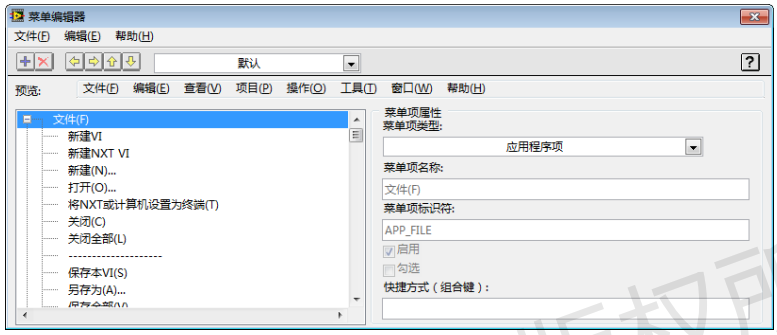
除了对齐工具，LabVIEW 还提供了其他功能帮助控件对齐。当移动控件时，在面板下方

电子工业出版社版权所有 盗版必究

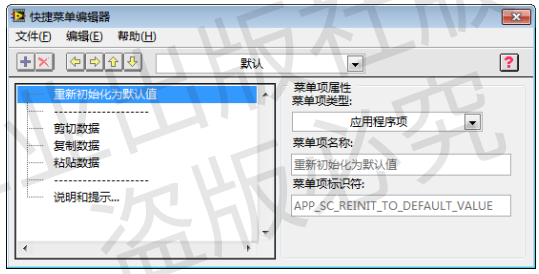
会显示控件的坐标；在控件上右击选择“根据窗格缩放对象”项将有垂直的线条帮助控件定位；当把标签在控件周围移动时，软件会提供自对准功能，标签的外框变成蓝色，只能在几个特定位置放置标签。

1.3.5 运行菜单设置

用以编辑程序运行时显示的菜单。选择“编辑→运行时菜单”可以调出程序运行菜单编辑器，在控件的右键快捷菜单中选择“高级→运行时快捷菜单”项，可以调出控件的运行时快捷菜单编辑器，如图 1-14 所示。两种菜单的编辑方式基本一致。



(a) 程序运行菜单编辑器



(b) 控件运行时快捷菜单编辑器

图 1-14　运行菜单编辑器

程序运行时菜单的“菜单类型”（Menu Type）有 3 种：默认（Default）、最小（Minimal）和自定义（Custom）；控件自定义菜单有两种：默认（Default）和自定义（Custom）。其中，默认菜单和最小菜单是系统定义好的，不可修改。

用户可以设定自定义菜单，在菜单中可以添加的“项类型”（Item Type）有 3 种：“用户项”（User Item）、“分隔符”（Separator）或“应用程序项”（Application Item）。其中，“用户项”完全由用户定义，而“应用程序项”是调用系统默认菜单的一部分，调用后不能修改。

每个“用户项”应包含以下 3 个部分。

① 项名称（Item Name）：在菜单上显示的字符串。

② 项标识符（Item Tag）：每个菜单项必须有一个唯一的标识符，程序框图使用该标识符字符串唯一标识菜单项。标识符是区分大小写的，LabVIEW 忽略标识符前后的空白字符。如果输入的标识符无效或不唯一，LabVIEW 将红色高亮显示该标记。

③ 快捷键（Shortcut）：对于程序菜单项，可以设置快捷键。LabVIEW 中所有快捷菜单的应用程序项标识符的前缀均为“APP_SC”。

用户在编辑菜单的过程中可以通过“预览”观察编辑效果。

电子工业出版社版权所有 盗版必究

1.3.6 自定义控件

控件是每个 VI 程序的必要组成部分，控件的属性会影响程序的功能、显示的效果。在较大的程序中控件的统一也是必要的。

自定义控件（Customs Control）是将 LabVIEW 提供的控件进行修改得到的。一般只是修改控件的默认属性，包括数据格式、颜色、默认值等。右击控件选择“高级→自定义…”，便可弹出控件面板，再单击左上角的“扳手”图标，即可对系统控件进行自定义，或者选择“文件→新建(N)…→其他项目→自定义控件”，也可创建自定义控件，保存后就是可以独立调用的、具有独特属性的自定义控件了。

1.4 其 他 工 具

LabVIEW 提供了一些独特的工具，一方面增强了软件功能；另一方面也支持了一些特殊的服务。熟练使用这些工具可以提高编程效率。

1.4.1 数据日志

LabVIEW 提供前面板数据记录功能，即每次 VI 运行时，软件可以将前面板控件的数据记录到日志文件中，通过编程提取日志文件中记录的数据从而重现 VI 上次运行的状态。采用数据日志功能得到的记录数据将是两个簇，第一个簇记录保存时间，第二个簇记录前面板上所有控件的参数。与数据日志相关的配置项有以下几种。

① 结束时记录（Log at Completion）：数据日志功能默认是不启用的，希望启用数据日志功能应勾选“操作→结束时记录”项，在启用后的首次运行时，会弹出对话框提示选择记录文件。

② 默认数据目录（Default Data Directory）：数据保存的默认路径在“工具→选项→路径→默认数据目录”中指定。

③ 修改记录文件绑定（Change Log File Binding）：在“操作→数据记录→修改记录文件绑定”中可以修改记录当前 VI 数据的文件，单击以后会弹出文件选择对话框，输入文件名即可。

④ 清除记录文件绑定（Clear Log File Binding）：在“操作→数据记录→清除记录文件绑定”中可以取消 VI 和数据的文件绑定，取消绑定后数据文件不会被删除，还可以重新绑定。

⑤ 查看数据记录（Data Retrieve）：可以通过 LabVIEW 软件来查看、编辑当前 VI 绑定的数据文件中记录的数据，选择“操作→数据记录→获取”菜单项后界面如图 1-15 所示。上方的工具栏中显示了文件中记录的索引编号，以及当前记录的索引、日期，也可以在这个界面中删除某个记录。

图 1-15 软件获取数据记录

⑥ 编程读取数据记录（Read Log File）：还可以通过编程直接读取保存在数据文件中的前面板参数，如图 1-16 所示。其中，“record#”用来索引记录，“1”为最近一次记录的数据，“2”为第二临近的记录，依此类推；“invalid record#”标明是否索引有效数据。

电子工业出版社版权所有 盗版必究

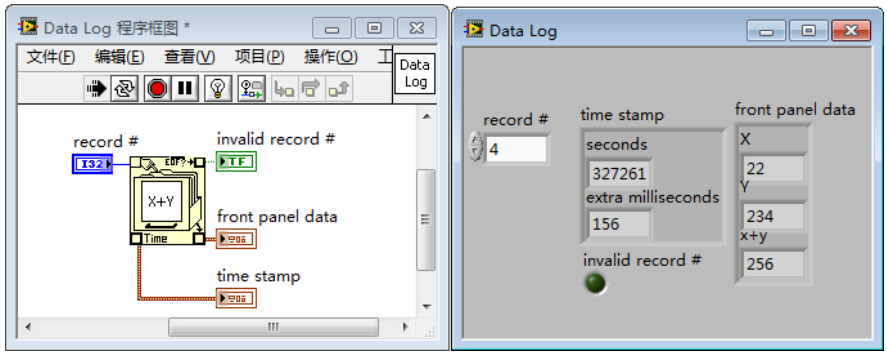


图 1-16　编程读取数据记录

应用数据日志功能可以实现前面板参数的重现，提供更良好的用户界面；也可以通过编写配置文件实现此功能。前者使用方便，而后者更具灵活性。

1.4.2　源代码控制

选择“工具→选项…”，在类别列表中选择源代码控制，或选择“工具→源代码控制→配置源代码控制”也可显示该页。该页用于管理第三方源代码控制软件，并在 LabVIEW 中设置源代码控制选项，主要包括以下几个部分。

① 源代码控制软件名称（Source Control Provider Name）：该项用于指定在 LabVIEW 中使用的源代码控制软件的第三方供应者。LabVIEW 将自动检测已安装的软件作为选项。如果供应者发生改变，则所有 LabVIEW 中打开的项目将会使用新的源代码控制软件。改变供应者后，需要刷新所有已经打开的项目。

在 Windows 操作系统中，LabVIEW 会通过扫描 Windows 注册表，确认已经安装的源代码控制软件，并根据该信息构成源代码控制供应者名称的选项。而在非 Windows 平台上，LabVIEW 将运行一个查询程序以确认是否安装了 Perforce。如果已经安装，“Perforce Command Line”选项将出现在源代码控制供应者名称下拉菜单中。

② 高级（Advanced）：该选项允许设置指定第三方源代码控制软件的属性。在 LabVIEW 中，并非所有的源代码控制软件都支持该选项。如果源代码控制软件不支持，该选项将不会在源代码控制选项页显示。

③ 源代码控制项目（Source Control Project）：该选项用于显示已配置的源代码控制项目的信息，如路径或名称。如果需指定不同的源代码控制项目，可单击修改按钮。并非所有的源代码控制软件都支持源代码控制项目结构。

④ 修改（Change）：选择该项可打开特定源代码控制软件的对话框，设置源代码控制项目的具体信息，包括服务器地址、用户名、客户端详细信息等。仅当源代码控制软件同时支持多个源代码控制项目时，该按钮可用。修改源代码控制项目后，所有打开的 LabVIEW 项目也将发生相应的改动。

⑤ 添加文件时包括层次结构（Include hierarchy when adding files）：将文件添加至源代码控制时，添加内容包括 LabVIEW 文件之间的依赖关系。该复选框默认为选中。如果取消勾选该复选框，向源代码控制添加文件时，“源代码控制操作”对话框将不显示文件的层次结构。

⑥ 签出文件时包括调用方（Include callers when checking out files）：从源代码控制签出文件时，在源代码控制操作对话框中选择包括作为 LabVIEW 文件调用方的文件。该复选框默认

电子工业出版社版权所有 盗版必究

为选中。如果需查看文件列表，必须启用“显示源代码控制操作”对话框以签出文件选项。签出文件时，仅包括进行操作时仍在内存中的调用方。

调用方由文件的源代码控制签出，用户可查看签出的调用方。如果需查看调用方列表，必须启用“显示源代码控制操作”对话框以签出文件选项。

⑦ 显示源代码控制操作对话框以签出文件（Display Source Control Operations dialog box for file checkout）：选择签出源代码控制的文件时，显示“源代码控制操作”对话框。该对话框也可用于签出原本选中的文件之外的其他文件，取消选择不再需要签出的文件及设置签出文件的高级选项。默认状态下，该复选框未勾选。LabVIEW 在签出文件时不作提示。如果同时启用签出文件时包括调用方，签出文件时对话框中将出现调用方文件。

⑧ 编辑时提示签出文件（Prompt to check out files when edited）：编辑尚未签出的 VI 时会出现提示。提示仅在首次编辑 VI 时出现，直至签出文件或关闭 LabVIEW 项目时才再次出现。该复选框默认为选中。该选项对非 VI 文件不适用。

⑨ 添加至 LabVIEW 项目时提示将文件添加至源代码控制（Prompt to add files to source control when adding to LabVIEW project）：向 LabVIEW 项目添加文件时，出现提示以确定是否向源代码控制添加选中的文件及其他相关文件。该复选框默认为选中。如果取消勾选该复选框，LabVIEW 在添加文件时不作提示。向源代码控制添加任何新 VI 或 LabVIEW 项目库时，必须首先保存并命名这些 VI 或库。

⑩ 文件已被签出时提示通知（Notify if files are already checked out）：对已签出的文件进行签出时出现对话框。该对话框将显示由其他用户签出的文件列表。此时可继续签出操作，或返回至“源代码控制操作”对话框取消选择文件，或取消此次签出操作。该复选框默认为选中。如果取消勾选该复选框，LabVIEW 在有多个用户签出文件时不作提示。

1.4.3 LLB 管理

LabVIEW 支持 LLB 文件格式，利用 LLB 打包文件可以像压缩软件一样将许多文件合并成一个文件并压缩文件大小，也可以在不解包的情况下直接运行程序。

LLB 管理器是专门用来进行 LLB 管理的，可以选择菜单中的“工具→LLB 管理器”项调用，如图 1-17 所示，在其中可以复制、重命名和删除文件。LLB 管理器可以创建新的 LLB 和目录，也可将 LLB 转化为目录，或将目录转化为 LLB。右击文件列表中的项，显示快捷菜单，在菜单中可打开、剪切、复制、删除或重命名该项；也可通过该快捷菜单将文件移至 LLB 顶层，或从 LLB 顶层移除文件，一般将主程序放在顶层。

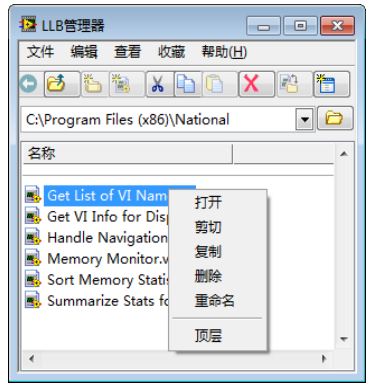


图 1-17 LabVIEW 文件管理器

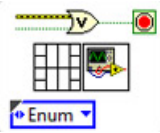
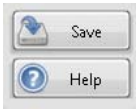
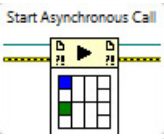
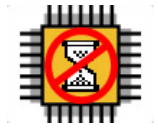
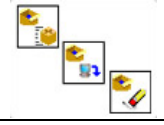
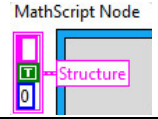
窗口操作和 Windows 资源管理器类似，只是将 LLB 文件当做目录处理。用户可以完成创建新目录（Create New Directory）、创建新 LLB（Create New LLB）、剪切（Cut）、复制（Copy）、粘贴（Paste）、删除（Delete）等操作。针对 LLB 文件的特殊性，管理器还提供了“转化目录和 LLB”（Convert Directories and LLBs）功能。需要注意的是，在 LLB 管理器窗口所做的改动都不可撤销，删除的文件也不会放进“回收站”。

电子工业出版社版权所有 盗版必究

1.5　LabVIEW 2011 新特性

LabVIEW2011 已于 2011 年 8 月初发布，新版本的 LabVIEW 继承之前版本的优势，并在此基础上加入编程和提升实现效率的特性，帮助用户结合不断发展的 PC 和嵌入式技术平台，提高应用效率并实现更多性能，主要的新特性见表 1-7。

表 1-7　LabVIEW2011 新特性

图　标	新　特　性
	更好的稳定性 LabVIEW2011 对系统稳定性及错误报告功能进行了优化，能够快速地解决问题
	根据反馈进行的优化 基于用户反馈新增了 13 项新特性，提高应用开发效率
	UI 改进 通过输入控件和显示控件的新型“银色”选板，可供开发新型应用程序界面
	异步引用调用 使用新型异步 API，轻松调用和运行多个并行子 VI
	更优化的实时部署 通过缓存在内存中先行创建的文件，缩短实时终端的部署时间
	有效缩短 LabVIEW FPGA 的编辑时间 借助经过优化的 FPGA 节点，大幅缩短在 LabVIEW FPGA 中的开发时间
	新型数学和信号处理函数 新增了用于常规工程任务的 20 多项新型数学和信号处理函数来简化开发
	应用程序生成器 API 凭借用于创建可执行程序的内置 API，自动进行代码发布
	更优良的支持 多个重大支持问题的解决带来更好的使用体验
	LabVIEW MathScript RT 模块中的结构 加入对结构体数据结构的支持，可采用更多自定义的.m 文件

电子工业出版社版权所有 盗版必究

1.6 提示与建议

开发人员应仔细阅读下面列出的编程注意事项，以形成良好的编程风格。在开发所有LabVIEW 程序过程中，都可以参考这些注意事项。

1．程序注意事项

① 程序开发过程中所有文件的存储应采用树形结构，主程序放在最上层，子程序放在子目录中。

② 使用库文件时，库文件中不推荐存放大量函数，因为这样会降低函数的存储速度。

③ 如果函数被用做子函数，应创建一个.mnu 文件。排列好前面板，命名每个菜单，并隐藏附属的子 VI。

④ 为每个子程序取一个有意义的名字，但名字中不能包含特殊字符，如\、/、:、~等。

⑤ 每个文件都要使用标准的后缀名（.vi，.ctl），以保证操作系统正确识别文件类型。

⑥ 大写每个程序名的首字母。

⑦ 将实例程序、最上层函数、子程序、控件、全局变量或库文件分开存放在不同的文件夹中以示区别，或给每个文件命名以显示各自用途，如 X Main.vi、X Example.vi、X Globle.vi 和 TypeDef.ctl。

⑧ 为每个程序添加描述，并检查文字帮助窗口显示的内容是否正确。

⑨ 将开发者的名字或开发公司的名字加入到程序的描述文档或程序属性对话框中。

⑩ 在为每个程序分配一个有意义的彩色图标的同时，也分配一个黑白的图标。

⑪ 选择函数接线端模式时，采用 4 × 2 × 2 × 4 模式，为以后的程序开发提供预留空间。相关联的程序，也使用相同的连接方式。

⑫ 避免使用 16 个端口以上的接线端模式。

⑬ 在接线端上为每个端口选择接线端类型（必需、推荐、可选）。

⑭ 使用相应的文件夹保存测试程序，以便在以后的开发中可以重复使用这些程序。

⑮ 在不同的平台下运行程序，确保函数标签正确、程序窗口大小适中、放置在合适的位置上。

⑯ 如果程序将运行在不同操作系统平台上，避免使用基于平台的函数，为不同的平台创建不同的版本。

2．前面板编程注意事项

① 为每个控件分配一个有意义的名字，并将名字的首字母大写。

② 将每个控件的标签背景色设置为透明。

③ 尽量保持所有控件的标签摆放在相同的位置。

④ 整个程序使用统一的字体格式——应用程序字体、系统字体、对话框字体。

⑤ 设置显示文字控件的属性为自动适应文字大小，如果需要可以添加回车。

⑥ 指定文件夹或文件具体位置时使用路径控件而不是字符串控件。

⑦ 子程序中，将控件的默认值和单位放置到控件名后，并用圆括号括起来，如 Time Limit（10s）。

⑧ 为控件加入文字描述。在复制这些控件的同时注意更改控件的文字描述。

⑨ 为用户接触到的窗口控件添加说明，这样有助于用户更好地理解程序。

电子工业出版社版权所有 侵权必究

⑩ 仔细安排控件位置。对于用户可见的窗口，应将最重要的控件放在面板上最醒目的位置；对于子程序，应将输入控件放在面板的左半部，将输出控件放在面板的右半部，并将控件位置与连线板中的端口位置对应。

⑪ 使用程序前面板上的对齐对象、分布对象工具排列控件。

⑫ 不要重叠控件。

⑬ 在同一个程序中避免使用过多的色彩。

⑭ 为程序添加一个停止按钮，不要使用开发界面提供的中止运行按钮，最终生成可执行程序时需隐藏开发界面上的所有调试按钮。

⑮ 在合适的条件下使用下拉列表控件和枚举控件。如果使用布尔控件控制两种状态时，可以考虑使用枚举控件来控制这两种状态，为以后状态添加选项预留空间。

⑯ 对于通用控件，可以使用用户自定义控件进行替代，这种方式对于下拉列表控件和枚举控件尤其有效。

⑰ 保证前面板上的控件风格一样，如都使用经典控件。

3．背面板编程注意事项

① 避免背面板过大。应使背面板占满整个屏幕，且保证滚动条只需要向一个方向滚动就可以浏览整个背面板代码。

② 注释控件、重要的函数、常量、属性节点、本地变量、局域变量和结构体。

③ 如果标签放在控件左边应注意文字右对齐。

④ 程序中使用统一的字体。

⑤ 所有的标签均使用“调整为文本大小”属性，如果可以，应在文字最后一行加上一个回车。

⑥ 遵循从左到右的数据流规则，对所有节点，数据从左边进入，从右边输出。

⑦ 排列好所有的函数、节点和常量。

⑧ 使用自由标签注释较长的连线。

⑨ 不要在子程序或结构下布线。

⑩ 指定文件或文件夹的位置时，使用路径常量替代字符串常量。

⑪ 利用好可重用的、经过测试的子程序。

⑫ 在所有的子程序中使用错误输入/输出簇。

⑬ 确保程序可以处理错误和无效数据。

⑭ 对于 CIN 节点，标识源代码或包含源代码。

⑮ 使用顺序结构时，应采用平铺式顺序结构替代层叠式顺序结构。

⑯ 如果结构（条件结构、层叠式顺序结构、事件结构）存在很多帧，应将最重要的帧凸显出来。

⑰ 优化程序，检查数据备份和精度，程序不存在数据依赖时这样的工作极为重要。

⑱ 保证子程序的图标在背面板可见。

⑲ 使用引用时，不论引用指向何种控件，使用完后都应销毁这些引用。否则，即使程序不再运行，引用也会停留在内存中。

⑳ 将所有的属性节点和方法节点的名字设定为短名字格式，以方便其他开发人员阅读程序。

㉑ 子程序中保证所有的控制控件都摆放在接线端的左半部。

㉒ 编写子程序时，保证所有的控件都不在结构中。

电子工业出版社版权所有
盗版必究

4．编程环境注意事项

① 控件/函数选板显示风格：在 LabVIEW 2009 中默认的显示风格是“类”（Icons and Text），这种风格会占用太多的空间，对编程造成不便；一般选择“类”（Icons）既节省空间又可以享受分类带来的快捷。

② 撤销步骤次数：默认为 30 次，但在实际编程中可能会显得不够，尤其是对新用户而言，根据实际情况可适当增减撤销步骤次数，这样当发现错误时就有更多的机会用 Ctrl+Z 键恢复到先前的状态。

③ 背面板连线端模式：数据类型模式相对于简单的端子表现更加丰富，但是要占据更大的空间；程序框图的重点在于数据处理的流程和结构，端子不用图标可以使程序更加明晰。

④ VI 显示和运行属性：根据 VI 的特性，设置显示和运行属性是编程中必不可少的步骤。

⑤ 运行时菜单和快捷键：通过菜单能将软件功能分类的展现给用户，设置快捷菜单可使软件操作更加快捷。

关于图形化系统设计 LabVIEW 程序设计的一套编程规范，将在本书第 11 章中进一步详细阐述。读者应在学习编程的过程中培养良好的编程规范，以便能在今后的开发工作中编写出效率更高的代码。

习　　题

1-1　编写程序绘制三阶多项式 $y=Ax^3+Bx^2+Cx+D$ 的曲线。使用前面板控件来输入系数 A、B、C 和 D，用 For 循环控制 x 的计算区间，并在 XY Graph 控件中显示 Y–X 曲线。

1-2　完成程序，绘制 $\sin x$ 及积分函数 $y=\int_0^{n\pi}\sin x\mathrm{d}x$ 的曲线。n 的取值应该可以从前面板输入，函数 $y=\int_0^{n\pi}\sin x\mathrm{d}x$ 使用子函数实现，并设计子函数的图标。

1-3　路径设置的作用是什么？体会不同路径设置环境对程序编译的影响。

1-4　在局域网内设置 LabVIEW 工作域，使用共享变量等技术，构建联机工作环境，体会多机协作工作。

1-5　利用“打印”功能保存程序面板、程序框图、程序结构图。

1-6　比较不同式样的控件的区别。编辑程序，比较在不同操作系统环境下程序控件的改变。

1-7　编辑函数选择面板，熟悉操作，将选择板按照自己的习惯编排。

1-8　建立 VI 服务器，配置不同参数，观察效果。

1-9　练习用 Web 服务器发布程序，远程访问程序。

1-10　设置 VI 执行属性和 VI 外观属性，比较不同设置下程序的区别，理解各个参数的意义。

1-11　熟悉 LLB 管理，建立 LLB 文件或将已存在程序封装成 LLB 文件。

电子工业出版社版权所有 盗版必究

第 2 章　LabVIEW 程序设计模式

20 世纪 60 年代的软件危机使人们开始重视软件工程的研究。起初，人们把软件设计的重点放在数据结构和算法的选择上，随着软件系统规模越来越庞大、越来越复杂，整个系统的结构和规格说明也显得越来越重要。面对日益复杂的软件系统，人们开始认识到，要真正实现软件的工业化生产方式，避免软件结构的重复工作，以达到软件产业发展所需要的软件生产率和质量，软件复用是一条现实可行的途径。

- ▶ 面向对象设计模式
- ▶ LabVIEW 程序设计模式
- ▶ 状态机模式
- ▶ 消息队列处理模式
- ▶ 用户界面事件模式
- ▶ 状态机—用户界面事件混合模式
- ▶ 其他模式

面向对象的程序设计模式的提出，开创了面向对象编程中使用模式化方法对软件设计进行复用的思维模式。设计模式是指软件实践中通用的程序架构，其本质是在对很多十分类似的问题进行总结归纳的基础上提炼出的一些具有代表性的软件开发规范。这些成熟的面向对象程序设计模式被广泛应用于采用不同面向对象程序设计语言开发的应用系统中，对软件产业的发展产生了巨大的推动作用。

由于 LabVIEW 采用图形化数据流编程的设计方法，因此传统的面向对象设计模式并不能直接适用于 LabVIEW 的程序设计中，但设计模式的思想对于 LabVIEW 程序设计同样具有重要的参考价值。本章将在介绍面向对象程序设计模式的基础上，分析 LabVIEW 程序设计中设计模式的基本特征，并结合 LabVIEW 的不同应用情况，给出一些 LabVIEW 程序设计中常用的设计模式。

2.1　面向对象设计模式

设计模式（Design Patterns）是一套被反复使用、为多数人所知晓的、经过对分类编目的代码设计经验进行总结而得出的设计规范。使用设计模式是为了保证程序源代码的可复用性、可读性和可维护性。每个设计模式都系统地命名、解释和评价了面向对象系统中一个重要的和重复出现的设计。

研究设计模式的目的是将设计经验以人们能够有效利用的形式记录下来。设计模式使人们可以更加简单方便地复用成功的设计和体系结构，帮助开发人员作出有利于系统复用的选择，避免设计损害系统复用性。此外，通过提供一个显式的类及对象间的作用关系和隐含联系的说明规范，设计模式能够提高已有系统的文档管理和系统维护的有效性。简而言之，设计模式可以帮助设计者更快、更好地完成系统设计。

2.1.1 作用

面向对象设计方法可用来促进良好的设计、教会新手如何设计，以及对设计活动进行标准化。它通常定义了一组用来为设计问题各方面进行建模的记号，以及一组决定在什么样情况下以什么样的方式使用这些记号的规则。另外，它展示了使用诸如对象、继承和多态等基本技术的方法，也展示了如何以算法、行为、状态或者需生成的对象类型来将一个系统参数化。而设计模式使程序开发人员可以更多地描述“为什么”这样设计，而不仅仅是记录下他们的设计结果。设计模式的适用性、效果和实现部分都会帮助指导程序开发人员作出各个必要的设计选择。

在多个程序员协作撰写程序之前，通常需要将程序的分析模型转换为实现模型。由于各个程序员采用的程序分析模型不同，因此很难在总体的面向对象程序设计中进行综合和调用，这无疑需要付出更多的代价去解决这些矛盾，而采用程序设计模式却能够比较容易地解决这个问题。它允许各程序员采用统一的程序设计标准，兼容各种底层执行方式，这在以前的面向对象程序设计过程中却往往被忽略了。

2.1.2 要素

设计模式是对特定问题经过无数次经验总结后提出的，但它并不是一成不变的“定律”。程序员不仅需要明确具有哪些常用的程序设计模式，还必须知道设计模式所解决的是实际应用中的哪种问题，是如何解决的及解决的效果如何等，只有这样才能够在自己的设计中正确、恰当地使用设计模式。这就如“习武”一般，武功套路（相当于设计模式）是习武的门径。新手要一招一式地练习套路，烂熟于心之后，熟能生巧，在实战之中才能见招拆招，运用自如；而“高手”则没有套路，实战之中只有自然反应，因此一个优秀的程序设计员应清楚各种模式的原理和用途，习惯于充分利用模式解决实际的问题。

设计模式通常具有 4 个基本要素，包括设计模式名称、应用问题、解决方案和解决效果。

① 模式名称是一个设计模式助记名，用来描述设计模式的应用问题、解决方案和解决效果。它主要用于帮助思考、交流及编写文档，因此应该注重简明扼要，充分反映模式的功能。

② 应用问题描述了应该在何时使用一个设计模式。它解释了为什么需要设计模式，描述了特定的设计问题，如怎样用对象表示算法等。有时候，应用问题部分会提出使用模式必须满足的一系列先决条件。

③ 解决方案描述了设计模式的组成成分，它们之间的相互关系及各自的职责和协作方式。因为设计模式就像一个模板，可应用于多种不同场合，所以解决方案并不描述一个特定而具体的设计或实现，而是提供设计问题的抽象描述和怎样用一个具有一般意义的元素组合（类或对象组合）来解决特定的问题。

④ 解决效果描述了模式应用的效果及使用模式应权衡的问题。尽管我们描述设计决策时，并不常提到模式的效果，但它们对于评价设计选择和理解使用模式的代价具有重要意义。因为可复用是面向对象设计的要素之一，所以模式效果包括它对系统的灵活性、扩充性或可移植性的影响，显式地列出这些效果对理解和评价这些模式很有帮助。

电子工业出版社版权所有

2.2 LabVIEW 程序设计模式

LabVIEW 是一种针对性比较强的图形化程序设计语言，无论语言针对的主要用户对象和应用领域，还是语言的使用和编程风格，都与传统的文本式程序设计语言有很大的差别。因此，在文本式编程语言中广泛采用的设计模式并不能直接应用于 LabVIEW 程序设计过程中，但设计模式的这种将设计经验以能够有效利用的形式记录下来并加以利用的软件复用思想，在 LabVIEW 的程序设计过程中同样具有重要的价值。

将那些在 LabVIEW 应用程序过程中被反复使用的设计结构、程序框图代码等设计经验进行有效的总结，提炼出 LabVIEW 程序的设计模式对 LabVIEW 语言的应用和发展具有重要的意义。但在研究 LabVIEW 程序设计模式的时候，必须充分认识到 LabVIEW 程序设计的特点，并结合这些特点对 LabVIEW 程序设计模式进行分析和描述。这些特点主要包括 LabVIEW 的面向测试测量应用的程序基本框架和面向数据流的图形化编程方式。

2.2.1 应用程序的基本框架

源于虚拟仪器技术的 LabVIEW 程序设计语言，从被创建开始就是面向测量和控制应用的，并且绝大多数采用 LabVIEW 开发的应用程序都同测控仪器等硬件设备紧密结合。虽然这些设备的类型和规模各不相同，应用领域的差异也很大，但从测量和控制过程的基本步骤来看，绝大多数 LabVIEW 程序的基本框架是有章可循的，具有一定的模式特征。这正是讨论 LabVIEW 程序设计模式的重要基础。

本章以测量过程为例讨论 LabVIEW 程序的基本框架。例如，在多数测量系统的应用程序框架基本可以分为 8 个部分，如图 2-1 所示，各组成部分的功能如下。

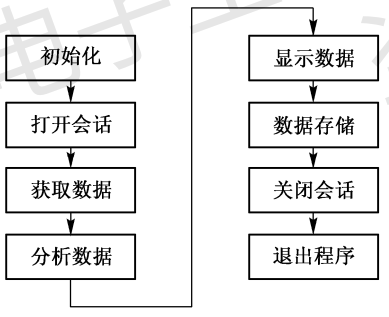


图 2-1 测量系统的 LabVIEW 程序框架

① 初始化。主要用于初始化系统中的软、硬件系统，如将控件中的值清空、进行仪器自检工作、系统复位等。

② 打开会话。主要用于与仪器建立通信，准备获取数据，也可以将寄存器清空，及时释放上次因为异常等原因没有释放的资源。

③ 获取数据。用于得到测量数据，可以通过各种渠道获取，如数据采集硬件设备等。这也是至关重要的一步，数据的可靠性与实时性都直接影响到整个测量的效果。根据实际情况，可以决定是否需要进行信号调理，以完成信号的硬件预处理，获得更加真实的信号。

④ 分析数据。一般而言，直接获取的数据并不都是有效数据，也不能直接表达实际的物理意义和量纲。因此需要后期为其赋予具体的物理意义，如电压、电流等，甚至通过比较复杂的数据处理过程，如借助 DSP、小波分析等高级数学工具，才能提取出真正有价值的数据。

⑤ 显示数据。为了能够将数据简单明了地传递给用户，这一步是必不可少的，在 LabVIEW 中提供了多种显示控件。对波形而言，可以使用波形图，波形图标，XY 图，强度图等。

⑥ 数据存储。测量的数据需要通过各种方式被存储起来，如数据库、存储介质、网络等，便于以后查看和分析，当然也可以打印出相应的测试报表。

⑦ 关闭会话。这个步骤是不可忽略的，在前面的操作中，程序会使用操作系统分配的各种资源，如文件句柄、仪器句柄等。这些资源在使用完后并不会自动释放，而需要及时手动销毁，否则会使应用程序占用更多的系统资源，甚至导致内存泄露。

⑧ 退出程序。在完成了所有的工作后，需要退出应用程序并释放系统资源。

这 8 个部分基本包括了一般测量过程所需要的所有步骤，因此这个 LabVIEW 程序基本框架可以广泛应用于测量系统的设计与开发。如果在此基础之上，结合 LabVIEW 进行深入的抽象分析则可以提取出有价值的设计模式框架。对于控制过程的应用，其基本框架同这个框架非常相似，不同的是，在数据分析之后会增加反馈控制的部分。

2.2.2 图形化数据流编程

LabVIEW 的设计目标是为测试测量行业的工程师提供一种简捷灵活的编程工具，用于开发满足实际工程中遇到的测量控制需要的工程应用系统。因此，与传统的文本编程方式不同，LabVIEW 采用了独特的图形化数据流编程模式，这使程序开发人员可以很快地掌握 LabVIEW 编程技术并实现应用系统的构建。

LabVIEW 数据流编程模式类似于传统的面向过程编程模式，同样通过程序执行控制结构和子程序等组件来构成整个程序的框架，但采用面向过程编程模式实现的代码是以程序命令的执行过程为主线来展开设计的，而 LabVIEW 数据流编程实现的代码是以程序数据的处理过程为主线来展开的。此外，图形化编程模式也使得所实现的代码在执行上具有一定特殊性，与文本的顺序行执行有所不同，LabVIEW 程序框图中节点间的数据流确定了代码的执行次序，这使得互不关联的代码可以简单地建立并行性程序。从图 2-2 中可以看出，与传统的文本编程方式不同，整个源代码完全采用图形构建。代码的执行采用数据流（Data Flow）方式，在运行过程中，可以通过调试工具看到各个节点的数值。

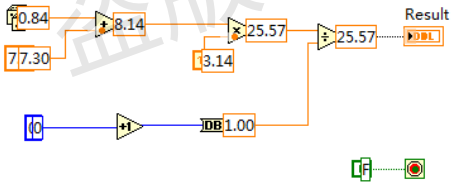


图 2-2 LabVIEW 图形化源代码

2.2.3 设计模式分类

本章介绍的 LabVIEW 程序设计模式是在大量开发实践的基础上，结合国内外 LabVIEW 程序设计经验后研究提出的图形化编程思想。这些设计模式从实际出发，给出了满足应用需求的 LabVIEW 程序代码框架。根据其针对问题的不同，可以分为通用型和专用型两类。通用型的 LabVIEW 程序设计模式是针对一般性测量控制应用程序提出的，而专用型的 LabVIEW 程序设计模式是针对某些特殊的应用或应用中某些特殊功能提出的解决方案。按照这两种类型分类，各种类型分别包含以下设计模式：

① 通用型 LabVIEW 程序设计模式主要包括状态机模式、消息队列处理模式和用户界面事件模式；

② 专用型 LabVIEW 程序设计模式主要包括主从线程模式、生产消费模式、后台服务模式、异常处理模式和代理模式。

下面将详细讨论上述典型的程序设计模式。

电子工业出版社版权所有 盗版必究

2.3 状态机模式

本节将介绍一种使用最广泛的 LabVIEW 程序设计模式——状态机模式。状态机是一种成熟的适用于大部分软件系统开发的设计模型，它使程序开发人员能够很容易地描述出对象或系统在整个生命周期或执行过程中的各种活动情况，有利于代码的开发和维护等工作。

作为一种程序设计语言，很多传统的软件设计方法都可以应用于 LabVIEW 的程序设计过程中，但状态机是最有效的方法之一，这是由 LabVIEW 的特性决定的。绝大多数的测试测量系统在运行时需要从一个状态转换到另一个状态，或者在不同的状态之间互相切换，直至结束。因此状态机模式作为一种典型的类顺序结构方式，被广泛应用于各种自动化测试系统中。

2.3.1 状态机

状态机具有 3 个基本的要素：状态、事件和动作，任何一个状态机的执行都离不开这 3 要素。而状态的选择是保证其他步骤有条不紊进行的前提，通常把程序需要经历的状态称作一个“状态序列”，因此我们就从状态的概念开始讨论状态机的构造。

“状态”是一个抽象的术语，但是程序员往往误用它。在设计可交互式状态序列时，“等待”似乎是一个必不可少的状态。例如常有一个状态需要等待用户“确认”，这个状态决定了下一个状态，这取决于与外部对象的交互。此外，状态序列描述了程序当前的运行情况，各个状态之间是可以相互转换的。所以，选择合理的状态序列可以增加代码的易读性和健壮性。

状态机在控制状态的同时，与各个状态对应的事件也会随之触发。例如，在“确认已经与仪器建立连接”就是一个具体的事件，该外部事件将会被告知状态机，以便控制程序的执行顺序。当然，事件也可以由状态机中的代码在内部产生。

动作是事件的响应。当一个事件发生时，状态机会决定应该执行什么样的动作，这主要取决于目前所处的状态和发生的事件。在本章的例子中，只有当一些特殊的动作发生时，状态机的状态才会改变。这就好比通信控制，除非某个特定的指令到达，通信的状态才会改变，否则会一直处于当前状态。值得注意的是，状态机本身并不会改变状态，当前的状态也并不会传递任何的外部代码给状态机，而仅仅是传递发生的事件。

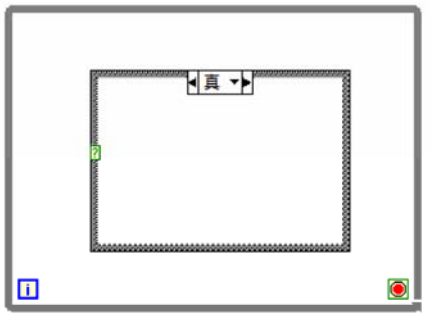


图 2-3 LabVIEW 中的状态机框架

如图 2-3 所示，在 While 循环中加上一个 Case 结构就可以构成一个简单的状态机框架，其中循环主要用来使程序连续执行直至应用程序结束，Case 结构允许程序员定义各种状态。Case 结构的状态通常是由循环的前一次迭代决定的，而位于其子框图中的代码则用于确定状态的变化及执行相应的任务。当然，对于一个最简单的状态机结构而言，它是可以用顺序结构代替的；而对于相对复杂的状态机结构而言，却可以作为测试流程的组织者和管理者。

2.3.2 枚举类型

枚举类型是一种允许程序员自定义元素的数据类型，它可以显示数值对应的具体含义，帮助程序员理解程序中使用的变量值。因此，可以将 LabVIEW 状态机结构中的状态设置为这种数据类型，一个自定义的枚举类型就对应于一个状态机中的状态集合。

枚举型控件位于控件选板的“新式→下拉列表与枚举”子选板中，图 2-4 所示为枚举型变量的控制型、显示型和常量型控件。控件中的枚举值列表可以通过以下方法输入。

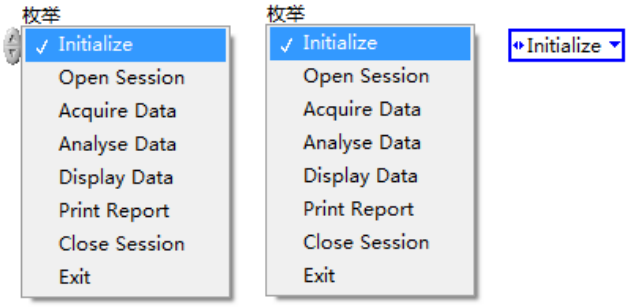


图 2-4　枚举型变量的控制型、显示型和常量型控件

① 使用右键快捷菜单中的“编辑项...”菜单项，弹出“枚举属性：枚举”对话框，显示“编辑项”页，即可以在该对话框中增加、修改或删除枚举值。

② 使用工具面板中的A工具，在枚举型控件中输入枚举值，利用 Shift+Enter 键可以输入下一个枚举值。

③ 使用右键快捷菜单中的“在后面添加项”、“在前面添加项”和“删除项”菜单项。

枚举型控件实质上属于数值型，默认的数据类型是 U16 型（无符号字型），通过类型选择可以把它定义为 U8 型（无符号字节型）或 U32 型（无符号整型），如图 2-5 所示。

读者可能会留意到枚举类型如何与数值型相互转换呢？虽然枚举值实质上属于数值型，但是二者的显示方式和存储方式是不一样的，并且可以相互转换。与其他类型的转换一样，类型转换分为隐式类型转换（强制类型转换）和显式类型转换。

隐式类型转换由 LabVIEW 自动完成，不需要加入任何额外的代码，可以将枚举型控件和数值型控件直接相连，此时相当于完成了类型转换，LabVIEW 会在存在强制类型转换的控件之间加上一个小红点，如图 2-6 所示。

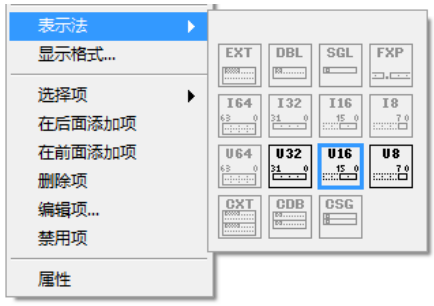


图 2-5　枚举型控件的数据类型（灰色表示该类型不可用）

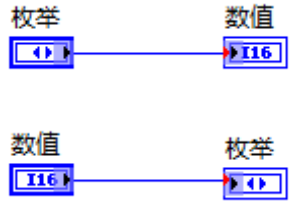
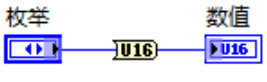


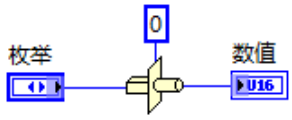
图 2-6　枚举型控件与数值型控件之间的隐式类型转换

中国铁道出版社版权所有

显式类型转换需要添加适当的转换代码。图 2-7(a)、(b)所示分别为使用“转换为无符号双字节整型”函数和“强制类型转换”函数将枚举型控件显式转化为数值型控件。图 2-8 所示为使用“强制类型转换”函数将数值型控件显式转化为枚举型控件。使用显式类型转换是没有红点显示的。



(a) 使用转换为无符号双字节整型函数



(b) 使用强制类型转换函数

图 2-7 将枚举型控件显式转化为数值型控件

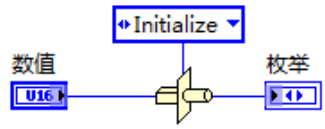
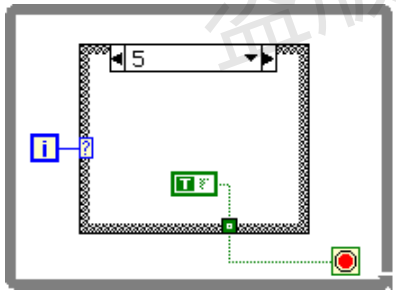


图 2-8 将数值型控件显式转化为枚举型控件

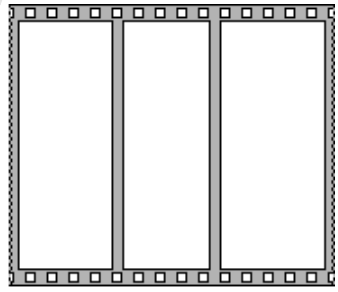
2.3.3 顺序型状态机模式

顺序型状态机是最简单的一种状态机结构，它和顺序结构等价。如图 2-9 所示，在状态机的基本构架上，将循环索引端连接到 Case 结构的选择端口上，并在最后一个 Case 子框图中控制循环结束。显然，这个过程与顺序结构是相同的，其最大的特点是：整个状态序列的顺序是固定的，在程序运行时无法改变。也正是这一点制约了顺序型状态机的应用，因为它妨碍了作为 LabVIEW 优点之一的程序并行运行机制。

但是，在状态之间的数据传递中，二者的实现方式是不同的。前者使用的是移位寄存器，后者使用的是顺序结构的数据通道或者顺序局部变量。典型的实例是图 2-10 所示的利用顺序型状态机计算某个动作运行的时间，这个程序共需要 3 个子框图，调用“定时”函数子面板中的“时间计数器”函数开始计时。“时间计数器”函数返回计算机开机到当前的时间毫秒数。在第 2 个子框图中，放入需要计算的动作模块，并将初始的时间值传递给移位寄存器。在第 3 个子框图中将利用移位寄存器的值进行时间差的计算，得到动作模块运行所消耗的时间，并退出循环。

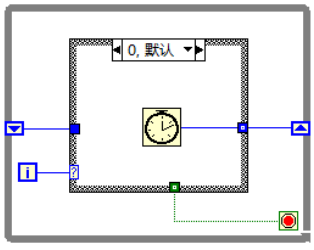


(a) 顺序型状态机模式

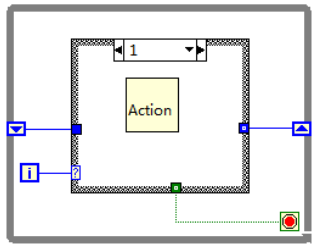


(b) 顺序结构

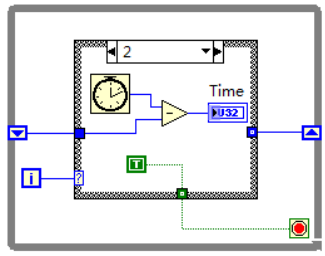
图 2-9 顺序型状态机



(a) 顺序型状态机计时——状态 0



(b) 顺序型状态机计时——状态 1



(c) 顺序型状态机计时——状态 2

图 2-10 利用顺序型状态机计时

电子工业出版社版权所有

作为对比，图 2-11 也给出了利用顺序结构计时的源代码，具体的分析同上，读者可以自行解释其原理。

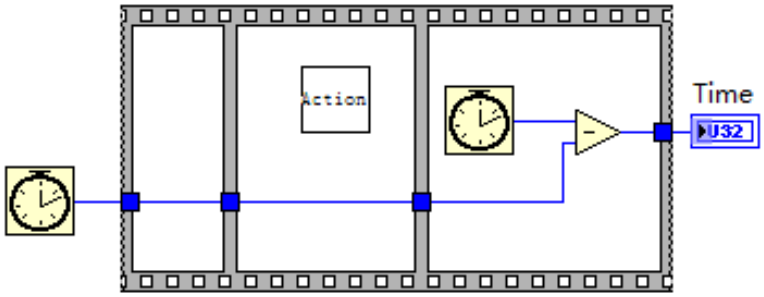


图 2-11　利用顺序结构计时

从上面的分析可知，当需要完成任务的状态顺序是预先确定的时候，顺序型状态机结构比较适合使用，它会直接按照顺序执行。一般而言，移位寄存器型的状态机可以直接替换顺序结构，并且一般都会采用这种替换，因为移位寄存器相对顺序局部变量更容易让人理解。

2.3.4　改进的顺序型状态机模式

在上一节中指出，顺序型状态机的特点限制了它的使用范围。为了能够在程序运行中改变状态序列的执行顺序，可以对其加以改进，采用移位寄存器代替循环索引控制状态机的执行。移位寄存器的高度灵活性使得程序员可以按照实际情况设定状态序列的实际执行顺序，图 2-12 所示的状态机中采用移位寄存器，可以在每个 Case 子框图中指定下一个状态。

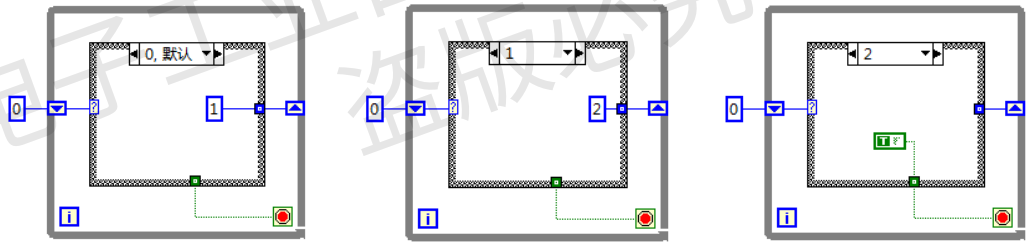


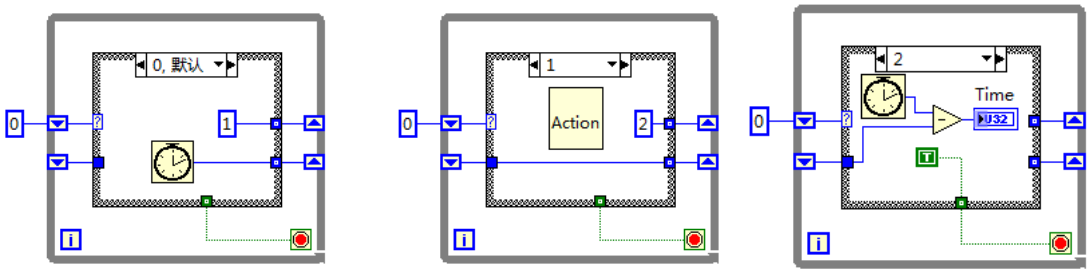
图 2-12　改进的顺序型状态机

现在可以利用改进的顺序型状态机模式改写上节的计时程序，如图 2-13 所示。程序中使用了两个移位寄存器，上面的一个用于控制状态机的运行，如同图 2-10 中的循环索引端 i 一样，但是移位寄存器的输出值可以自定义；另一个是用于数据传递，将第一个状态中得到的时间值传递给第三个状态参与计算。相比图 2-10，该程序使用更加广泛。需要说明的是，并不一定要按照图中的顺序安排各个子框图，可以将图中的 3 个状态放置在任意的 Case 子框图中，只需要利用移位寄存器的输出值将各个状态之间串起来即可。

2.3.5　测试流程型状态机

顺序型状态机还有一个缺点：不便于阅读和修改程序，Case 结构的子框图列表中显示的是数值，不具有任何的实际意义。所以需要找到一种方式，不仅能够保证 Case 结构的正常运行，还要能够很方便地识别 Case 结构中各个子框图的功能。前面提到，枚举型常量实质上也

电子工业出版社版权所有 盗版必究

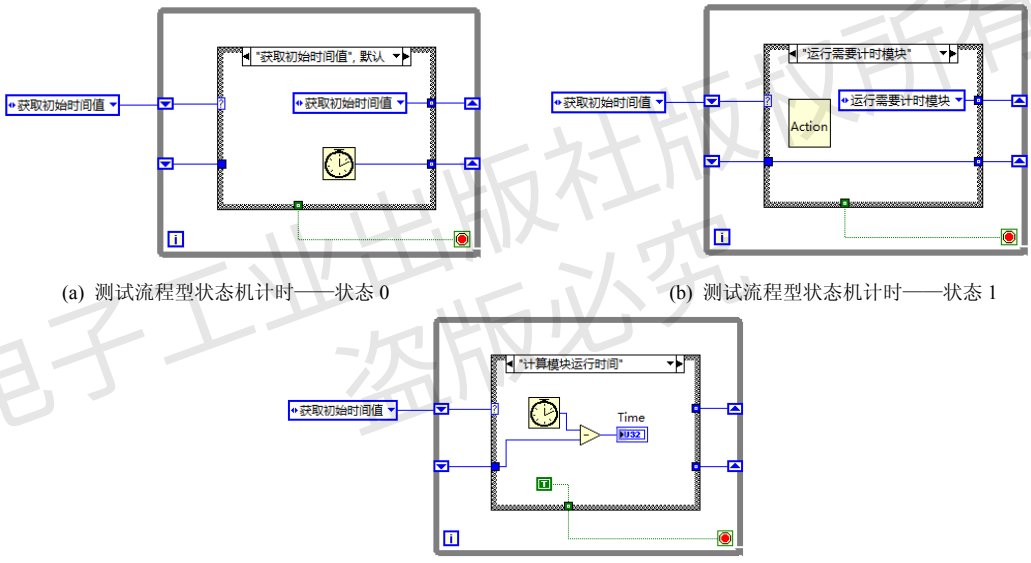


(a) 改进的顺序型状态机计时——状态 0　(b) 改进的顺序型状态机计时——状态 1　(c) 改进的顺序型状态机计时——状态 2

图 2-13　利用改进的顺序型状态机计时

是数字类型，但是枚举值可以使用字符串。这个特点正好与上面的需求相吻合，因此可以考虑使用枚举型常量代替数值型常量控制状态机运行。

图 2-14 所示为利用测试流程型状态机构建同样功能的应用程序，与图 2-10 和图 2-13 相比，其优点不言而喻。程序员可以定义枚举值为各个状态的功能，在 Case 结构的子框图列表中，这些枚举值会显示出来，这样就可以很清楚地知道各个 Case 子框图的具体含义。



(a) 测试流程型状态机计时——状态 0　(b) 测试流程型状态机计时——状态 1

(c) 测试流程型状态机计时——状态 2

图 2-14　利用测试流程型状态机计时

从图 2-14 可以看出，一个枚举型常量需要在程序中的多个地方用到，当在程序开发后期需要增加或修改某个枚举值时，不得不修改所有的枚举型控件，一旦开发大型应用程序时，这项任务将会变得异常烦琐。为了解决这个问题，可以采用前面提到的自定义型控件，将枚举型常量定义为类型自定义控件，而使用时则利用该自定义类型控件的实例。当程序员需要增加、删除或修改枚举值时，只要更新这个自定义类型的枚举型控件就可以了。

在测试流程型状态机的基础上，通过简单转换可以获得很多种类似的状态机模式。如果利用图 2-8 中的枚举型和数值型的相互转换方法，可以修改顺序型状态机模式得到如图 2-15 所示的状态机模式。

电子工业出版社版权所有
盗版必究

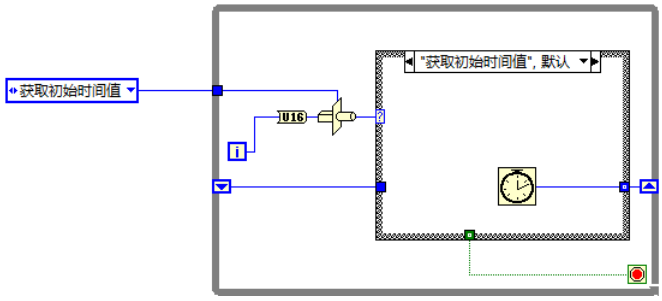


图 2-15　将顺序型状态机模式转换为测试流程型状态机

2.4　消息队列处理模式

前面介绍了应用最为广泛的状态机模式，但是它也存在着一些缺点和限制。例如，某个应用程序共要处理 4 个状态，分别是 A、B、C 和 D，前面板有 3 个按钮分别控制 3 种运行时的状态序列 ABCD、DCAB 和 BDCA。状态机模式无法解决这类问题，因为其中状态序列是事先规定好的，无法动态地根据用户的输入改变状态序列，即使通过选择 Case 等结构解决也非常复杂，不便于程序扩展和维护。

为此，需要引入消息队列处理模式，通过建立队列缓冲区来解决这个问题。这种模式也称为“队列型状态机模式”，但是由于其应用和原理都并不局限于状态机模式，所以为了避免混淆，一般称为“消息队列处理模式”。这种模式通常用于存在于用户界面的程序面板中，也可以用于其他的情况。

2.4.1　消息队列

顾名思义，这种模式需要使用消息队列。通俗地讲，就是建立一个保存消息的队列，这就如同银行的排队系统一样，储户需要办理某种金融业务时，需要先到银行的叫号机处领取一个号码，排队系统会根据储户到来的先后顺序要求储户按照秩序到指定的窗口办理，此时将从系统中把该储户的号码删除，并指定下一位储户办理。我们可以把储户比做应用程序中产生消息的源（即消息发送方），把号码比做消息，把银行窗口比做消息的接收方，把排队系统比做消息处理机构，号码队列比做消息队列。

程序员可以将消息队列看成一段存储空间，用来暂存各种消息。之所以称为队列，是由其消息处理机制决定的，按照 FIFO（先进先出）的思想，需要使用队列的方式处理各种消息。在程序初始化时，首先创建消息队列缓冲器，程序可以根据发生的事件将相应的消息投入到消息队列中，消息处理机构会实时探测消息队列中的消息并按照消息处理机制进行处理；当消息被接收后程序会执行相应的代码，并将该消息从消息队列中删除；当接收到消息“Exit”时，应用程序会停止运行，并释放队列空间。

综上所述，图 2-16 所示为该模式中整个消息处理的过程。需要特别指出的是，消息队列的销毁非常重要，很多程序员都把这一步漏掉，实际上它关系到整个应用程序的稳定性和高效性，这样主要是为了释放队列资源，避免内存泄露。从图 2-16 中还可以看出，消息源将消息投入到消息队列中与消息的探测是同时进行的，这正是 LabVIEW 程序设计的优点，可以充分利用并行程序运行机制处理消息循环。

电子工业出版社版权所有 盗版必究

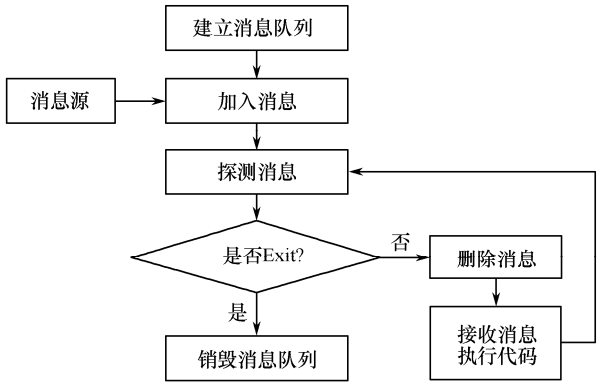


图 2-16　消息处理过程

2.4.2　队列方式

LabVIEW 中有多种方式可以建立消息队列，最常用的是使用队列函数和数组。图 2-17 所示的队列函数位于“同步→队列操作”函数子选板中，共有 9 个函数，常用的 4 个是获取队列引用、元素入队列、元素出队列和释放队列引用，分别用于产生或获取队列、将元素加入队列、将元素移出队列及释放队列空间。

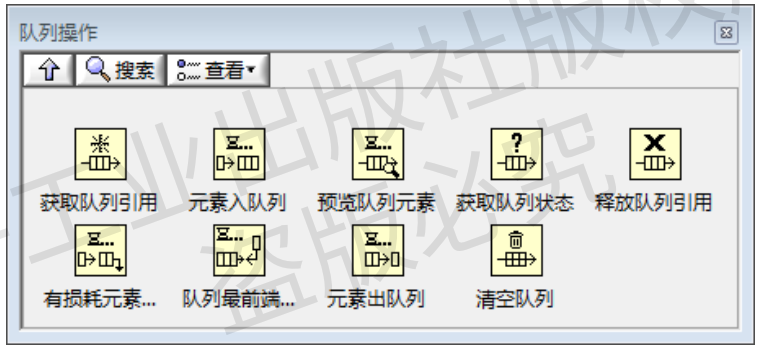


图 2-17　队列处理函数

数组方式的队列具有很多队列函数方式所不具有的优点，这种方式使用数组操作函数对消息进行操作，结构简单，只需要配合移位寄存器使用即可，不需要额外的函数。与队列函数方式不同的是，这种方式不需要使用特别的函数手动销毁队列空间，而是在应用程序退出时会自动销毁队列。

2.4.3　使用数组处理消息队列

现在考虑使用数组方式的消息队列解决本节开始时提到的问题，为了讨论方便，可将问题具体化。假如执行某种操作需要经历 4 部分扫描区域：区域 A、区域 B、区域 C 和区域 D，用户可以使用前面板的按钮控制 3 种扫描顺序，分别是 ABCD、DCAB 和 BDCA。

这个例子与前面不同，它存在一个“等待”状态，在该状态下程序一直探测前面板 3 个按钮的动作。为了醒目起见，将 3 个按钮的标签分别设置为“ABCD”、“DCAB”和“BDCA”，再加上“退出”按钮控制程序结束。此外，再放置一个字符串型显示形控件，命名为“执行顺序”，用以显示单击各个按钮时程序状态执行的顺序，例如单击“ABCD”按钮后，应显示“ABCD”，如图 2-18 所示。

电子工业出版社版权所有 盗版必究

首先需要确定程序的状态，“初始化”状态是必不可少的，它用以复位前面板控件、中间变量值、寄存器值和打开扫描仪器等；“退出”状态用于销毁空间，关闭扫描仪器等；此外，还需要“扫描区域 A”、“扫描区域 B”、“扫描区域 C”和“扫描区域 D”分别控制各个不同的扫描区域。到目前为止，读者可能认为已经定义了所有的状态，但是试想：如果当前程序运行了“初始化”状态，那么下一个状态应该是什么呢？实际上，此时需要等待用户操作，程序处于空闲状态，所以需要定义“等待”状态，其主要用途是根据用户输入决定整个状态序列的执行顺序。图 2-19 所示为将这 7 个状态作为枚举类型的元素，利用自定义控件命令定义一个枚举型控件。

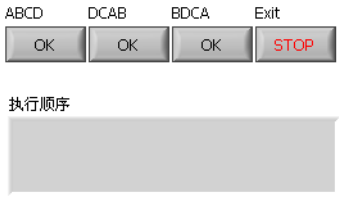


图 2-18　消息队列例程前面板

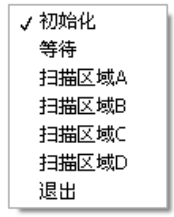


图 2-19　自定义的枚举型控件

图 2-20～图 2-23 所示为该实例的背面板图，整个消息队列采用移位寄存器控制。使用删除数组元素函数将消息从队列中移出，而使用“创建数组”函数将消息加入队列中。当运行到循环里面时，将移出消息队列中最前端的消息，并判断该消息的类型，执行相应的消息处理子框图。在运行中也有可能增加新的消息到消息队列中，并通过移位寄存器传递给下一次循环判断，这样就可以控制消息队列中的消息循环并驱动状态机连续执行。与前面不同，这里是将状态装入一个数组中，构成消息队列，当“初始化”状态执行后，程序转入“等待”状态，等待用户的指令，如图 2-20 所示。

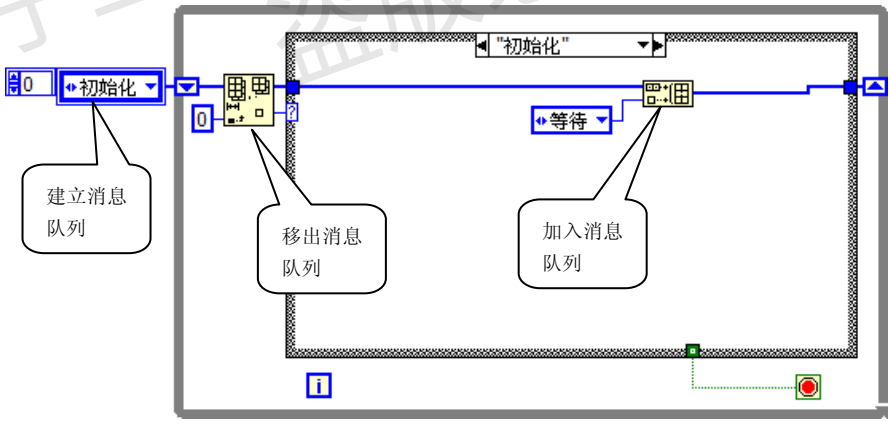


图 2-20　扫描例程——初始化状态

下面考虑程序中最复杂的“等待”状态。如图 2-21 所示，系统将前面板的 4 个按钮集合成一个数组，并实时探测该数组中值为“真”的按钮，并利用 Case 结构将不同情况下对应的状态序列加入消息队列中。如果在这个数组中没有值为“真”的按钮，则函数将返回“–1”，并把“等待”状态再次加入消息队列，如此反复，直到探测到用户动作为止。

一旦用户单击前面板的按钮，这个信息将会被系统探知，并执行相应的消息处理函数，图中列举了 Case 子框图标识为“–1”、“1”和“3”的源代码。当没有搜索到任何“真”值时，

电子工业出版社版权所有 盗版必究

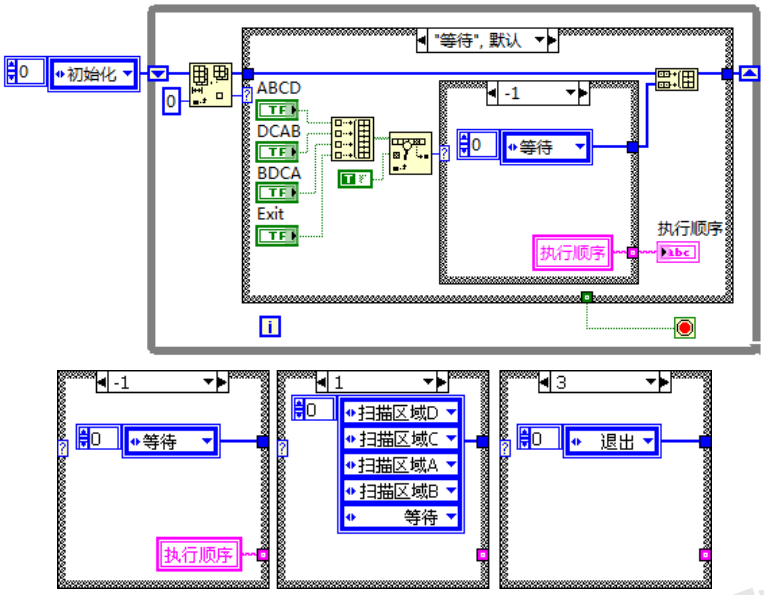


图 2-21　扫描例程——等待状态

便将“等待”状态加入消息队列，以便不断探测消息队列中的值，维持循环的运行。当搜索到“0”～“2”时，将相应需要执行的状态序列加入消息队列。值得注意的是，运行完各个扫描区域的代码后，程序应该继续回到“等待”状态。当搜索到“3”时，表示结束循环并退出应用程序，所以显然应该把“退出”状态加入到消息队列中。“执行顺序”控件是用来显示状态执行的实际序列的，为了方便用户查看状态序列的改变，需要在“等待”状态中保留其现有的值。而当用户单击某个按钮后，就需要清空该控件的值，说明将显示新的状态序列。这里使用了“数据隧道”右键快捷菜单中的“未连线时使用默认”命令，LabVIEW 会将默认值（空值 NULL）传递给控件。

如图 2-22 所示，在单个的扫描区域子框图中，系统执行相应的扫描代码。读者可以根据实际情况添加合适的代码，同时还要把字符“A”加在“执行顺序”字符串的后面，表示经历了“扫描区域 A”。对于其他的扫描区域可以采用相同的做法，这里不再一一描述。

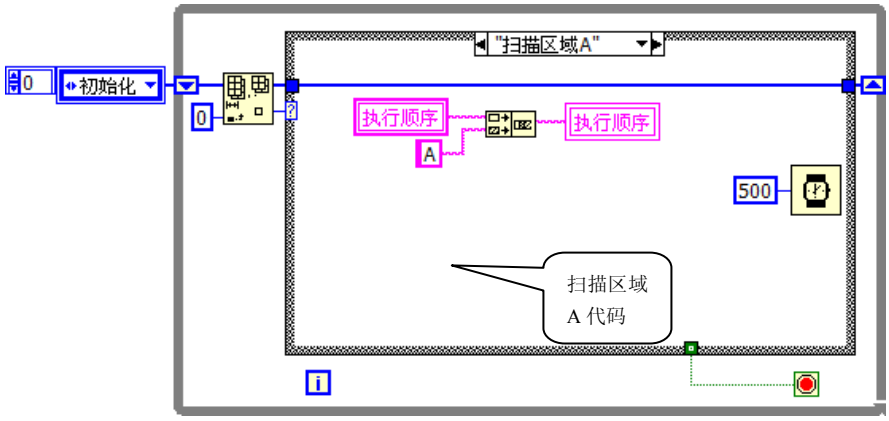


图 2-22　扫描例程——扫描区域“A”状态

最后是“退出”状态，如图 2-23 所示。只有在这个状态才把“真”值传递给 While 循环判断端子，使循环中止，结束程序运行。当然，程序员还可以在这个子框图中添加一些代码，

如销毁释放控件、关闭仪器会话等。

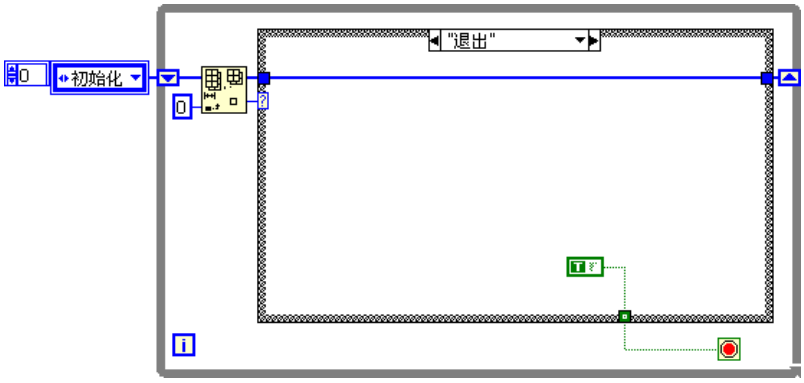


图 2-23　扫描例程——退出状态

至此，我们已经完成了整个程序的构建。运行的效果如图2-24 所示。当单击“ABCD”按钮时，执行顺序显示为“ABCD”，这说明了模式控制的正确性。

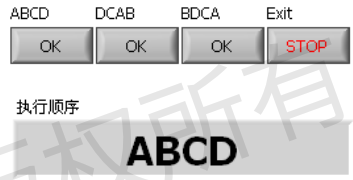


图 2-24　运行效果

2.4.4　使用队列函数处理消息队列

使用队列函数处理消息队列的原理和使用数组方式是一样的，源代码框图如图 2-25 所示。与 2.4.3 节中的源代码相比，二者的构造和流程是相同的。“删除数组中的元素”相当于“从队列中移出元素”函数，“往数组中增加元素”相当于“将元素移入队列”函数，实现的方式也是一样的，这一点读者容易理解，在此不再赘述。

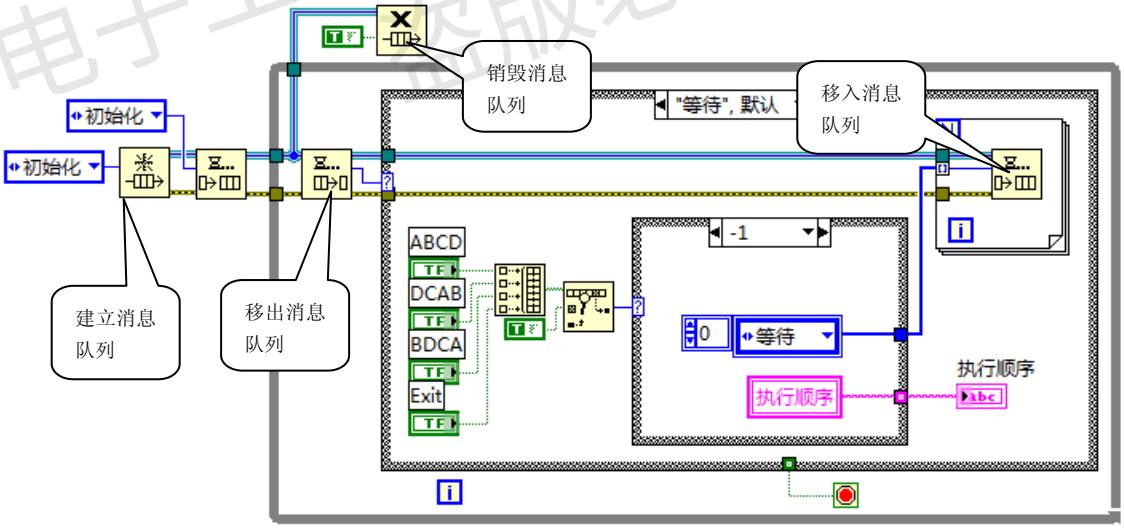


图 2-25　使用队列函数处理消息队列的源代码框图

当然，从图中也可以看出两种方式的不同点。

① 消息传递的方式不同。前者采用移位寄存器方式，而后者则采用 LabVIEW 中的队列技术。

② 释放消息队列的方式不同。前者可以在程序结束时由 LabVIEW 自动释放，后者的队

电子工业出版社版权所有 侵权必究

列资源也可以在程序结束后释放。但是当程序作为子程序时，队列资源并不会随着子程序的结束而自动销毁，而是需要等到主程序结束时才释放。所以有必要手动销毁队列，使用“释放队列引用”函数。

图 2-25 中所示仅仅是一种具体的使用队列函数处理消息队列的方式，实际上还可以使用其他的方式。不仅可以使用枚举型常量作为队列的元素，还可以使用其他任何类型，如字符串型、数值型和布尔型等，读者可以尝试将上例中的枚举型用这些类型的控件代替。

2.5 用户界面事件模式

在实际应用中，状态机模式并不能够满足所有的需求。例如，在上面的例子中，状态机能够捕捉所有的“单击按钮”事件（或者是“按钮控件的值改变”事件），但这是通过搜索的方式完成的，会占用大量的 CPU 资源。另外，状态机模式并不能捕捉其他一些常见的事件，如鼠标移动、关闭窗口和单击某个菜单项等。

为了解决这些问题，程序员可以使用用户界面事件模式。这种交互方式能够处理目前使用到的绝大部分事件，这是 LabVIEW 中用于人机交互的一种强大而高效的模式，可以处理诸如鼠标动作、键盘动作、值改变等事件。此外，事件捕获的方式并不是靠循环结构中的不断搜索来实现的，而是采用中断的实现方式，在事件没有发生期间，CPU 可以处理其他的操作，这就极大地减轻了 CPU 的负担。

2.5.1 事件框架

事件是对活动发生的异步通知。根据来源的不同，事件可分为用户界面事件、外部 I/O 事件和其他程序事件。其中，用户界面事件包括鼠标单击、键盘按键等动作；外部 I/O 事件包括当数据采集完毕或发生错误时硬件定时器或触发器发出信号等情况；其他程序事件可通过编程生成并与程序的不同部分通信。LabVIEW 支持用户界面事件和通过编程生成的事件，但不支持外部 I/O 事件。

在前面的例子中，程序员可以预知程序运行的自然顺序，但是在由事件驱动的程序中，系统中发生的事件将直接影响执行流程。事件驱动程序通常包含一个循环，如图 2-26 所示，该循环等待事件的发生并执行代码来响应事件，然后不断重复以等待下一个事件的发生。程序如何响应事件取决于为该事件所编写的代码。这种程序模式的执行顺序取决于具体所发生的事件及事件发生的顺序。程序的某些部分可能因其所处理的事件的频繁发生而频繁执行，而其他部分也可能由于相应事件从未发生而根本不执行。

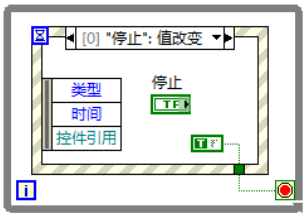


图 2-26　事件结构

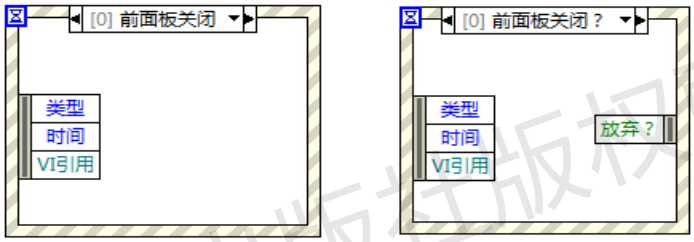
与 Case 选择结构类似，事件结构具有事件选择器标签，用来表示对应框图下的事件。每个事件分支结构的左边框内侧都有“事件数据节点”，用于识别事件发生时 LabVIEW 提供的数据。该节点可纵向调整大小以添加更多数据项，而节点中的每个数据项也可访问任何事件数据元素。根据各个分支所要处理的不同事件，该节点在每个分支中指定相应的数据元素。如果配置单个分支来处理多个事件，那么该“事件数据”节点仅提供这些事件所共有的事件数据元素。

电子工业出版社版权所有 盗版必究

2.5.2 用户界面事件

用户界面事件有两种类型：通知事件和过滤事件，如图 2-27 所示。通知事件表明某个用户操作已经发生，如用户改变了控件的值。它用于在事件发生且 LabVIEW 已对事件处理后对事件作出响应，可以配置一个或多个事件结构对一个对象上同一通知事件作出响应。当事件发生时，LabVIEW 会将该事件的副本发送到每个并行处理该事件的事件结构中。

过滤事件允许用户对发生的事件作出响应，人为控制事件的发生。在过滤事件的事件结构分支中，可在 LabVIEW 结束处理该事件之前验证或改变事件数据，或完全放弃该事件以防止数据的改变影响到 VI。例如，将一个事件结构配置为“放弃前面板关闭？”过滤事件，并不是一定会将 VI 的前面板关闭，用户可以决定是否需要撤销该事件。这种类型的事件分支结构的右边框内侧有“过滤数据节点”，可以改变对事件参数的一些设置。可将新的数据值连接至这些接线端以改变事件数据，如果不对某一数据项连线，那么该数据项将保持不变，当然也可将“真”值连接至“放弃？”接线端以完全放弃某个事件。



(a) 通知事件　　(b) 过滤事件

图 2-27　用户界面事件

2.5.3 用户自定义事件

根据事件的发出源，事件可以抽象地分为用户界面事件和用户自定义事件。前面提到的都是界面事件，如鼠标单击、值改变、菜单项被选中和键盘单击等，这些事件无法人为控制和修改。此外，还可以通过编程创建和命名自己的事件，即用户自定义事件，来传送用户自定义数据。与队列和通知器一样，用户事件允许应用程序的不同部分之间进行异步通信，还可以在同一事件结构中处理用户界面和用户自定义事件。

用户自定义事件需要使用的函数如图 2-28 所示，与队列函数类似，该函数集包括创建自定义事件、产生自定义事件、取消自定义事件、销毁自定义事件和注册自定义事件。要定义用户事件，可将程序框图对象（如前面板接线端或程序框图常量）连接到创建用户事件函数。该对象的数据类型定义了用户事件的数据类型，该对象的标签称为用户事件的名称。如果数据类型是簇，则该簇中每个字段的名称和类型便对用户事件所传送的数据进行定义。如果数据类型不是簇，则用户事件将传送该类型的单个值，同时该对象的标签将成为用户事件和单个数据元素的名称，所以自定义事件中元素的标签不能为空，否则程序无法运行。

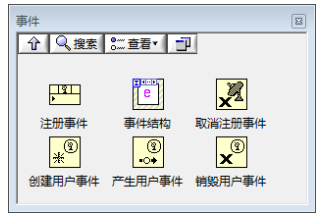


图 2-28　用户自定义事件函数

图 2-29 所示为一个用户自定义事件的例子，其中选用了字符串型（String）作为自定义事件的元素类型。将标签为“我的事件”的字符串常量连接到“创建用户事件”函数，输出

电子工业出版社版权所有 盗版必究

端是一个严格类型的引用。该引用传送了用户事件的名称和数据类型，可以直接连接到“注册事件”函数的事件源输入端（这一点会在 2.5.4 节中详细阐述），再将“注册事件”函数的事件注册引用句柄输出端连接到该事件结构左侧的动态事件接线端。通过编辑事件对话框可配置事件结构中的一个分支以处理该事件，用户事件的名称就会出现该对话框中事件源部分的动态子标题下。

“产生用户事件”函数用于接收一个用户事件引用句柄和事件数据值，将用户事件及相关数据传送到应用程序的其他部分。本例中是一个字符串常量，该常量会传送给操作系统，从而触发用户事件“我的事件”。

当不再需要用户事件时就需要取消用户事件注册，将用户事件句柄连接到“销毁用户事件”函数的输入端就可以销毁用户事件。此时需要将“取消注册事件”函数的错误输出端连接到“销毁用户事件”函数的错误输入端，以确保这两个函数按正确的顺序执行。当然，当顶层 VI 结束运行时，LabVIEW 将自动取消所有事件注册并销毁已有的用户事件。但如果某个自定义事件不再使用，仍然建议程序员手动取消注册并销毁用户事件，以节省内存资源。

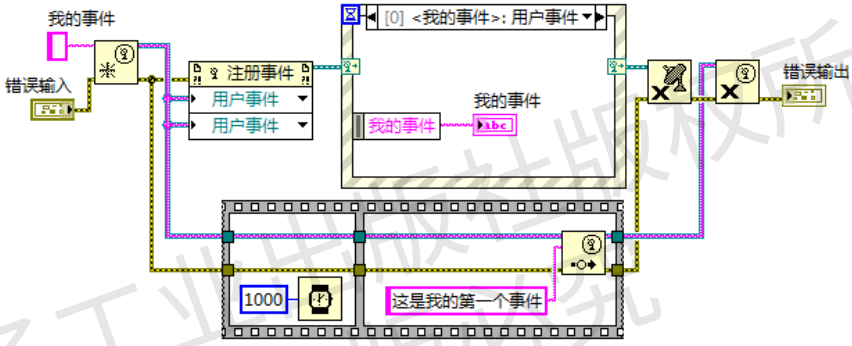


图 2-29　用户自定义事件

2.5.4　事件注册

自定义事件并不能使程序产生对应的事件，还需要事件结构能够识别这些事件，这就是“事件注册”。当向事件结构注册自定义的事件后，使用“产生用户事件”函数时，LabVIEW 才会将用户事件及相关事件数据注册到与事件队列中。LabVIEW 可产生多种不同的事件，为避免产生不需要的事件，可以使用事件注册来指定希望 LabVIEW 通知的事件。从事件类型上而言，LabVIEW 支持静态和动态两种事件注册模式。其中动态事件不仅可以注册，还可以修改。

1．静态事件注册

只有用户界面事件才可以进行静态事件注册，允许指定 VI 在程序框图上的事件结构的每个分支具体处理该 VI 在前面板上的那些事件。首先需要选择事件源，它可以是程序、VI 或某个控件。同时选择一个事件源可产生特定的事件，如前面板大小调整、值改变等。其次，根据应用程序的需求，编辑该分支来处理事件数据。LabVIEW 在 VI 运行时将自动注册这些事件，一旦 VI 开始运行，事件结构便开始等待事件，程序员无法在程序运行时改变事件结构所处理的事件。

假设需要完成某个虚拟仪器的开关控制，在前面板放置一个布尔型控件，使用静态事件注册方法，如图 2-30 所示。当探测到发生“开关”控件的“值改变”事件且新值为“真”时，会提示“开始测试？”；当值为“假”时，会退出虚拟仪器。

电子工业出版社版权所有 盗版必究

2．动态事件注册

动态事件注册通过将事件注册与 VI 服务器相结合，允许在运行时使用应用程序、VI 和控件引用来指定希望产生事件的对象。由于动态事件注册可以完全控制 LabVIEW 产生事件的类型和时间，所以在控制 LabVIEW 产生何种事件及何时产生事件方面比静态事件注册更为灵活。但是，动态事件注册比静态事件注册复杂，它需要将 VI 服务器引用和程序框图函数同时使用以注册和取消注册事件，而无法通过事件结构的配置信息自动注册。此外，动态事件可以使事件仅在应用程序的某个部分发生，或在应用程序运行时改变产生事件的 VI 或控件。因此，使用动态事件注册可以在子 VI 中处理事件而不是仅在产生事件的 VI 中处理事件。

处理动态注册的事件主要有以下 4 个步骤：

① 获取要处理事件对象的 VI 服务器引用；

② 将 VI 服务器引用连接至“注册事件”函数以注册对象的事件；

③ 将事件结构放在 While 循环中，等待处理对象事件至出现终止条件为止；

④ 通过取消注册事件函数停止事件发生。

在图 2-30 中，使用了静态事件完成仪器开关的模拟。现在需要扩展其功能，假定该仪器用于求取给定 5 个输入值的最大值、最小值和平均值，命名为“最值器”。前面板如图 2-31 所示，但是只有当仪器开关设置为“真”时，即仪器打开后，仪器面板上的“输入值”控件才会起作用。当改变控件的值后，3 个输出量也会实时改变。

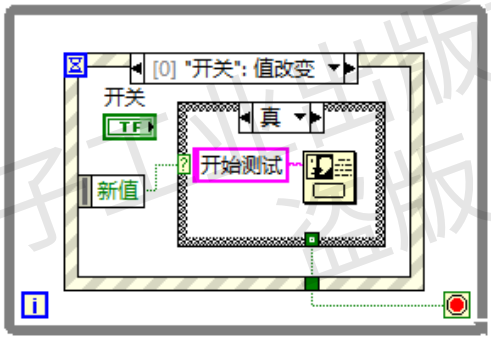


图 2-30　静态事件注册

图 2-31　“最值器”前面板

“最值器”程序框图中的事件结构由 3 个子框图构成，其中使用了动态注册事件、用户界面事件和用户自定义事件。用户自定义事件用于“初始化”命令（见图 2-32），并且使用动态注册方式。在事件结构的外部还产生自定义事件，这样程序进入事件结构后可以直接进入这个自定义事件的子框图中。该框图主要用于将开关量设置为“假”（复位），并将输出显示端清空。

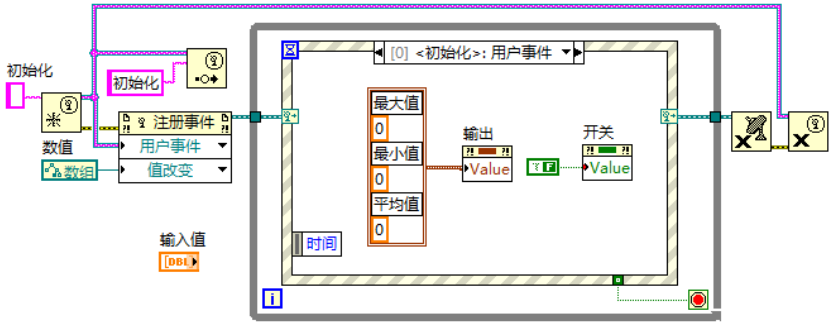


图 2-32　“<初始化>：用户事件”事件框图

另外一个需要动态注册的事件是“输入值”控件的“值改变”事件。如果将这个事件设置为静态注册，则无法保证只有当“开关”控件为“真”时才有效，所以需要引入“动态注册事件”。这样，当“开关”值设为“真”时，才将该事件注册以供事件结构监听；否则就不注册，此时即使改变“输入值”控件的值，也不会有任何作用。

在动态注册数组“值改变”事件时，将对象选择为常量数组，因此当改变前面板的“输入值”控件时，并不能触发数组“值改变”这个事件。而在“"开关"：值改变”子框图中（见图 2-33），当“开关”控件为“真”时，需要动态修改注册事件。将初始化中的数组型常量引用修改为“输入值”控件的引用，再连接到“动态注册端子”，此后再改变控件“输入值”的值，输出就会发生相应的变化。

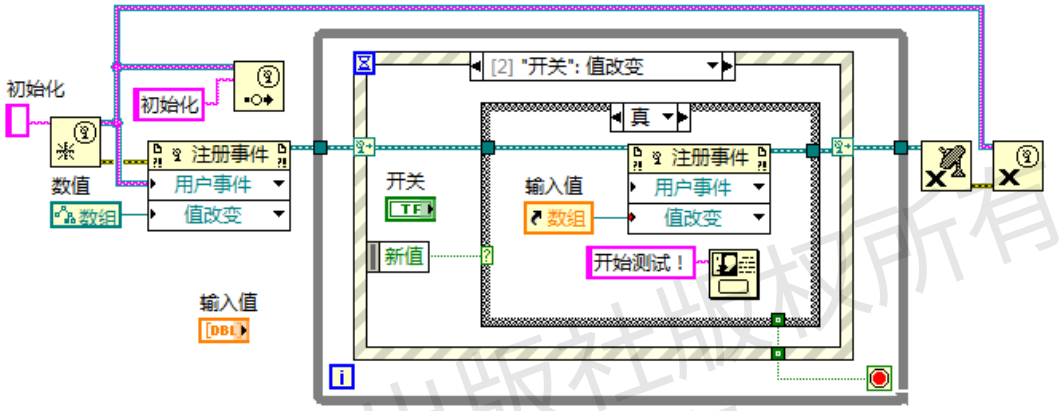


图 2-33　“"开关"：值改变”事件框图

在“<数值>：值改变”这个子框图中（见图 2-34），将获得的值转换为数组型值，并得到数组中值的最大值、最小值和平均值。在循环结束后需要取消注册事件并销毁自定义事件。

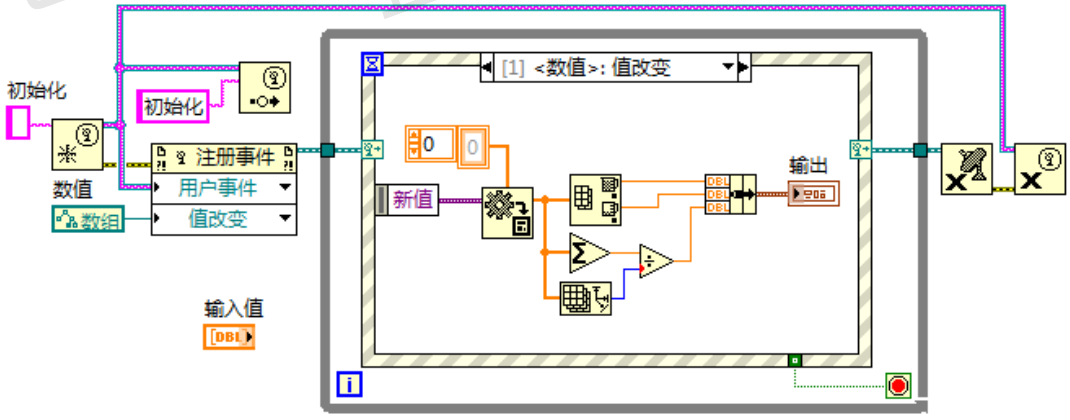


图 2-34　“<数值>：值改变”事件框图

2.5.5　用户界面事件示例

为了统一和比较，仍然采用 2.4.3 节中的例子，实现“A，B，C，D”4 个区域不同顺序的扫描。前面板如图 2-18 所示，本节将使用用户界面事件方式解决同样的问题，分别采用 1 个动态事件注册和 4 个静态事件注册，因此共分为 5 个子框图，分别是：“"ABCD":

值改变”、“"DCAB"：值改变”、“"BDCA"：值改变”、“"Exit"：值改变”和“<扫描区域>：用户事件”。

首先需要创建“扫描区域”这个用户自定义事件，并注册该事件。在“<扫描区域>：用户事件”子框图中（见图 2-35），需要建立 Case 结构，执行各个区域的扫描代码，并且在“执行顺序”控件中显示相应的区域字符串代码。此外，在程序运行的初期，还需要执行简单的“初始化”过程，即初始化“执行顺序”控件的值。当然如果初始化的代码太多，可以使用图 2-32 中的方法，使用用户自定义事件执行初始化。

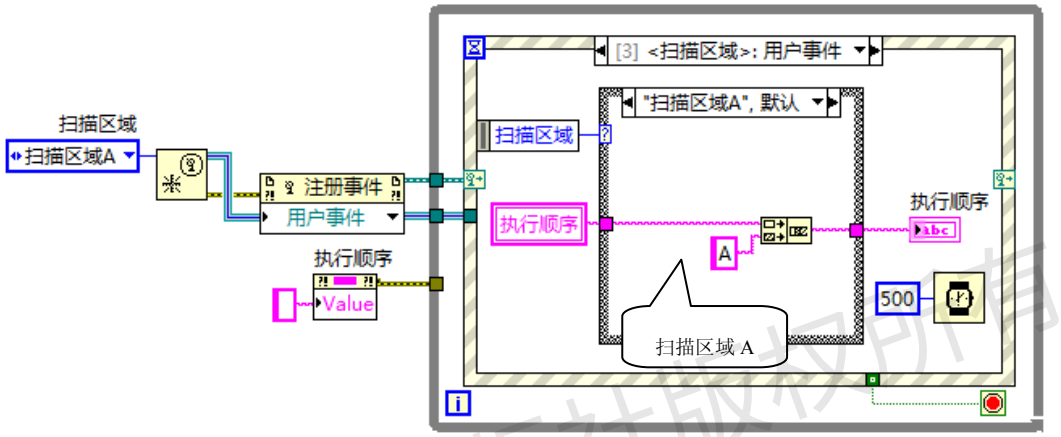


图 2-35　“<扫描区域>：用户事件”事件框图

在“"ABCD"：值改变”、“"DCAB"：值改变”和“"BDCA"：值改变”3 个事件子框图中，均采用“创建用户自定义事件”函数，并按照各自的扫描顺序将值传送给用户自定义事件，这样在自定义事件子框图中会根据接收到的枚举值选择相应的扫描代码。由于其他控件的“值改变”事件与图 2-36 中的代码类似，在此就不一一列出，读者可以尝试自己编写其他控件的事件处理代码。

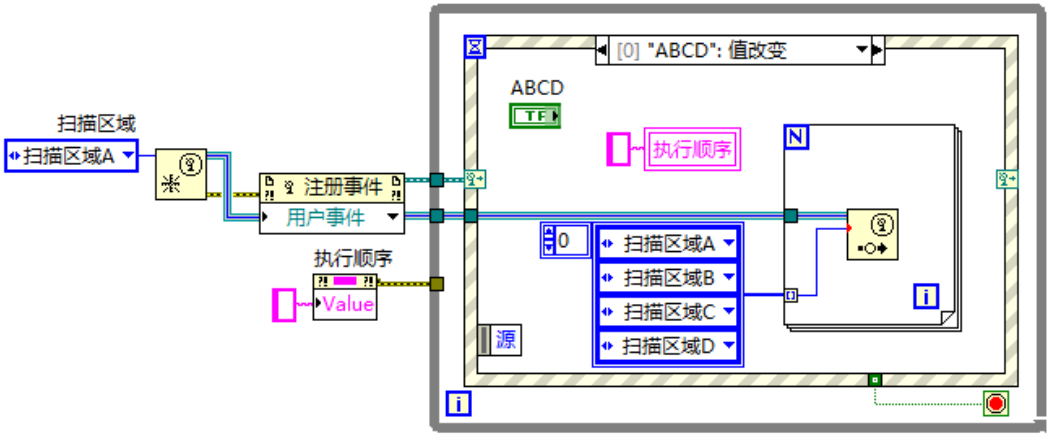


图 2-36　“"ABCD"：值改变”事件框图

在“"Exit"：值改变”子框图中，需要将“真”值传送给“While 循环终止端子”，此时程序将会退出 While 循环，停止运行，如图 2-37 所示。

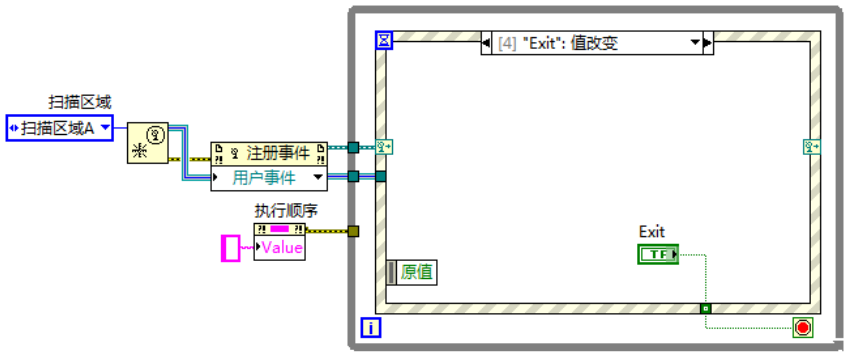


图 2-37　“"Exit"：值改变”事件框图

2.6　状态机-用户界面事件混合模式

前面几节分析了 LabVIEW 中各种设计模式的具体做法和优缺点，从中可以看出，使用状态机和事件结构各有优劣。相比状态机而言，事件结构能够捕获一些状态机无法获取的事件，并且采用中断方式处理消息，能够更加节省计算机资源。但是，事件结构的缺点也是很明显的：设置初始化时需要使用用户自定义事件，不仅占据背面板空间，而且无法利用 Case 结构分层次的优点，特别是当创建多个用户自定义事件时，整个程序会显得杂乱无章，打乱了程序的流程。另外，如图 2-36 所示，当需要连续触发多个事件时需要使用循环结构，不如“创建数组”函数使用起来方便快捷。结合二者的优点，程序员可以使用状态机—用户界面事件混合模式（Compound Design Patterns）设计应用程序，这样可以取二者的长处，有效地组织应用程序。

2.6.1　状态机与用户界面事件的结合

顾名思义，这种混合模式实质上是将状态机模式和用户界面事件模式融为一体，可以有效地避免单个模式带来的缺点。图 2-38 所示为这种模式的具体做法，其主体框架仍然由状态机构成，唯一不同的是在“等待”子框图中，不再是使用“搜索数组”函数获取前面板控件值的改变，而是采用事件结构探测各种发生的事件。这样可以充分发挥事件结构的优点，既不会遗漏部分事件，也不会过于占据计算机资源。

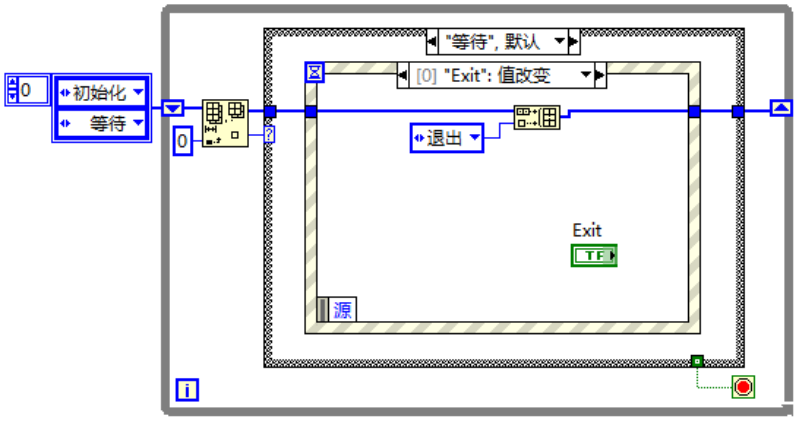


图 2-38　状态机与用户界面事件的结合

模式的运行过程与使用数组处理消息队列类似，只是在事件结构中获取系统事件。如图 2-38 中的“"Exit"：值改变”事件，当探测到事件发生时，将“退出加入到消息队列中”。此时，下一次循环将会执行“退出”状态，此时应用程序退出运行。

2.6.2 状态机-用户界面事件示例

为了比较，这里仍然采用前面的区域扫描例子，按照前面提到的步骤，在状态机中选择 7 种状态：初始化、等待（默认）、扫描区域 A、扫描区域 B、扫描区域 C、扫描区域 D 和退出。而在等待子框图中，事件结构用于获取前面板 4 个按钮的“值改变”事件。如图 2-39 所示，以“"ABCD"：值改变”这个事件为例，在事件框图中依次将“扫描区域 A”、“扫描区域 B”、“扫描区域 C”、“扫描区域 D”和“等待”这 5 个状态加入消息队列。需要说明的是，该模式并没有严格定义哪一种状态机模式和用户界面事件模式搭配，也可以把枚举型换成字符串型，程序员可以根据自己的实际要求选择相应的模式或者其变体。

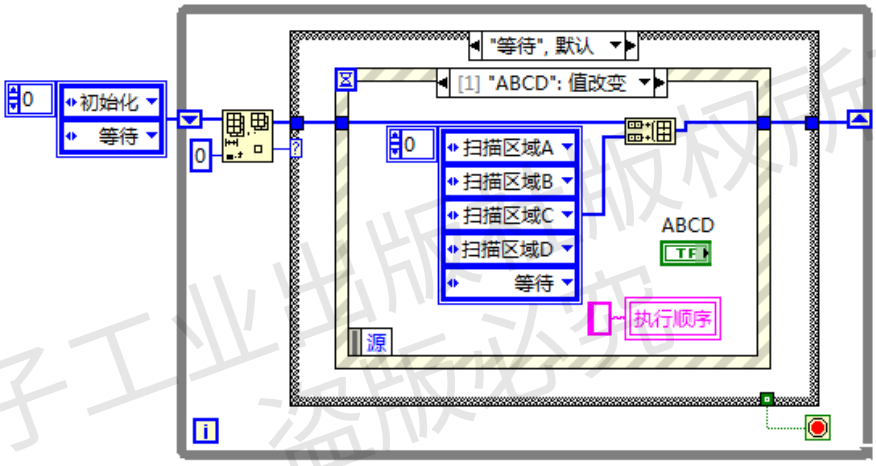


图 2-39　状态机与用户界面事件结合模式——等待

在各个 Case 框图中，可以放置消息响应代码。以“扫描区域 A”消息为例（见图 2-40），Case 框图中是对 A 区域执行扫描的代码，在本例中只是改变输出值“执行顺序”，用于证明流程执行的正确性。

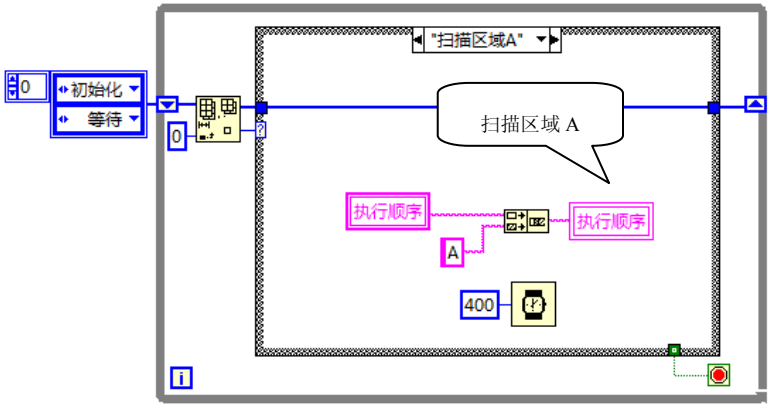


图 2-40　状态机与用户界面事件结合模式——扫描区域 A

电子工业出版社版权所有 盗版必究

2.7 其 他 模 式

前面介绍的所有模式都具有很强的扩展性，在流程控制中能够满足绝大多数应用的需要。但是在某些特殊的应用中仍然存在局限性，需要更具有针对性的程序框架来满足这些需求。例如，前面介绍的模式在同一时刻仅仅能够处理一个消息事件，这是由消息队列导致的。同时出列的消息只能有一个，这样下一个消息响应代码必须等到上一次响应完成后才能执行。试想，如果某个测试系统需要在采集数据的同时将数据记录下来，而不是等到数据采集完成后才记录数据，这种应用是无法利用上面的任何一种模式处理的。

因此，本节结合一些 LabVIEW 程序员在开发中经常遇到的开发问题，给出几种非常具有实用价值的 LabVIEW 程序代码框架，作为专用型的设计模式。

2.7.1 主从线程模式

主从（Master/Slave）线程模式通常应用于具有两个或多个同时发生的并且拥有不同运行速率的线程的程序中。例如，假定需要在测量某些电压值的同时将这些数据写入磁盘中，如果电压的采样率是每 100ms 获取一条曲线（100 点）并显示在图形面板上，而记录的速率要比这低得多，是每5s 记录一次，同样也需要放置一些按钮在前面板实现打印、退出等功能。很多人会认为可以把这两个动作放在同一个 While 循环中完成，这的确是可以满足任务的并行要求，但是却无法满足时间的要求。如果以记录的速率时间为标准，则下一次采集必须等到上一次循环结束才完成，这样就会遗漏一些真实采集到的测量数据；而记录的过程由于涉及与磁盘交互，会比较慢，当速率太快时，这一过程将无法完成。另外，用户也无法做到改变数据采集的速率而不影响数据记录的速率。

图 2-41 主从线程模式框架

图 2-41 所示为主从线程模式的具体做法，可以看出整个框架由两个 While 循环组成，而主循环可以控制从循环的结束。程序员可以在两个循环中放置不同的任务，二者是互不影响的，数据通信采用全局变量或共享变量的形式。为了能够使用 Stop 控件的局域变量，必须把控件设置为“Switch”模式，而不能是“Latch”模式。最后当循环结束后将 Stop 的值复位，这样可以很好地处理异步线程之间的联系。

当然，上面的框架也不是一成不变的，仅仅是给出了一个参考。主模式可以采用前面提到的任何一种模式结构，这样也比较容易捕获系统事件和用户自定义事件。还有一种情况是同步线程，在图 2-41 中，主从之间只有附属关系，并没有发生交互。如果从循环只有当主循环满足一定的情况才可以运行，即由主循环控制从循环的运行，这就需要使用 LabVIEW 中的事件发生模块（Occurrence），这一点会在第 3 章中详细讨论。

2.7.2 生产消费模式

主从模式的数据通信是利用全局变量、局域变量或共享变量实现的，由于这些变量的每

次复制都是原始数据的一个副本，占据了大量的空间。实际上，只需要使用一部分缓冲区作为数据存储的中间部分，这需要借助队列技术，也称为“生产消费模式”。

如图 2-42 所示，整个过程就是一个生产者和消费者之间相互作用的过程。数据从一端不断地流入存储单元，而同时也从另一端流出存储单元，这样可以保证在一定的存储空间中连续地进行数据的交互，实现数据采集、数据共享和数据处理。

图 2-43 所示为生产消费模式的具体做法。这种模式仍然采用两个简单的 While 循环，但是数据通信采用队列结构，一方面将采集到的数据传送给队列空间，另一方面从队列空间中读出数据，二者是相互独立而又紧密联系的。同样，生产者和消费者实现的具体形式并不局限于 While 循环，也可以是上面提过的任何模式，这要由程序员根据实际情况选择。

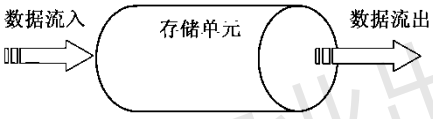


图 2-42　生产消费模式原理

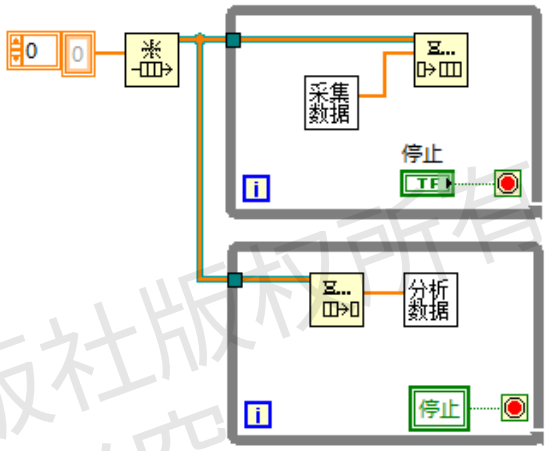


图 2-43　生产消费模式

2.7.3　后台服务模式

还有一类特殊的应用程序——后台服务程序，这种程序一般不需用户的干涉而在计算机后台运行，即不需要显示应用程序的前面板。例如，一个网络服务监听程序，程序只需要接收与客户端 Sockets 的连接，实现三次握手。连接完成后将引用交给前端应用程序处理，同时继续监听，因此该程序的前面板并不需要显示出来，不存在用户交互界面。实际上，LabVIEW 也提供了这个问题的解决途径——VI 属性和方法接口，通过这些接口程序员可以很方便地控制 VI 的一些固有属性和运行时的状态。

使用图 2-44 所示的方法，可以使 VI 的前面板不显示但是载入内存，并且利用 LabVIEW 提供的 VI 属性和方法接口，程序员能够控制 VI 的运行、停止及设置 VI 前面板控件的参数，获取 VI 运行的结果。图中仅仅使用了“Run VI”方法以控制 VI 的运行，实际上 LabVIEW 提供了丰富的控制 VI 接口，足以使程序员控制后台运行的 VI。

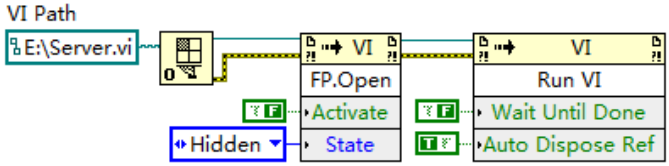


图 2-44　后台服务模式

电子工业出版社版权所有 盗版必究

2.7.4 应用程序启动模式

利用 LabVIEW 打开一个大型的应用程序源文件时，会出现“Load VI…”窗口，该窗口显示了将当前 VI 调用的所有子 VI 载入内存的过程，用以标识 VI 的打开进度。在大型应用程序中这是有必要的，但该界面是 LabVIEW 自带的，程序员无法根据自己的实际应用改变界面的显示方式，如语言等。通常程序员需要在该界面上显示应用程序的名字和版本，这就是通常所说的“开机画面”。当然，在子 VI 装载完成之后，该启动画面将自动退出并启动主应用程序。

图 2-45 “网络实验系统”源代码文件逻辑组织方式

以“网络实验系统”为例，源代码文件的逻辑组织方式如图 2-45 所示。“启动应用程序.vi”用做开机画面，利用应用程序启动模式编写；“网络实验系统.vi”用做整个系统的主程序；而 SubVI 文件夹中存放该系统的所有子 VI。这里并不是为了介绍“网络实验系统”，而只是利用该系统说明应用程序启动模式的具体做法，我们关心的是如何利用应用程序模式启动主程序。

在“启动应用程序.vi”中，前面板如图 2-46 所示，界面上显示了该系统的名称、版权信息等。此外，还显示了当前程序加载子模块的进度和具体名称。程序员也可以适当增加其他的模块，但是应确保整个画面的简洁。

图 2-46 “应用程序启动模式”前面板

程序框图如图 2-47 所示，根据图 2-45 显示的系统源文件逻辑组织方式，将 SubVI 目录下的子程序一一加载直至完成或者出错。图中首先使用“路径”函数获取所有子 VI 的文件名，然后再使用“打开 VI 引用”函数加载这些 VI，最后启动主程序——“网络实验系统.vi”并将启动画面关闭。在实际应用中，文件组织的方式并不一定是按照图 2-45 所示的安排，此时只需要简单地修改各个子模块调用的目录即可。

2.7.5 代理模式

LabVIEW 默认在主程序打开时就将其调用的所有子 VI 载入内存。在大型的应用程序中，子程序会成百上千，这就势必会减慢应用程序打开的速度。为了解决这个问题，可以在 LabVIEW 应用程序中引入“迟载入”技术，即在顶层 VI 需要某个子 VI 时才将其载入内存。这对于那些使用非常频繁的子 VI 特别有用，这种模式就是代理模式。

电子工业出版社版权所有 盗版必究