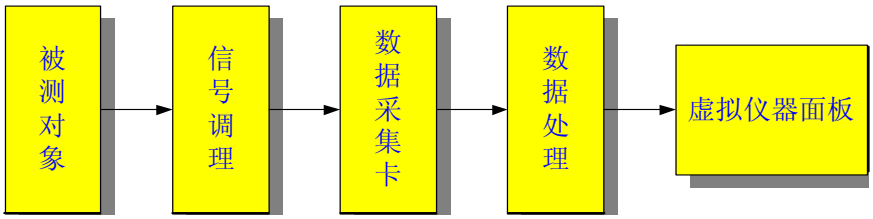
第一章 虚拟仪器及 LabVIEW 入门

1．1 虚拟仪器概述

虚拟仪器（virtual instrumention）是基于计算机的仪器。计算机和仪器的密切结合是目前仪器发展的一个重要方向。粗略地说这种结合有两种方式，一种是将计算机装入仪器，其典型的例子就是所谓智能化的仪器。随着计算机功能的日益强大以及其体积的日趋缩小，这类仪器功能也越来越强大，目前已经出现含嵌入式系统的仪器。另一种方式是将仪器装入计算机。以通用的计算机硬件及操作系统为依托，实现各种仪器功能。虚拟仪器主要是指这种方式。下面的框图反映了常见的虚拟仪器方案。



虚拟仪器的主要特点有：

- 尽可能采用了通用的硬件，各种仪器的差异主要是软件。
- 可充分发挥计算机的能力，有强大的数据处理功能，可以创造出功能更强的仪器。
- 用户可以根据自己的需要定义和制造各种仪器。

虚拟仪器实际上是一个按照仪器需求组织的数据采集系统。虚拟仪器的研究中涉及的基础理论主要有计算机数据采集和数字信号处理。目前在这一领域内，使用较为广泛的计算机语言是美国 NI 公司的 LabVIEW。

虚拟仪器的起源可以追朔到 20 世纪 70 年代，那时计算机测控系统在国防、航天等领域已经有了相当的发展。PC 机出现以后，仪器级的计算机化成为可能，甚至在 Microsoft 公司的 Windows 诞生之前，NI 公司已经在 Macintosh 计算机上推出了 LabVIEW2.0 以前的版本。对虚拟仪器和 LabVIEW 长期、系统、有效的研究开发使得该公司成为业界公认的权威。

普通的 PC 有一些不可避免的弱点。用它构建的虚拟仪器或计算机测试系统性能不可能太高。目前作为计算机化仪器的一个重要发展方向是制定了 VXI 标准，这是一种插卡式的仪器。每一种仪器是一个插卡，为了保证仪器的性能，又采用了较多的硬件，但这些卡式仪器本身都没有面板，其面板仍然用虚拟的方式在计算机屏幕上出现。这些卡插入标准的 VXI 机箱，再与计算机相连，就组成了一个测试系统。VXI 仪器价格昂贵，目前又推出了一种较为便宜的 PXI 标准仪器。

虚拟仪器研究的另一个问题是各种标准仪器的互连及与计算机的连接。目前使用较多的是 IEEE 488 或 GPIB 协议。未来的仪器也应当是网络化的。

1．2 LabVIEW 是什么？

LabVIEW（Laboratory Virtual instrument Engineering）是一种图形化的编程语言，它广泛地被工业界、学术界和研究实验室所接受，视为一个标准的数据采集和仪器控制软件。LabVIEW 集成了与满足 GPIB、VXI、RS-232 和 RS-485 协议的硬件及数据采集卡通讯的全部功能。它还内置了便于应用 TCP/IP、ActiveX 等软件标准的库函数。这是一个功能强大且灵活的软件。利用它可以方便地建立自己的虚拟仪器，其图形化的界面使得编程及使用过程

都生动有趣。

图形化的程序语言，又称为“G”语言。使用这种语言编程时，基本上不写程序代码，取而代之的是流程图或流程图。它尽可能利用了技术人员、科学家、工程师所熟悉的术语、图标和概念，因此，LabVIEW 是一个面向最终用户的工具。它可以增强你构建自己的科学和工程系统的能力，提供了实现仪器编程和数据采集系统的便捷途径。使用它进行原理研究、设计、测试并实现仪器系统时，可以大大提高工作效率。

利用 LabVIEW，可产生独立运行的可执行文件，它是一个真正的 3 2 位编译器。像许多重要的软件一样，LabVIEW 提供了 Windows、UNIX、Linux、Macintosh 的多种版本。

1．3　LabVIEW 的运行机制

1．3．1　LabVIEW 应用程序的构成

所有的 LabVIEW 应用程序，即虚拟仪器（VI），它包括前面板（front panel）、流程图（block diagram）以及图标/连结器(icon/connector)三部分。

前面板

前面板是图形用户界面，也就是 VI 的虚拟仪器面板，这一界面上有用户输入和显示输出两类对象，具体表现有开关、旋钮、图形以及其他控制（control）和显示对象（indicator）。图 1 所示是一个随机信号发生和显示的简单 VI 是它的前面板，上面有一个显示对象，以曲线的方式显示了所产生的一系列随机数。还有一个控制对象——开关，可以启动和停止工作。显然，并非简单地画两个控件就可以运行，在前面板后还有一个与之配套的流程图。

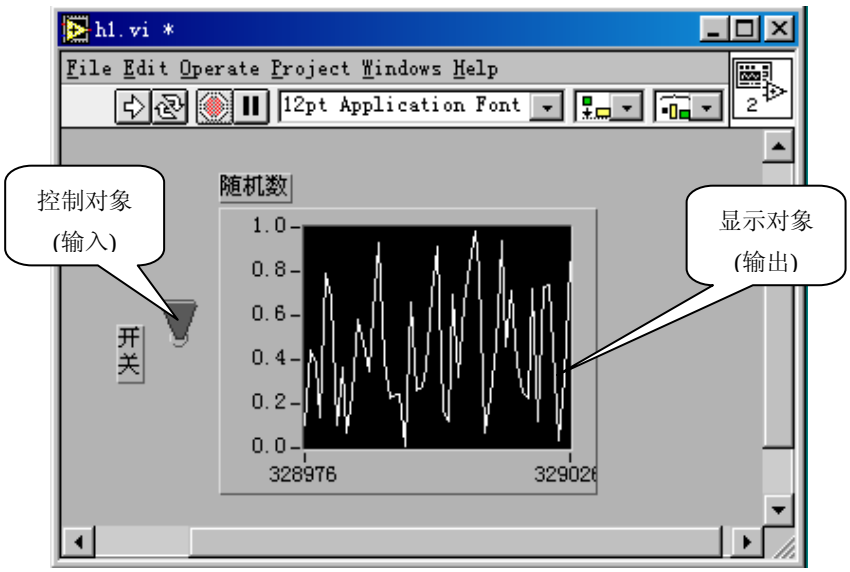
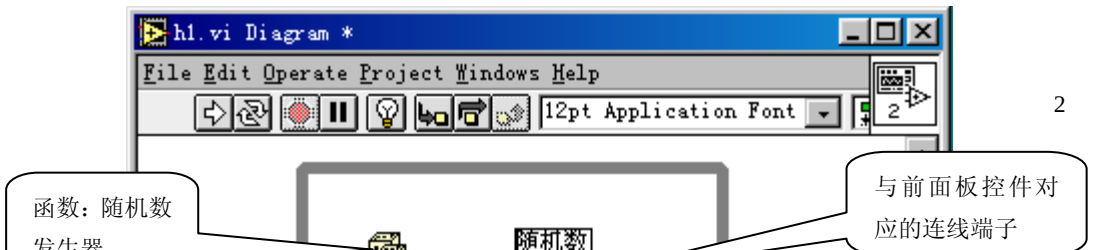


图 1 － 1　随机信号发生器的前

流程图

流程图提供 VI 的图形化源程序。在流程图中对 VI 编程，以控制和操纵定义在前面板上的输入和输出功能。流程图中包括前面板上的控件的连线端子，还有一些前面板上没有，但编程必须有的东西，例如函数、结构和连线等。图 1 － 2 是与图 1 － 1 对应的流程图。我们可以看到流程图中包括了前面板上的开关和随机数显示器的连线端子，还有一个随机数发生器的函数及程序的循环结构。随机数发生器通过连线将产生的随机信号送到显示控件，为了使它持续工作下去，设置了一个 While Loop 循环，由开关控制这一循环的结束。



如果将 VI 与标准仪器相比较，那么前面板上的东西就是仪器面板上的东西，而流程图上的东西相当于仪器箱内的东西。在许多情况下，使用 VI 可以仿真标准仪器，不仅在屏幕上出现一个惟妙惟肖的标准仪器面板，而且其功能也与标准仪器相差无几。

图标/连接器

VI 具有层次化和结构化的特征。一个 VI 可以作为子程序，这里称为子 VI（subVI），被其他 VI 调用。图标与连接器在这里相当于图形化的参数，详细情况稍后介绍。

1．2．2　LabVIEW 的操作模板

在 LabVIEW 的用户界面上，应特别注意它提供的操作模板，包括工具（Tools）模板、控制（Controls）模板和函数（Functions）模板。这些模板集中反映了该软件的功能与特征。下面我们来大致浏览一下。

工具模板（Tools Palette）

该模板提供了各种用于创建、修改和调试 VI 程序的工具。如果该模板没有出现，则可以在 Windows 菜单下选择 Show Tools Palette 命令以显示该模板。当从模板内选择了任一种工具后，鼠标箭头就会变成该工具相应的形状。当从 Windows 菜单下选择了 Show Help Window 功能后，把工具模板内选定的任一种工具光标放在流程图程序的子程序（Sub VI）或图标上，就会显示相应的帮助信息。

下述工具中注意 1 和 2 的区别，2 用于编程时，1 用于运行程序时。4 是一个特有的工具，它并不是一个简单的画线工具，而是一个符合 LabVIEW 语言规定的对象连接工具。

工具图标有如下几种：

	图标	名称	功　　能

1		Operate Value（操作值）	用于操作前面板的控制和显示。使用它向数字或字符串控制中键入值时，工具会变成标签工具
2		Position/Size/Select （选择）	用于选择、移动或改变对象的大小。当它用于改变对象的连框大小时，会变成相应形状。
3		Edit Text（编辑文本）	用于输入标签文本或者创建自由标签。当创建自由标签时它会变成相应形状。
4		Connect Wire（连线）	用于在流程图程序上连接对象。如果联机帮助的窗口被打开时，把该工具放在任一条连线上，就会显示相应的数据类型。
5		Object Shortcut Menu（对象菜单）	用鼠标左键可以弹出对象的弹出式菜单。
6		Scroll Windows（窗口漫游）	使用该工具就可以不需要使用滚动条而在窗口中漫游。
7		Set/Clear Breakpoint(断点设置／清除）	使用该工具在 VI 的流程图对象上设置断点。
8		Probe Data（数据探针）	可在框图程序内的数据流线上设置探针。通过控针窗口来观察该数据流线上的数据变化状况。
9		Get Color（颜色提取）	使用该工具来提取颜色用于编辑其他的对象。
10		Set Color（颜色设置）	用来给对象定义颜色。它也显示出对象的前景色和背景色。

下面的两个模板是多层的，其中每一个子模板下还包括多个对象。

控制模板（Control Palette）

注意：只有打开前面板时才能调用该模板

该模板用来给前面板设置各种所需的输出显示对象和输入控制对象。每个图标代表一类子模板。如果控制模板不显示，可以用 Windows 菜单的 Show Controls Palette 功能打开它，也可以在前面板的空白处，点击鼠标右键，以弹出控制模板。

控制模板如左图所示，它包括如下所示的一些子模板。子模板中包括的对象，我们在功能中用文字简要介绍。

	图标	子模板名称	功　　能

1		Numeric(数值量)	数值的控制和显示。包含数字式、指针式显示表盘及各种输入框。
2		Boolean(布尔量)	逻辑数值的控制和显示。包含各种布尔开关、按钮以及指示灯等。
3		String & Path(字符串和路径)	字符串和路径的控制和显示。
4		Array & Cluster（数组和簇）	数组和簇的控制和显示。
5		List & Table（列表和表格）	列表和表格的控制和显示
6		Graph(图形显示)	显示数据结果的趋势图和曲线图。
7		Ring & Enum（环与枚举）	环与枚举的控制和显示。
8		I/O(输入/输出功能)	输入/输出功能。于操作 OLE、ActiveX 等功能。
9		Refnum	参考数
10		Digilog Controls(数字控制)	数字控制
11		Classic Controls(经典控制)	经典控制，指以前版本软件的面板图标。
12		Activex	用于 ActiveX 等功能。
13		Decorations（装饰）	用于给前面板进行装饰的各种图形对象。
14		Select a Controls(控制选择)	调用存储在文件中的控制和显示的接口。
15		User Controls（用户控制）	用户自定义的控制和显示。

功能模板(Functions Palette)

注：只有打开了流程图程序窗口，才能出现功能模板。

功能模板是创建流程图程序的工具。该模板上的每一个顶层图标都表示一个子模板。若功能模板不出现，则可以用 Windows 菜单下的 Show Functions Palette 功能打开它，也可以在流程图程序窗口的空白处点击鼠标右键以弹出功能模板。

功能模板如右图所示，其子模块如下所示。（个别不常用的子模块未包含）

	图标	子模板名称	功　　能
1		Structure(结构)	包括程序控制结构命令，例如循环控制等，以及全局变量和局部变量。
2		Numeric（数值运算）	包括各种常用的数值运算，还包括数制转换、三角函数、对数、复数等运算，以及各种数值常数。
3		Boolean（布尔运算）	包括各种逻辑运算符以及布尔常数。
4		String(字符串运算)	包含各种字符串操作函数、数值与字符串之间的转换函数，以及字符(串)常数等。
5		Array（数组）	包括数组运算函数、数组转换函数，以及常数数组等。
6		Cluster（簇）	包括簇的处理函数，以及群常数等。这里的群相当于C语言中的结构。
7		Comparison（比较）	包括各种比较运算函数，如大于、小于、等于。
8		Time & Dialog(时间和对话框)	包括对话框窗口、时间和出错处理函数等。
9		File I/O（文件输入/输出）	包括处理文件输入/输出的程序和函数。
10		Data Acquisition（数据采集）	包括数据采集硬件的驱动，以及信号调理所需的各种功能模块。
11		Waveform（波形）	各种波形处理工具
12		Analyze（分析）	信号发生、时域及频域分析功能模块及数学工具 。
13		Instrument I/O（仪器输入/输出）	包括GPIB(488、488.2)、串行、VXI仪器控制的程序和函数，以及VISA的操作功能函数。
14		Motion & Vision（运动与景像）	
15		Mathematics（数学）	包括统计、曲线拟合、公式框节点等功能模块，以及数值微分、积分等数值计算工具模块。
16		Communication（通讯）	包括TCP、DDE、ActiveX和OLE等功能的处理模块。
17		Application Control（应用控制）	包括动态调用VI、标准可执行程序的功能函数。
18		Graphics & Sound（图形与声音）	包括3D、OpenGL、声音播放等功能模块。包括调用动态连接库和CIN节点等功能的处理模块。
19		Tutorial(示教课程)	包括LabVIEW示教程序。
20		Report Generation(文档生成)	
21		Advanced(高级功能)	
22		Select a VI（选	

		择子 VI）	
2 3		User Library（用户子 VI 库）	

1．4　LabVIEW 的初步操作

1．4．1　创建 VI 和调用子 VI

我们通过例子来说明如何创建一个 VI。

练习 1－1：

建立一个测量温度和容积的 VI，其中须调用一个仿真测量温度和容积的传感器子 VI。步骤如下：

1．选择 **File»New**，打开一个新的前面板窗口。

2．从 **Controls»Numeric** 中选择 **Tank** 放到前面板中。

3．在标签文本框中输入“容积”，然后在前面板中的其他任何位置单击一下。

4．把容器显示对象的显示范围设置为 0.0 到 1000.0。

a.　使用文本编辑工具（Text Edit Tool），双击容器坐标的 10.0 标度，使它高亮显示。

b.　在坐标中输入 1000，再在前面板中的其他任何地方单击一下。这时 0.0 到 1000.0 之间的增量将被自动显示。

5．在容器旁配数据显示。

将鼠标移到容器上，点右键，在出现的快速菜单中选 **Visible Iterms»Digital Display** 即可。

6．从 **Controls»Numeric** 中选择一个温度计，将它放到前面板中。设置其标签为“温度”,显示范围为 0 到 100，同时配数字显示。可得到如下的前面板图。

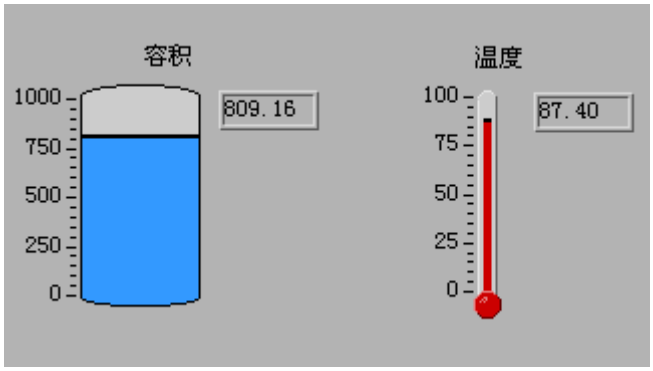


图 1－3　练习 1－1 的前面板图

7．Windows»Show Diagram 打开流程图窗口。从功能模板中选择对象，将它们放到流程图上组成下图（其中的标注是后加的）。

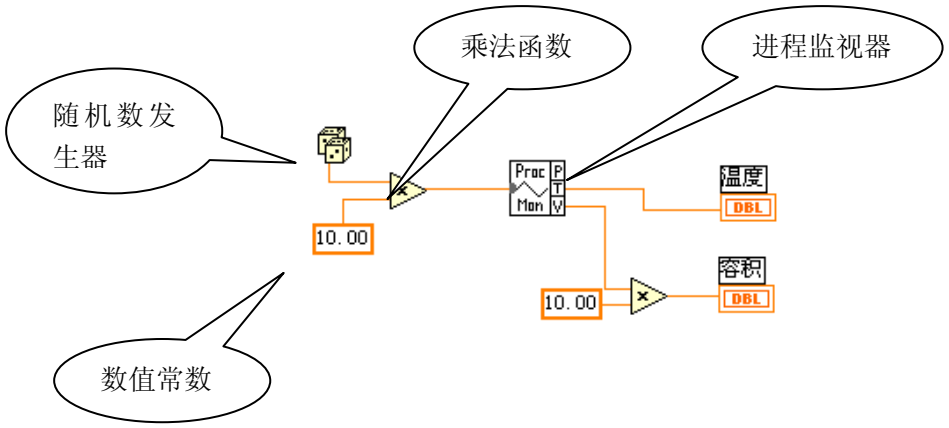


图 1 － 4　练习 1 － 1 的流程图

该流程图中新增的对象有两个乘法器、两个数值常数、一个随机数发生器、一个进程监视器，温度和容积对象是由前棉板的设置自动带出来的。

a. 乘法器和随机数发生器由 **Functions»Numeric** 中拖出，尽管数值常数也可以这样得到，但是建议使用 c 中的方法更好些。

b. 进程监视器（Process Monitor）不是一个函数，而是以子 VI 的方式提供的，它存放在 **LabVIEW\Activity** 目录中，调用它的方法是在 **Functions»Select a VI** 下打开 Process Monitor，然后在流程图上点击一下，就可以出现它的图标。

注意：LabVIEW 目录一般在 Program Files\National Instruments\目录下。

8．用连线工具  将各对象按规定连接。a 中的遗留问题创建数值常数对象的另一种方法是在连线时一起完成。具体方法是：用连线工具在某个功能函数或 VI 的连线端子上单击鼠标右键，再从弹出的菜单中选择 Create Constant，就可以创建一个**具有正确的数据格式**的数值常数对象。

9．选择 **File»Save**，把该 VI 保存为 LabVIEW\Activity 目录中的 Temp & Vol.vi。

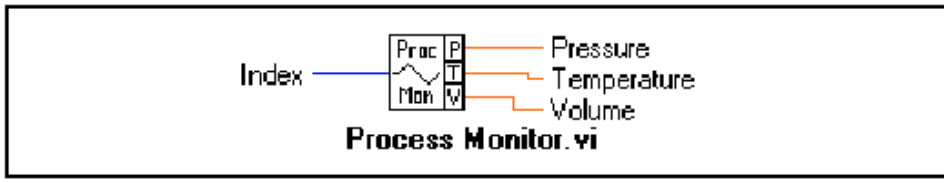
在前面板中，单击 **Run（运行）**按钮，运行该 VI。注意电压和温度的数值都显示在前面板中。

10．选择 **File»Close**，关闭该 VI 。

练习 1－1　结束

附注与说明：

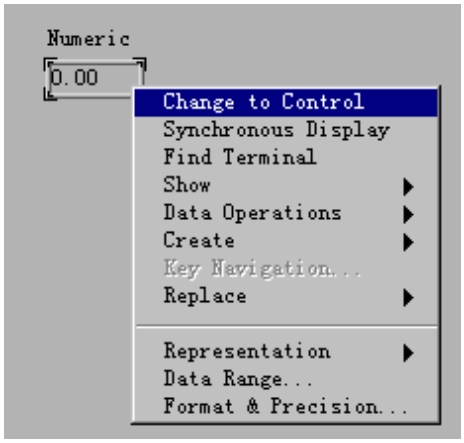
1．如果要查看某个功能函数或者 VI 的输入输出，需要从 **Help** 菜单中选择 **Show Help**，再把光标置于这个功能函数或者 VI 上。例如进程监视器 VI 的 Help 窗口显示如下：



2．显示对象（**Indicator**）、控制对象（**Control**）和数值常数对象

显示对象和控制对象都是前面板上的控件，前者有输入端子而无输出端子，后者正好相反，它们分别相当于普通编程语言中的输出参数和输入参数。数值常数对象可以看成是控制对象的一个特例。

在前面板中创建新的控制对象或显示对象时，LabVIEW 都会在流程图中创建对应的端子。端子的符号反映该对象的数据类型。例如，DBL 符号表示对象数据类型是双精度数；TF 符号表示布尔数；I16 符号表示 16 位整型数；ABC 符号表示对象数据类型是字符串。



一个对象应当是显示对象还是控制对象必须弄清楚，否则无法正

确连线。有时他们的图标是相似或相同的，可以根据需要明确规定它是显示对象还是控制对象。方法是将鼠标移到图标上，然后点右键，可出现快速菜单（例见右图）。如果菜单中的第一项是**Chang to Control**,说明这是一个显示对象，你可以根据需要，将其变为控制对象。如果菜单中的第一项是 **Chang to** Indicator，说明这是一个控制对象,你也可以根据需要,将其变为显示对象。

控制对象和显示对象都不能在流程图中删除,只能从前面板上删除。

3．关于连线

连线是程序设计中较为复杂的问题。流程图上的每一个对象都带有自己的连线端子，连线将构成对象之间的数据通道。因为这不是几何意义上的连线，因此并非任意两个端子间都可连线，连线类似于普通程序中的变量。数据单向流动，从源端口向一个或多个目的端口流动。不同的线型代表不同的数据类型。下面是一些常用数据类型所对应的线型和颜色：

类型	颜色	标量	一维数组	二维数组
整形数	兰色			
浮点数	橙色			
逻辑量	绿色			
字符串	粉色			
文件路径	青色			

当需要连接两个端点时，在第一个端点上点击连线工具（从工具模板栏调用），然后移动到另一个端点，再点击第二个端点。端点的先后次序不影响数据流动的方向。

当把连线工具放在端点上时，该端点区域将会闪烁，表示连线将会接通该端点。当把连线工具从一个端口接到另一个端口时，不需要按住鼠标键。当需要连线转弯时，点击一次鼠标键，即可以正交垂直方向地弯曲连线，按空格键可以改变转角的方向。

接线头是为了帮助正确连接端口的连线。当把连线工具放到端口上，接线头就会弹出。接线头还有一个黄色小标识框，显示该端口的名字。

线型为波折号的连线表示坏线。出现坏线的原因有很多，例如：连接了两个控制对象；源端子和终点端子的数据类型不匹配（例如一个是数字型，而另一个是布尔型）。可以通过使用定位工具点击坏线再按下 <Delete> 来删除它。选择 Edit»Remove Bad Wires 或者按下<Ctrl-B> 可以一次删除流程图中的所有坏线。当 VI 无法运行，或者显示 Signal has Loose Ends （信号丢失终端）的错误信息时，这是一个快捷的调试方法。

1．4．2　程序调试技术

1．找出语法错误

如果一个VI程序存在语法错误,则在面板工具条上的运行按钮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。点击它，则LabVIEW弹出错误清单窗口，点击其中任何一个所列出的错误，选用Find功能，则出错的对象或端口就会变成高亮。

2．设置执行程序高亮

在 LabVIEW 的工具条上有一个画着灯泡的按钮，这个按钮叫做“高亮执行”按钮上。点击这个按钮使它变成高亮形式，再点击运行按钮，VI 程序就以较慢的速度运行，没有被执行的代码灰色显示，执行后的代码高亮显示，并显示数据流线上的数据值。这样，你就可

以根据数据的流动状态跟踪程序的执行。

3．断点与单步执行

为了查找程序中的逻辑错误，有时希望流程图程序一个节点一个节点地执行。使用断点工具可以在程序的某一地点中止程序执行，用探针或者单步方式查看数据。使用断点工具时，点击你希望设置或者清除断点的地方。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 程序运行到断点被设置处，程序被暂停在将要执行的节点，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，指示它将被执行。你也可以点击暂停按钮，这样程序将连续执行直到下一个断点。

4．探针

可用探针工具来查看当流程图程序流经某一根连接线时的数据值。从 Tools 工具模板选择探针工具，再用鼠标左建点击你希望放置探针的连接线。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流程图窗口的上面。在流程图中使用选择工具或连线工具，在连线上点击鼠标右键，在连线的弹出式菜单中选择“探针”命令，同样可以为该连线加上一个探针。

1．4．3　子 VI 的建立

子 VI（SubVI）相当于普通编程语言中的子程序，也就是被其他的 VI 调用的 VI。可以将任何一个定义了图标和联接器的 VI 作为另一个 VI 的子程序。在流程图中打开 **Functions»Select a VI…**，就可以选择要调用的子 VI 。构造一个子 VI 主要的工作就是定义它的**图标和联接器**。

每个 VI 在前面板和流程图窗口的右上角都显示了一个默认的图标。启动图标编辑器的方法是，用鼠标右键单击面板窗口的右上角的默认图标，在弹出菜单中选择 **Edit Icon**。

下图显示了图标编辑器的窗口。可以用窗口左边的各种工具设计像素编辑区中的图标形状。编辑区右侧的一个方框中显示了一个实际大小的图标。图标编辑器的具体使用细节参阅有关资料。

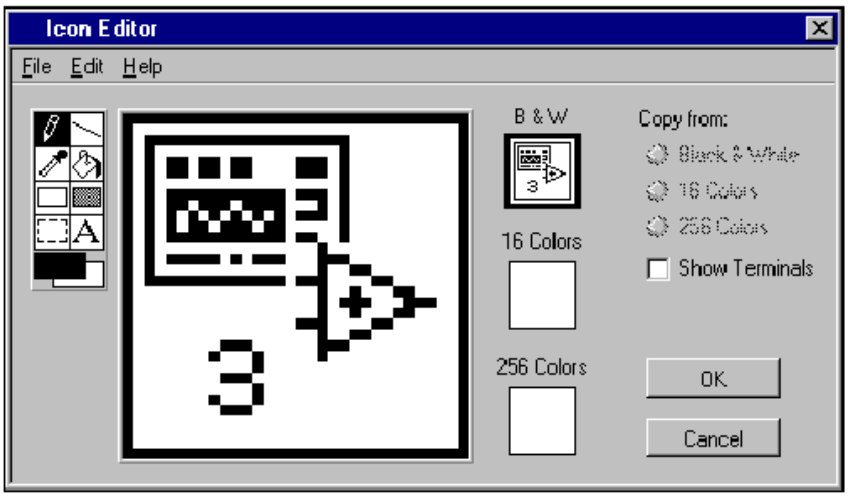


图 1 － 5　图标编辑器窗口

联接器是 VI 数据的输入输出接口。如果用面板控制对象或者显示对象从子 VI 中输出或者输入数据，那么这些对象都需要在联接器面板中有一个连线端子。您可以通过选择 VI 的端子数并为每个端子指定对应的前面板对象以定义联接器。

定义联接器的方法是，用鼠标右键单击面板窗口中的图标窗口，在快捷菜单中选择

Show Connector。

联接器图标会取代面板窗口右上角的图标。LabVIEW 自动选择的端子连接模式是控制对象的端子位于联接器窗口的左边，显示对象的端子位于联接器窗口右边。选择的端子数取决于前面板中控制对象和显示对象的个数。

联接器中的各个矩形表示各个端子所在的区域，可以用它们从 VI 中输入或者输出数据。如果必要，也可以选择另外一种端子连接模式。方法是在图标上单击鼠标右键单出快捷菜单，选择 **Show Connector**，再次弹出快捷菜单，选择 **Patterns**。下面我们通过一个练习说明具体操作。

练习 1－2　为 VI 创建图标和联接器

1．打开 **LabVIEW\Activity** 目录中的 **Temp & Vol.vi**，这是练习 1 － 1 做的程序。

2．在前面板中，用鼠标右键单击窗口右上角的图标，在快捷菜单中选择 **Edit Icon**…，也可以双击图标激活图标编辑器。注意只能在前面板中编辑图标和联接器。

3．删除默认图标。使用 **Select Tool**（矩形框），单击并拖动想要删除的部分，按下 **<Delete>**。也可以通过双击工具框中的阴影矩形删除图标。

4．用 **Pencil Tool** （铅笔工具）绘制一个温度计。 

5．用 **Text Tool**（文本工具）创建文本。得到图标将如下图所示。 

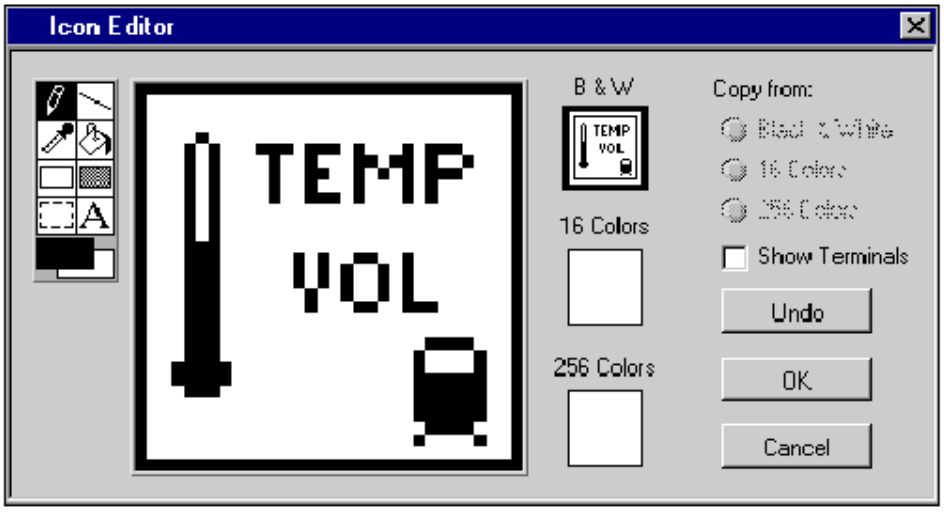
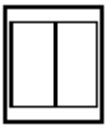
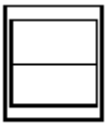


图 1 － 6　编辑后的图标编辑器窗口

6．单击 OK，关闭编辑器。新创建的图标就显示在屏幕右上角的图标窗口中。

7．用鼠标右键单击前面板中的图标窗口，在快捷菜单中选择 **Show Connector**，设置联接器端子连接模式。在默认情况下，LabVIEW 会根据前面板中的控制对象和显示对象的数目确定联接器的端子连接模式。因为前面板中有两个对象，所以联接器有两个端子，如左图所示。用鼠标右键单击联接器窗口，在快捷菜单中选择 **Rotate 90 Degrees**（旋转 90 度），注意联接器窗口的变化，如左图所示。





9．将端子连接到温度计和电压计：

a. 点击联接器上部端子。光标自动变成连线工具，同时端子变成黑色。

b. 单击温度显示对象。一个移动的虚线框把它包围起来，选中的端子的颜色变为与控制/显示对象的数据类型一致的颜色。

如果单击前面板中的任何空白区域以后，虚线消失，选中的端子变暗，这表示您已经成功地把显示对象和上部端子连接起来。如果端子是白色，则表示没有连接成功。

c. 重复步骤 a 和 b，把底部的端子和容积计连接起来。

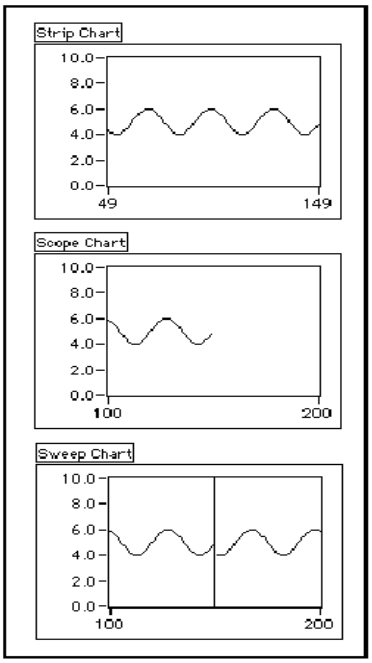
d. 用鼠标右键单击联接器，在快捷菜单中选择Show Icon….

１０．选择 File»Save ，保存该 VI。

这样这个 VI 就完成了，并也可以作为子 VI 被其他的 VI 调用。子 VI 的图标在主 VI 的流程图中代表它。VI 的联接器（含有两个端子）输出温度和电压。

练习 1－2　结束

１．５　图表（chart）入门



图表（chart）是一种周期性更新数据的数字式图形显示对象。可以在 Controls»Graph 模板中找到两种图表：波形图和强度图（后者不常用）。也可以自定义图表的格式，以满足自己的显示需要或者让它显示更多的数据。图表具有的特性有：滚动条、图例、模板、数值显示、时间坐标显示。

右图显示了快速菜单中 Advanced»Update Mode 子菜单中提供的三种图表显示类型——Strip chart(条状图)，Scope chart（示波器图）和 Sweep chart（扫描图）。默认模式是条状图。

练习 1-3　　使用三种图表模式

目的：查看 VI 分别在三种模式下执行时图表的显示。

1．建立前面板及流程图如下

该程序中利用一个循环产生连续的 sin(i)函数值，并及时地在 chart 图表上显示出来，现在前面板上的 chart 是一个 strip，这是一个坐标式显示器，与纸带式图表记录器相似。每接受一个新数据，新数据就将显示在右侧，而原有数据移动到左侧

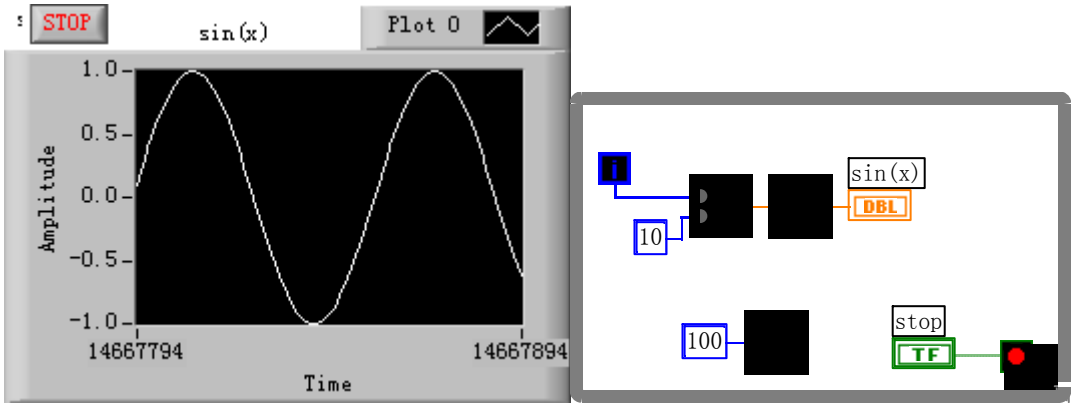


图 1 － 7　图表的例子

2. 用鼠标选中 chart,点击右键，可在快速菜单中选择 Advanced»Update Mode 子菜单。可以选择更换其他两种更新模式。

示波器模式是一个返回式的显示器，与示波器类似。每接受一个新数据时，它就把新数据绘制在原有数据的右侧。当数据曲线到达显示区的右边缘时，VI 会删除全部图形，从左边缘重新开始绘制曲线。示波器模式显然要快于条状图模式，因为它不会因为滚动产生溢出。

扫描模式更接近于示波器模式，但是当数据曲线到达显示区的右边时，不会变成空白，而是会出现一个移动的垂线，标记新数据的开始，并当 VI 添加新数据时穿过整个显示区。

练习 1-3　结束。

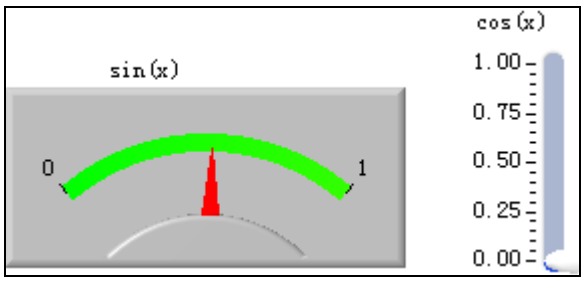
重叠式和堆栈式图区

LabVIEW 可以用同一个垂直坐标在一个图表中显示多个图区，这种图区被称为重叠式图区，也可使用多个垂直坐标，这时这种图形被称为堆栈式图区。请参考 Examples\General\Graphs\charts.llb 中的 charts.vi 示例。

补充练习：

通过下面的要求练习 LabVIEW 的操作。

画出如右下所示的流程图，配以适当的前面板。注意连线时快捷菜单和 Create 功能的使用，和在前面板上替换不同的 Indicator 的方法。



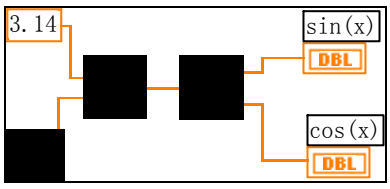


图 1 － 8　补充练习的面板及框图

第二章　程序结构

2．1　循环结构

2．1．1　While 循环

While 循环可以反复执行循环体的程序，直至到达某个边界条件。它类似于普通编程语言中的 Do 循环和 Repeat-Until 循环。While 循环的框图是一个大小可变的方框，用于执行框中的程序，直到条件端子接收到的布尔值为 FALSE。

- 该循环有如下特点：
- 计数从 0 开始（i=0）。
- 先执行循环体，而后 i+1，如果循环只执行一次，那么循环输出值 i=0。
- 循环至少要运行一次。

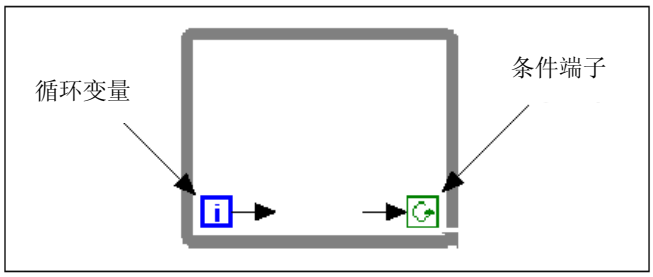


图 2 － 1　While 循环示意图

练习　2－1　使用 While 循环和图表

目的：用 While 循环和图表获得数据，并实时显示。

创建一个可以产生并在图表中显示随机数的 VI。 前面板有一个控制旋钮可在 0 到 10 秒之间调节循环时间，还有一个开关可以中止 VI 的运行。学习怎样改变开关的动作属性，以便不用每次运行 VI 时都要打开开关。操作步骤如下：

前面板

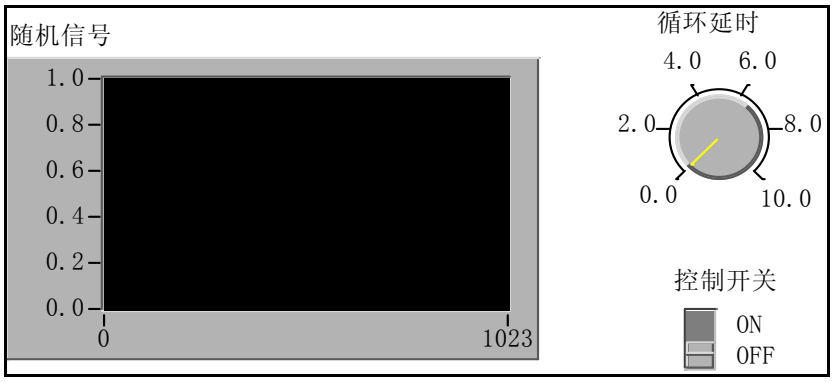


图 2 － 2　练习 2 － 1 的前面板

1．　选择 File»New，打开一个新的前面板。

2．　选择 Controls»Boolean，在前面板中放置一个开关。

设置开关的标签为控制开关。

3. 使用标签工具创建 ON 和 OFF 的标签，放置于开关旁。

4. 选中 **Controls»Graph**，在前面板中放置一个波形图（是 chart，而不是 graph）。设置它的标签为随机信号。这个图表用于实时显示随机数。

5. 把图表的纵坐标改为 0.0 到 1.0。方法是用标签工具把最大值从 10.0 改为 1.0。

6. 选择 **Controls»Numeric**，在前面板中放置一个旋钮。设置旋钮的标签为循环延时。这个旋钮用于控制 While 循环的循环时间。

流程图

7. 开流程图，按照下图创建流程图。

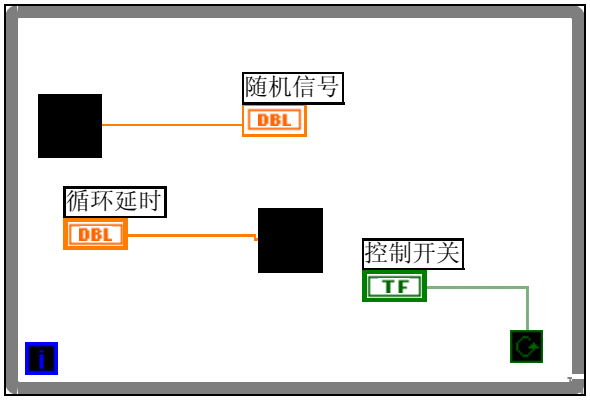


图 2 － 2　练习 2 － 1 的流程图

a. 从 **Functions»Structures** 中选择 While 循环，把它放置在流程图中。将其拖至适当大小，将相关对象移到循环圈内。

b. 从 **Functions» Numeric** 中选择随机数（0-1）功能函数放到循环内。

c. 在循环中设置 **Wait Until Next ms Multiple** 函数（**Functions»Time & Dialog**），该函数的时间单位是毫秒，按目前面板旋钮的标度，可将每次执行时间延迟 0 到 10 毫秒。

d. 照上面所示的流程图连线，把随机数功能函数和随机信号图表输入端子连接起来，并把启动开关和 While 循环的条件端子连接。

8. 返回前面板，调用操作工具后单击垂直开关将它打开。

9. 把该 VI 保存为 **LabVIEW\Activity** 目录中的 Random Signal.vi。

10. 执行该 VI。While 循环的执行次数是不确定的，只要设置的条件为真，循环程序就会持续运行。在这个例子中，只要开关打开（TRUE），框图程序就会一直产生随机数，并将其在图表中显示。

11. 单击垂直开关，中止该 VI。关闭开关这个动作会给循环条件端子发送一个 FALSE 值，从而中止循环。

12. 用鼠标右键单击图表，选择 **Data**

Operations»Clear Chart，清除显示缓存，重新设置图表。

练习　2－1 结束

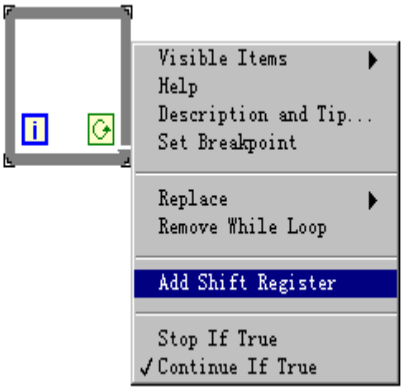
附注与说明

布尔开关的机械动作：

布尔开关有 6 种机械动作属性可供选择。在前面板上用鼠标右键单击开关，在快捷菜单中选择 Mechanical Action 就可以看到这些可选的动作。LabVIEW 还提供了一个范例示范这些动作，它是位于 Examples\General\Controls\booleans.llb 的 Mechanical Action of Booleans.vi。

2．1．2　移位寄存器（**Shift Register**）

移位寄存器可以将数据从一个循环周期传递到另外一个周期。在程序设计中，经常要用到它．创建一个移位寄存器的方法是，用鼠标右键单击循环的左边或者右边，在快捷菜单中选择 **Add Shift Register**。如右图所示。



移位寄存器在流程图上用在循环边框上相应的一对端子来表示。右边的端子中存储了一个周期完成后的数据，这些数据在这个周期完成之后将被转移到左边的端子，赋给下一个周期。移位寄存器可以转移各种类型的数据－－数值、布尔数、数组、字符串等等。它会自动适应与它连接的第一个对象的数据类型。下图表示了它的工作过程.

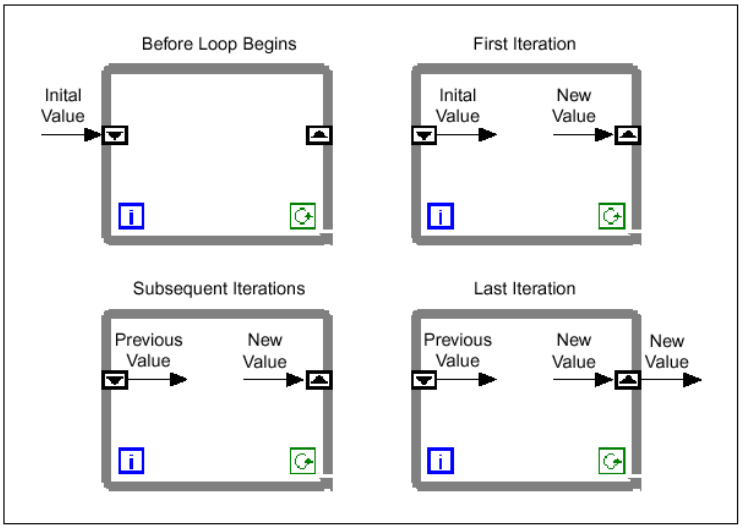


图 2－3　**Shift Register** 的工作过程

可以令移位寄存器记忆前面的多个周期的数值。这个功能对

于计算数据均值非常有用。还可以创建其他的端子访问先前的周期的数据，方法是用鼠标右键单击左边或者右边的端子，在快捷菜单中选择 Add Element。例如，如果某个移位寄存器左边的端口含有三个元素，那么就可以访问前三个周期的数据。

练习 2－2 使用移位寄存器

目的：创建一个可以在图表中显示运行平均数的VI。

前面板

1．打开一个新的前面板，按照下图所示创建对象。

2．把波形图表的坐标范围改为 0.0 到 2.0。

3．在添加竖直坐标之后，用鼠标右键单击它，在快捷菜单中选择 Mechanical Action»Latch When Pressed，再选择 Operate»Make Current Values Default，把 ON 状态设置为默认状态。

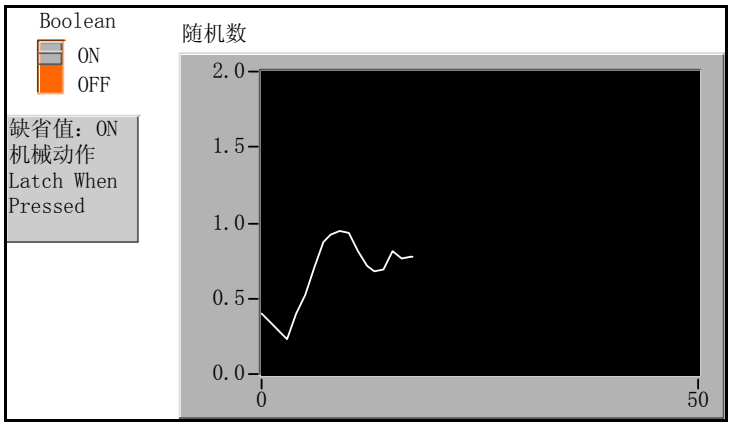


图 2－4 练习 2－2 的前面板

流程图

4．按下图创建流程图。

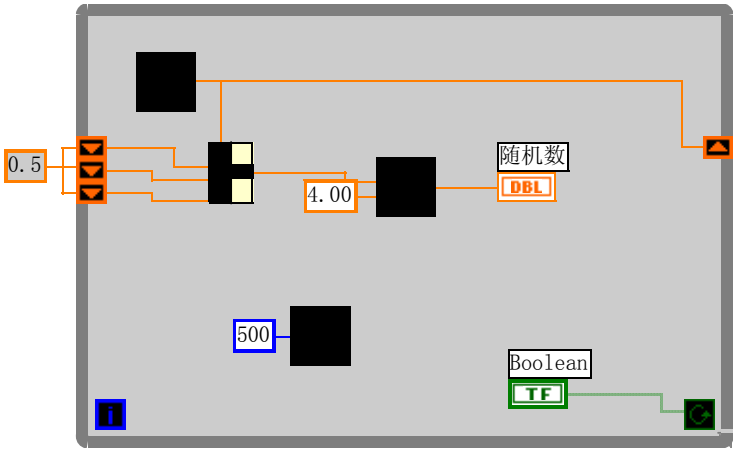


图 2－4 练习 2－2 的流程图

5．在流程图中添加 While 循环(**Functions»Structures**)，创建移位寄存器。

a. 用鼠标右键单击 While 循环的左边或者右边，在快捷菜单中选择 **Add Shift Register**。

b. 用鼠标右键单击寄存器的左端子，在快捷菜单中选择 **Add Element**，添加一个寄存器。用同样的方法创建第三个元素。

6．Random Number (0–1)函数（**Functions»Numeric**）——产生 0 到 1 之间的某个随机数。

7．Compound Arithmetic 函数（**Functions»Numeric**）——在本练习中，它将返回两个周期产生的随机数的和。如果要加入其他的输入，只需用右键单击某个输入，从快捷菜单中选择 Add Input。

8．除法函数（**Functions»Numeric**）——在本练习中，它用于返回最近四个随机数的平均值。

9．数值常数（**Functions»Numeric**）——在 While 循环的每个周期，Random Number (0–1) 函数将产生一个随机数。VI 就将把这个数加入到存储在寄存器中的最近三个数值中。Random Number (0–1)再将结果除以 4，就能得到这些数的平均值（当前数加上以前的三个数）。然后再将这个平均值显示在波形图中。

10． Wait Until Next ms Multiple 函数 （**Functions»Time & Dialog**）——它将确保循环的每个周期不会比毫秒输入快。在本练习中，毫秒输入的值是 500 毫秒。如果用鼠标右键单击图标，从快捷菜单中选择 **Visible»Label**，就可以看到 Wait Until Next ms Multiple 的标签。

11． 用鼠标右键单击 Wait Until Next ms Multiple 功能函数的输入端子，在快捷菜单中选择 **Create Constant**。出现一个数值常数，并自动与功能函数连接。

12． 将 **Constant 设置为** 500。这样连接到函数的数值常数设置了 500 毫秒的等待时间。因此循环每半秒执行一次。注意，VI 用一个随机数作为移位寄存器的初始值。如果没有设置移位寄存器端子的初始值，它就含有一个默认的数值，或者上次运行结束时的数值，因此开始得到的平均数没有任何意义。

13． 执行该 VI，观察过程。

14． 把该 VI 保存为 LabVIEW\Activity 目录下的 Random Average.vi。

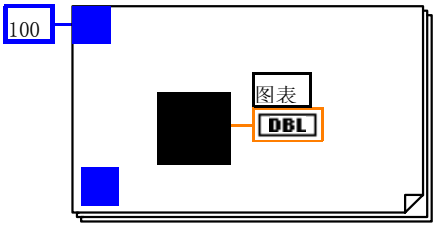
练习 2－2 结束。

附注：移位寄存器的初值：

上面的练习中对移位寄存器设置了初值0.5。如果不设这个初值，默认的初值是 0 。在这个例子中，一开始的计算结果是不对的，只有到循环完 3 次后移位寄存器中的过去值才填满，即第 4 次循环执行后可以得到正确的结果。

2．1．3 For 循环

For 循环用于将某段程序执行指定次数。和 While 循环一样，它不会立刻出现在流程图中，而是出现一个小的图标，而后您可以修改它的大小和位置。具体的方法是，先单击所有端子的左上方，然后按下鼠标，拖曳出一个包含所有端子的矩形。释放鼠标时就创建了一个指定大小和位置的 For 循环。



For 循环将把它的框图中的程序执行指定的次数，For 循环具有下面这两个端子：

N:　计数端子（输入端子）——用于指定循环执行的次数。

I:　周期端子（输出端子）——含有循环已经执行的次数。

上图显示了一可以产生 100 个随机数并将数据显示在一个图表上的 For 循环。在该例中，i 的初值是 0，终值是 99。

练习 2-3　使用 For 循环

目的：用 For 循环和移位寄存器计算一组随机数的最大值。

1．打开一个新的前面板，按照下图创建对象。

a.　将一个数字显示对象放在前面板，设置它的标签为“最大值”。

b.　将一个波形图表放在前面板，设置它的标签为“随机数”。将图表的纵坐标范围改为 0.0 到 1.0。

c.　在图表的快捷菜单中选择 **Visible Items»Scrollbar** 和 **Digital Display**，并隐藏 **Plot Legend**。

d.　用移位工具修改滚动栏的大小。

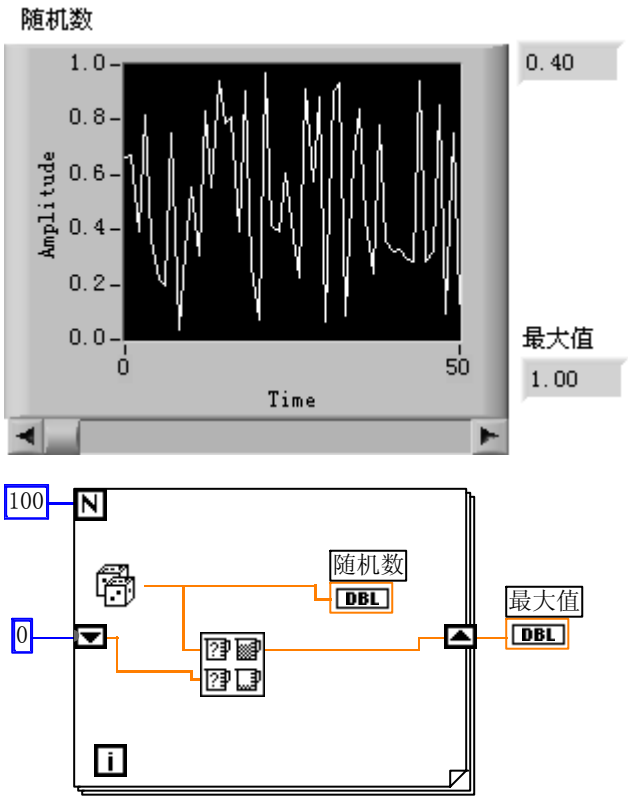


图 2－5　练习 2－3 的面板和流程图

2．按照上图画流程图。

3．在流程图中放置一个 For 循环（**Functions»Structures**）。

4．在 For 循环的边框处单击鼠标右键，在快捷菜单中选择 **Add Shift Register**。

５．将下列对象添加到流程图。

a　Random Number (0－1)函数（**Functions»Numeric**）——产生 0 到 1 之间的某个随机数。

b　数值常数（**Functions»Numeric**）——在这个练习中需要将移位寄存器的初始值设成 0。

c　Max&Min 函数（**Functions»Comparison**）——输入两个数值，再将它们的最大值输出到右上角，最小值输出到右下角。这里只需要最大值，只用连接最大值输出。

d　数值常数（**Functions»Numeric**）——For 循环需要知道需要执行的次数。本练习中是 100 次。

６．按照上图连接各个端子。

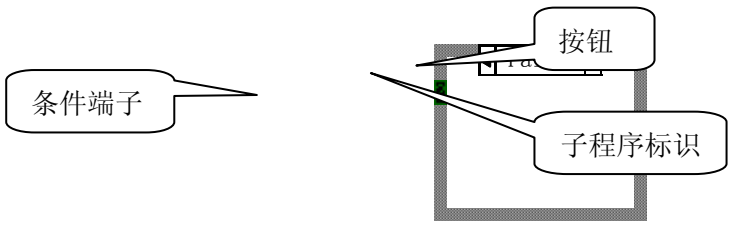
７．运行该 VI。

８．将该 VI 保存为 **LabVIEW\Activity** 目录下的 Calculate Max.vi。

练习 2-3　结束。

２．２　分支结构：Case

Case 结构含有两个或者更多的子程序（Case）,执行那一个取决于与选择端子或者选择对象的外部接口相连接的某个整数、布尔数、字符串或者标识的值。必须选择一个默认的 Case 以处理超出范围的数值，或者直接列出所有可能的输入数值。Case 结构见下图，各个子程序占有各自的流程框，在其上沿中央有相应的子程序标识：Ture、False 或１、２、３…。按钮用来改变当前显示的子程序（各子程序是重叠放在屏幕同一位置上的）。



练习 2－4　使用 Case 结构

目的：创建一个VI以检查一个数值是否为正数。如果它是正的，VI就计算它的平方根，反之则显示出错。

前面板

１．　打开一个新的前面板，并按照下图所示创建对象。控制对象用于输入数值，显示对象用于显示该数值的平方根。

流程图

2．照下图创建流程图。

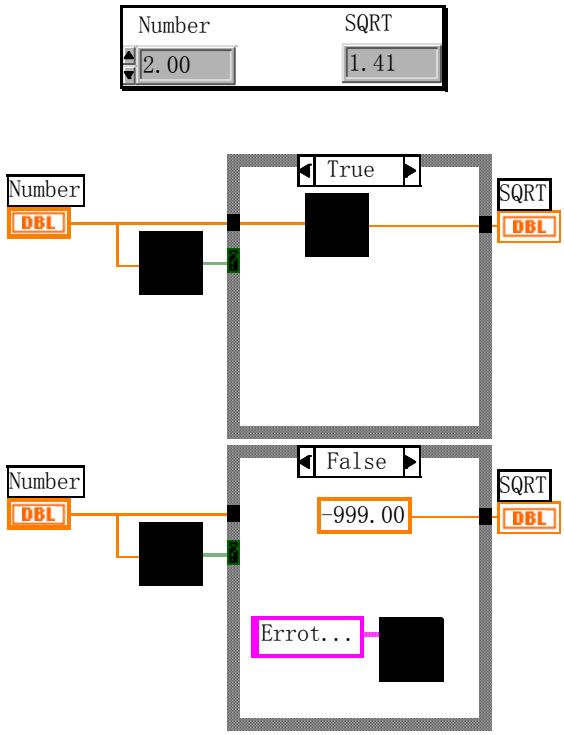


图 2 － 6　练习 2 － 4 的面板和流程图

3.　从 **Functions»Structures** 中选择一个 Case 结构，并放置在在流程图中。Case 结构是一个可以改变大小的方框。先来做 Ture 的情况，照流程图上半部分构造。

a　Greater Or Equal To 0? 函数（**Functions»Comparison**）——如果输入数值大于或者等于 0 就会返回一个 TRUE 值。

b　Square Root 函数（**Functions»Numeric**）——返回输入数值的平方根。

c　连好线

d　点击 Case 框的选择按钮，转入 False 情况编程

e　数值常数（Functions»Numeric）——这里用于显示错误的代数值-999.00。

f　One Button Dialog 函数（Functions»Time & Dialog）——在这里它用于显示一个对话框，内容是 Error...。

g　字符串常数 (Functions»String)——用 **Edit Text Tools** 在对话框中输入字符串。

h　该 VI 在 TRUE 或者 FALSE 情况下都会执行。如果输入的数值大于等于 0，VI 会执行 TRUE Case，返回该数的平方根，否则将会输出－999.00，并显示一个对话框，内容为 Error...。

4.　返回前面板，运行该 VI。修改标签为 Number 的数字式控制对象的数值，分别尝试一个正数和负数。注意，当把数字式控制对象的值改为负数时，LabVIEW 会显示 Case 结构的 FALSE

Case 中设置的出错信息。

5. 保存该 VI 到 **LabVIEW\Activity** 目录中的 Square Root.vi。

VI 的算法

本练习中的流程图功能相当于代码式编程语言中的下列伪代码：

```
if (Number >= 0) then
    Square Root Value = SQRT(Number)
else
    Square Root Value = -999.00
    Display Message "Error.. "
end if
```

练习 2-4 结束。

2．3 顺序结构和公式节点

2．3．1 顺序结构（Sequence Structure）

在代码式的传统编程语言中，默认的情况是，程序语句按照排列顺序执行，但 LabVIEW 中不同，它是一种图形化的数据流式编程语言。在图 2 － 7 左图中，假设有 A、B、C、D 4 个节点，其数据流向如右图所示。按照数据流式语言的约定，任何一个节点只有在所有

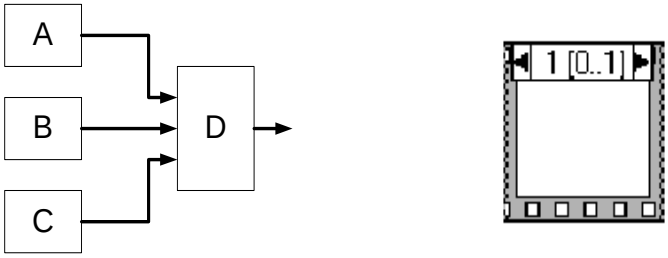


图 2 － 7 顺序结构的说明

的输入数据有效时才会执行，所以图中，当且仅当 A、B、C 3 个节点执行完，使得 D 节点的 3 个输入数据都到达 D 节点后，D 节点才执行。但是你要注意，这里并没有规定 A、B、C 3 个节点的执行顺序。在 LabVIEW 中这种情况下，A、B、C 的执行顺序是不确定的，如果你需要对它们规定一个确定的顺序，那就需要使用本节介绍的“顺序结构”。

图 2 － 7 中的右边是顺序结构的图标，它看上去像是电影胶片。它可以按一定顺序执行多个子程序。首先执行 0 帧中的程序，然后执行 1 帧中的程序，逐个执行下去。与 Case 结构类似，这多帧程序在流程图中占有同一个位置。

练习 2-5 使用顺序结构

目的：创建一个 VI，计算生成等于某个给定值的随机数所需要的时间。

前面板

1. 打开一个新的前面板，并按照下图所示创建对象。

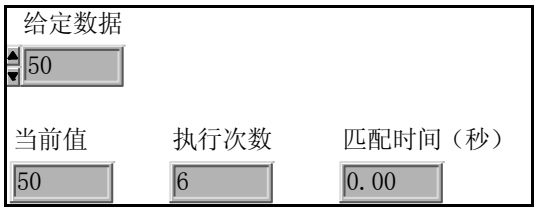
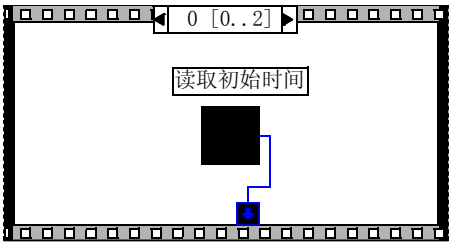
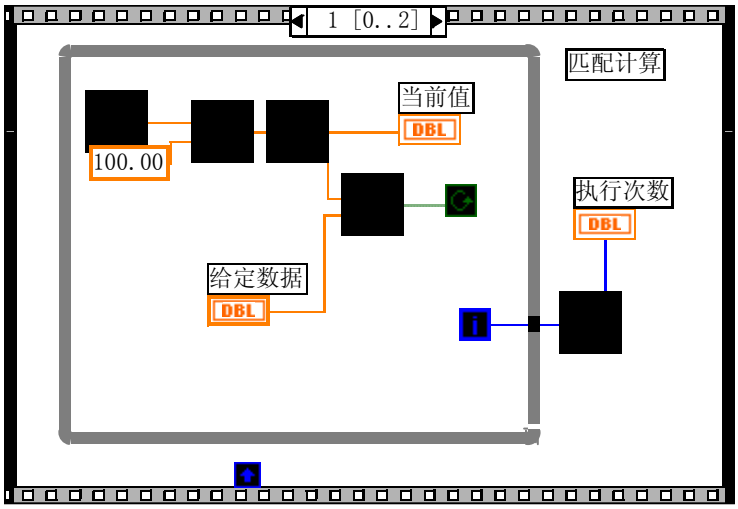


图 2 － 8　练习 2 － 5 的前面板

我们约定数据是 0 到 100 范围的整数。当前值用于显示当前产生的随机数。“执行次数”用于显示达到指定值循环执行的次数。匹配时间用来显示达到指定值所用的时间。

流程图





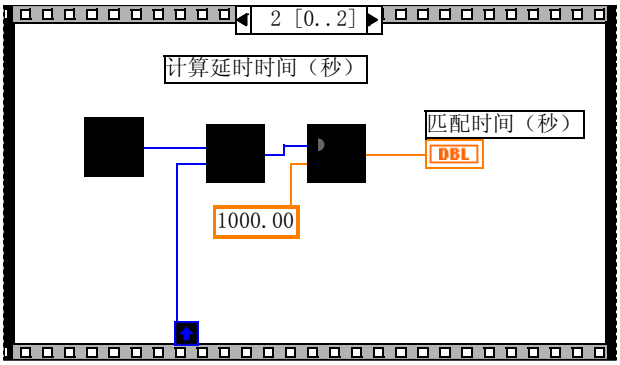


图 2 － 8　练习 2 － 5 的流程图（共 3 帧）

1．　在流程图中放置顺序结构(Functions»Structures)。

2． 用鼠标右键单击帧的边框，在快捷菜单中选择 **Add Frame After**，创建一个新帧。重复这个步骤，再创建一个帧。共 3 桢。

3． 选中第 0 桢，设置读取初始时间（子）程序

4． 第 0 帧的下边框上含有一个小方框，其中有一个箭头。这个方框叫做顺序局部变量，可以在同一个顺序结构中的各个帧之间传递数据。用鼠标右键单击第 0 帧的底部边框，选择 **Add Sequence Local**，创建顺序局部变量。顺序局部变量显示为一个空的方块。当您将某个功能函数与顺序局部变量相连时，方块中的箭头就会自动显示。

5．Tick Count (ms) 函数（**Functions»Time & Dialog**）——返回启动到现在的时间（以毫秒为单位）。在这里例子里需要使用两个这个函数。另一个在第 2 帧中。

6． 按图连好线。转入第 1 帧。该帧是匹配计算，内含一个循环结构。该图中使用的新函数有：

 Round to Nearest 函数（**Functions»Numeric**）——在该例中，它用于取 0 到 100 之间的随机数到距离最近的整数。

 Not Equal?函数（**Functions» Comparison**）——在该例中，它将随机数和前面板中设置的数相比较，如果两者不相等会返回 TRUE 值，否则返回 FALSE。

 Increment 函数（**Functions»Numeric**）——在该例中，它将 While 循环的计数器加 1。

7． 按图连好线。转入第 2 帧

在第 0 帧中，Tick Count （ms）功能函数将以毫秒为单位表示当前时间。这个数值被连到顺序局部变量，这样它就可以被后续的帧使用。在第 1 帧中，只要函数返回的值与指定值不等，VI 就会持续执行 While 循环。在第 2 帧中，Tick Count （ms）功能函数以毫秒为单位返回新的时间。VI 从中减去原来的时间（由第 0 帧通过顺序局部变量提供）就可以计算出花费的时间。

8． 返回前面板，在 Number to Match 控制对象中输入一个数值，执行该 VI。

9． 把该 VI 保存为 **LabVIEW\Activity** 目录下 Time to Match.vi。

练习 2-5 结束。

附注与说明：设置数据范围

在设定一个数据对象时，可以设置对输入数据的限制，利用快捷键选择 Data Range… 选项，将会出现如下对话框:

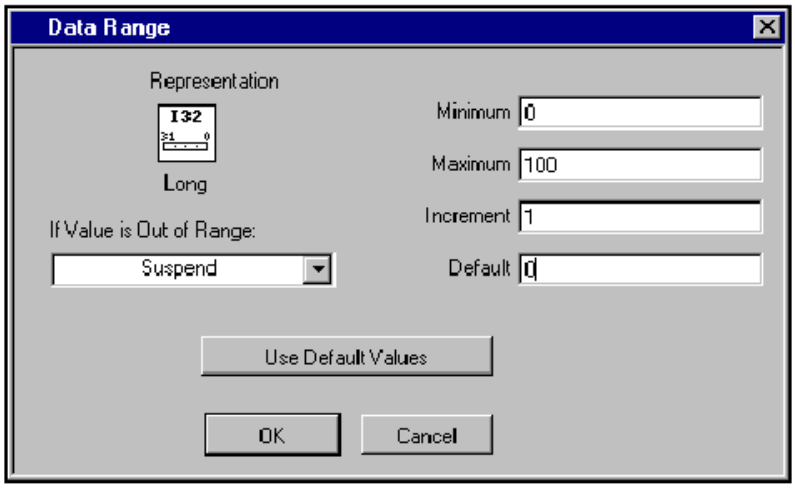


图 2 — 9　设置数据范围

它可以防止用户创建的控制对象或显示对象的值超出某个预设的范围。您可以选择忽略这个值，将它强制修改到范围以内，或暂停程序的执行。在程序执行时，如果发生溢出错误，溢出错误符号将显示在工具栏中的执行按钮的位置。而且，一个立体的黑框将把发生溢出的控制对象包围起来。

2．3．2　公式节点（Formula Node）

公式节点是一个大小可变的方框，可以利用它直接在流程图中输入公式。从 Functions»Structures 中选择公式节点就可以把它放到流程图中。当某个等式有很多变量或者非常复杂时，这个功能就非常有用。例如等式：$y = x^2 + x + 1$ 使用公式节点可以表示为：

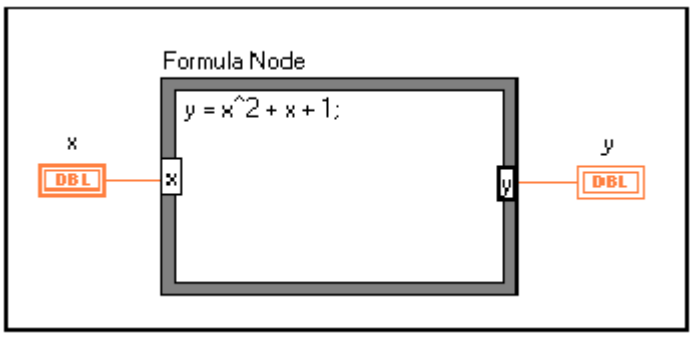


图 2 — 1 0　公式节点示意图

利用公式节点可以直接输入一个或者多个复杂的公式，而不用创建流程图的很多子程序。使用文本编辑工具来输入公式。创建公式节点的输入和输出端子的方法是，用鼠标右键单击第 0 帧的底部边框，选择 Add Input (Add Output)。再在节点框中输入变量名称。变量名对大小写敏感。然后就可以在框中输入公式。每个公式语句都必须以分号（;）结尾。

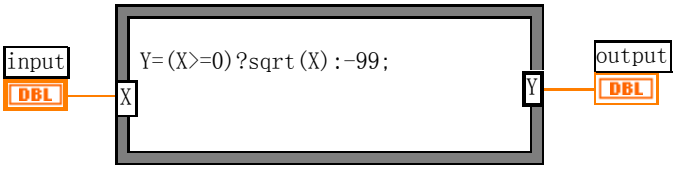
公式节点的帮助窗口中列出了可供公式节点使用的操作符、函数和语法规定。一般说来，它与 C 语言非常相似，大体上一个用 C 写的独立的程序块都可能用到公式节点中。但是仍然建议不要在一个公式节点中写过于复杂的代码程序。

下面这个例子显示了如何在一个公式节点中执行不同条件时的数据发送。

请阅读下面这段程序代码，如果 X 为正数，它将算出 X 的平方根并把该值赋给 Y，如果 X 为负数，程序就给 Y 赋值-99。

```
if (x >= 0) then
    y = sqrt(x)
else
    y = -99
end if
```

可以用公式节点取代上面这段代码，如下图所示：



注意：公式节点中变量字母 X,Y 大、小写是有区别的，开方的函数 sqrt(X)中函数名称是小写。

练习 2-6　使用公式节点

目的：创建一个VI，它用公式节点计算下列等式：

$y1 = x^3 - x^2 + 5$

$y2 = m* x + b$

x 的范围是从 0 到 10。可以对这两个公式使用同一个公式节点，并在同一个图表中显示结果。

前面板

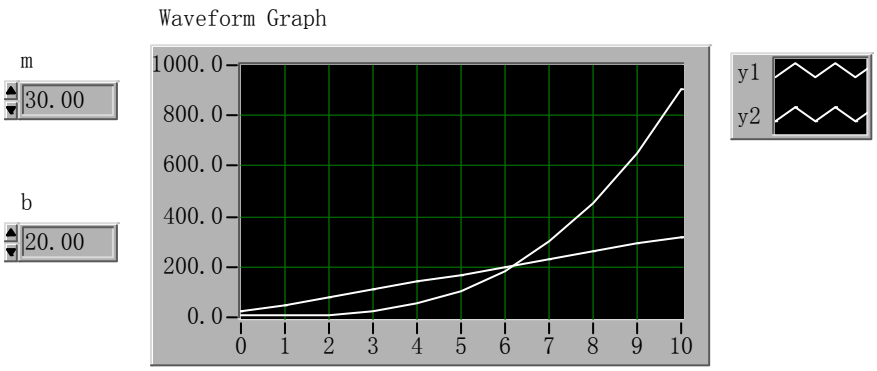


图 2 － 1 1　练习 2 － 6 的面板

1.　打开一个新的前面板，按照上图(该图中包含运行结果)创建前面板中的对象。波形图显示对象用于显示等式的图形。该

VI 使用两个数字式控制对象来输入 m 和 b 的值。

流程图

2. 按照下图创建流程图。

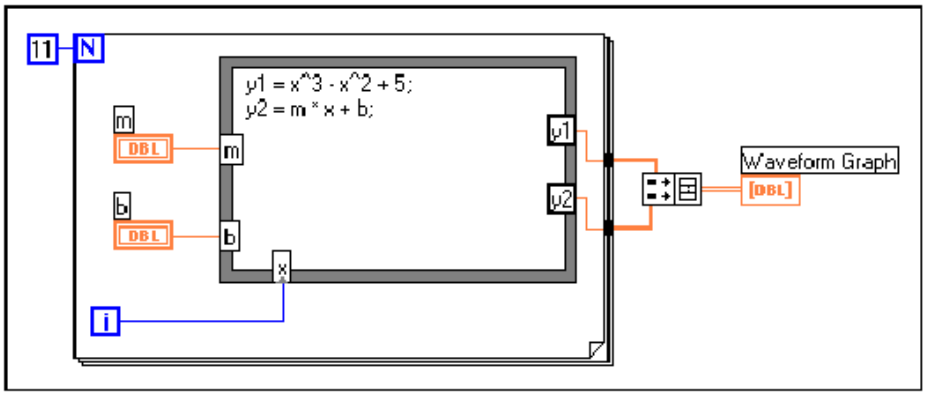


图 2 － 1 2　练习 2 － 6 的流程图

在创建某个输入或者输出端子时，必须给它指定一个变量名。这个变量名必须与公式节点中使用的变量名完全相符。

公式节点中，在边框上单击鼠标右键，在快捷菜单中选择 **Add Input**，可以创建三个输入端子。在快捷菜单中选择 **Add Output**，创建输出端子。

x 的范围是从 0 到 10（包括 10），就必须连接 11 到计数端子。

Build Array （**Functions»Array**）——在这个例子中，它用于将两个数据构成数组形式提供给一个多曲线的图形中。通过用变形工具拖拉边角就可以创建两个输入端子。

3. 返回前面板，尝试给 m 和 b 赋以不同的值再执行该 VI。

4. 把该 VI 保存为 **LabVIEW/Activity** 目录下的 Equations.vi。

练习 2-6 结束。

第三章 数据类型：数组、簇和波形

3．1 概述

数组是同类型元素的集合。一个数组可以是一维或者多维，如果必要，每维最多可有 $2^{31}-1$ 个元素。可以通过数组索引访问其中的每个元素。索引的范围是 0 到 n － 1，其中 n 是数组中元素的个数。图 3 － 1 所显示的是由数值构成的一维数组。注意第一个元素的索引号为 0，第二个是 1，依此类推。数组的元素可以是数据、字符串等，但所有元素的数据类型必须一致。

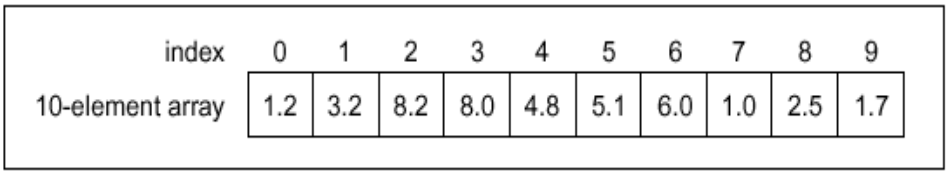


图 3 － 1 数组示意图

簇（Cluster）是另一种数据类型，它的元素可以是不同类型的数据。它类似于 C 语言中的 stuct。使用簇可以把分布在流程图中各个位置的数据元素组合起来，这样可以减少连线的拥挤程度。减少子 VI 的连接端子的数量。

波形（Waveform）可以理解为一种簇的变形，它不能算是一种有普遍意义的数据类型，但非常实用。

3．2 数组的创建及自动索引

3．2．1 创建数组

一般说来，创建一个数组有两件事要做，首先要建一个数组的“壳”（shell），然后在这个壳中置入数组元素（数或字符串等）。

如果需要用一个数组作为程序的数据源，可以选择 **Functions»Array»Array Constant**，将它放置在流程图中。然后再在数组框中放置数值常量、布尔数还是字符串常量。下图显示了在数组框放入字符串常量数组的例子。左边是一个数组壳，中间的图上已经置入了字符串元素，右边的图反映了数组的第 0 个元素为：”ABC”，后两个元素均为空。

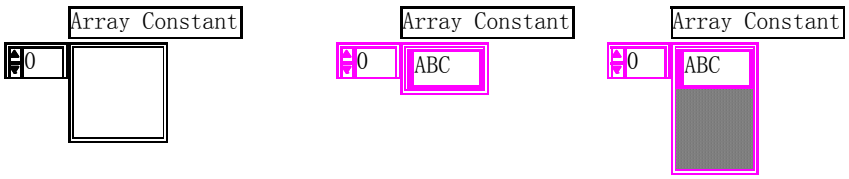


图 3 － 1 数组的创建

在前面板中创建数组的方法是，从 Controls 模板中选择 Array & Cluster，把数组放置在前面板中，然后选择一个对象（例如数值常量)插入到数组框中。这样就创建了一个数值数组。

也可以直接在前面板中创建数组和相应的控制对象，然后将它们复制或者拖曳到流程图中，创建对应的常数。

还有很多在流程图中创建和初始化数组的方法，有些功能函数也可以生成数组。

3．2．2　数组控制对象、常数对象和显示对象

通过把数组与数值、布尔数、字符串或者簇组合在一起，可以在前面板和流程图中创建任何一种控制对象、常数对象和显示对象。数组元素不能是数组、图表或者图形。如果您想查看一些数组的例子，请查看 Examples\General\arrays.llb 中的例子。

3．2．3　自动索引

For 循环和 While 循环可以自动地在数组的上下限范围内编索引和进行累计。这些功能称为自动索引。在启动自动索引功能以后，当把某个外部节点的任何一维元素连接到循环边框的某个输入通道时，该数组的各个元素就将按顺序一个一个地输入到循环中。循环会对一维数组中的标量元素，或者二维数组中的一维数组等编制索引。在输出通道也要执行同样的工作——数组元素按顺序进入一维数组，一维数组进入二维数组，依此类推。

在默认情况下，对于每个连接到 For 循环的数组都会执行自动索引功能。可以禁止这个功能的执行，方法是用鼠标右键单击通道（输入数组进入循环的位置），在快捷菜单中选择 Disable Indexing。

练习　3－1　创建一个自动索引的数组

目的：使用 For 循环的自动索引功能创建数组，并用一个图形（Graph）显示该数组。

前面板

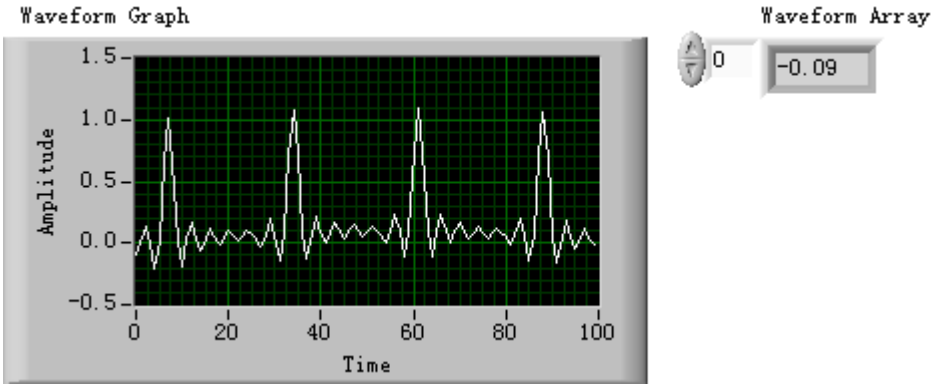
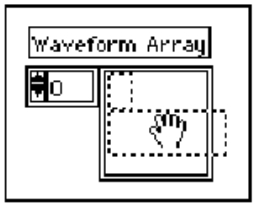


图 3－2　练习 3－1 的面板

1. 打开一个新的前面板。
2. 选择 Controls»Array & Cluster，在前面板中放置一个数组。设置它的标签为 Waveform Array。

3. 选择 Controls»Numeric，在



数组框中插入一个数字式显示对象。如右图所示。它用于显示数组的内容。

4. 选择 Controls»Graph，在前面板中放置一个波形图。设置它的标签为 Waveform Graph。

5. 隐藏图例和模板。

6. 用鼠标右键单击图形，并在快捷菜单中取消选中 Y Scale»Autoscale Y，禁止自动坐标功能。

7. 使用文本工具，把 Y 轴的范围改为-0.5 到 1.5。

流程图

8. 按照下图创建流程图。

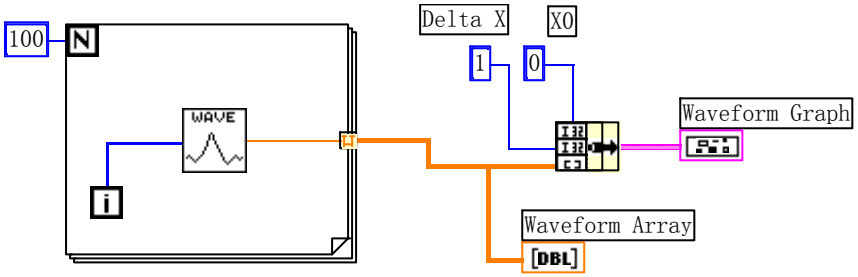


图 3 － 2　练习 3 － 1 的流程图

● 由 Functions»Select a VI… 寻找 LabVIEW\activity 目录下的 Generate Waveform VI ，它的作用是返回波形中的某一点。这个 VI 需要输入一个索引，我们将循环周期连接到这个输入。

● 注意 Generate Waveform VI 连出来的连线在循环边界变成一个数组时会变粗，正是在这个边界处形成了一维数组。

● For 循环会自动累计边界内的数组。这种功能叫做自动索引。在这个例子中，连接到循环计数输入的数值常数令 For 循环创建了一个由 100 个元素组成的数组。

● Bundle 函数（ Functions» Cluster）——将图块中的各个组件组合成一个簇，在正确连接以前需要改变该函数的图标的大小。将移位工具放在图标的左下角。变形工具会变成如左图所示，拖曳鼠标直到出现第三个输入端子。

● 数值常数（ Functions»Numeric）——三个数值常数用于设置 For 循环执行的周期数 N=100，初始 X=0 和 delta X=1。

9. 从前面板执行该 VI。该 VI 将把自动索引后的波形图数组显示在波形图中。

10. 把 X 的 delta 值改为 0.5，X 的初始值改为 20。再次执行该 VI。注意，波形图现在同样显示 100 个点，而每个点的初始值为 20，X 的 delta 值为 0.5（见 X 轴）。

11. 只需在显示器中输入元素的索引号就可以查看波形数组中的任何元素。如果输入的数比数组的元素个数大，那么显示

器将变暗，表示您没有为该元素设置索引。

如果需要一次查看多个元素，可以通过改变数组显示对象的大小来实现。把定位工具放置在数组框的右下角。工具将变成右图所示的变形工具。当工具变形时，用鼠标拖曳数组的右边或者下边。数组现在就可以按照元素索引的上升顺序显示多个素，以某个与指定索引对应的元素开始，如下图所示。

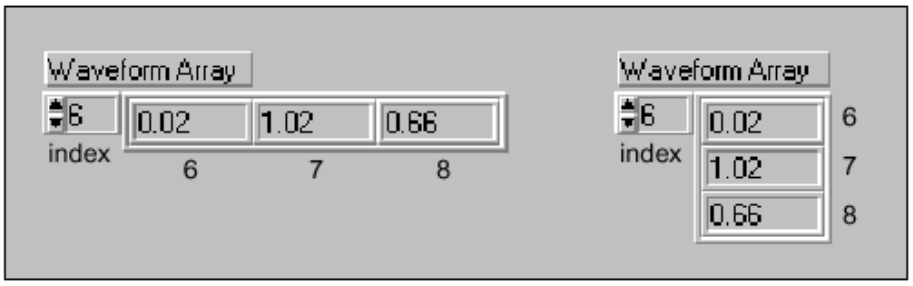


图 3 － 3　练习 3 － 1 中多个数组元素的同时观察

在前面的流程图中，您为波形图指定了初始的 X 值和 delta X 值。默认的 X 初始值是 0，delta X 值是 1。这样，也可以把波形数组直接连接到波形图端子， 而无需指定初始的 X 值和 delta X 值，如图 3 － 4 所示。

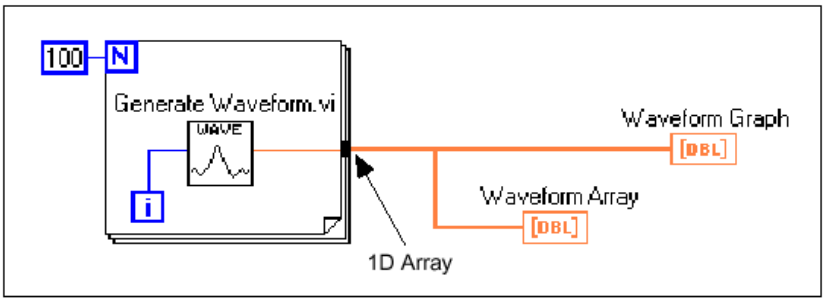


图 3 － 4　练习 3 － 1 使用默认 X 及 Delta X 时简化后的流程图

12. 按上图删除 Bundle 功能函数和它所连接的常数对象。方法是用移位工具选择该功能函数和连接的常数对象，按下〈Delete〉。再选择 Edit»Remove Bad Wires。按照上图完成流程图的连线。

13. 执行该 VI。注意初始的 X 值是 0，delta X 值是 1。

- **多图区图形**

可以创建含有多条曲线的图形，方法是创建一个数组，用它来汇集传给单图区图形的类型的数据元素。

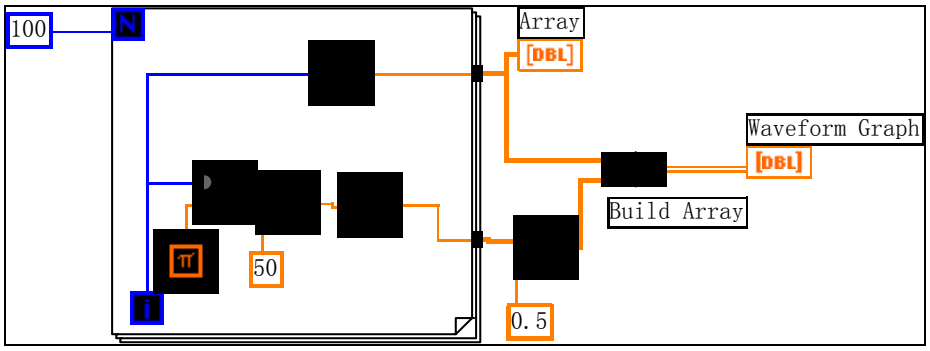


图 3 － 5　练习 3 － 1 多图区图形的流程图

14. 按照上图创建流程图。

● 正　　弦　　函　　数  （Functions»Numeric»Trigonometric）——在这里，它用于在 For 循环中创建一个由数据点组成的数组，表示一个正弦波周期。

● Pi 常数（Functions»Numeric»Additional Numeric Constants） 

● Build Array （Functions»Array）——在这里，它用于创建合适的数据结构(一个二维数组)，在波形图中绘制两条曲线。。用移位工具拖曳边角可以增大该函数的面积，创建两个输入端子。 

15. 返回前面板，执行该 VI。注意同一个波形中的两个图区。默认情况下，它们的 X 初始值都是 0，delta X 初始值都是 1。下图是该程序的运行结果（前面板未改动）。

16. 把该 VI 保存为 LabVIEW\Activity 目录中的 Graph Waveform Arrays.vi。

17. 可以修改图形中的某个图区的外观。方法是，用鼠标右键单击这个图形，再从弹出菜单选择对应的图例。

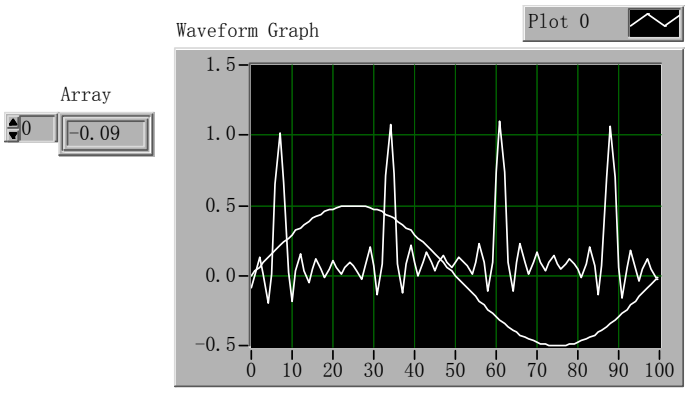


图 3 － 6　练习 3 － 1 多图区图形的面板显示

练习　3 － 1　结束。

在上面这个例子中，由于计算端子连接了一个值为 100 的常数对象，所以 For 循环将执行 100 次。下面这个例子显示了另

外一种控制循环执行次数的方法。

更详细的例子建议调阅 Examples》Fundamentals》Graphs and Charts》Graph Examples》Waveform Graph 例子。

练习 3－2 对输入数组使用自动索引功能

目的：打开并执行一个 VI，它将在一个 For 循环中使用自动索引功能处理一个数组。

1. 选择 File»Open…，打开 Examples\General\arrays.llb 中的 Separate Array Values VI 。

2. 打开流程图。下面的示意图显示的是在 TRUE 和 FALSE 时的情况。

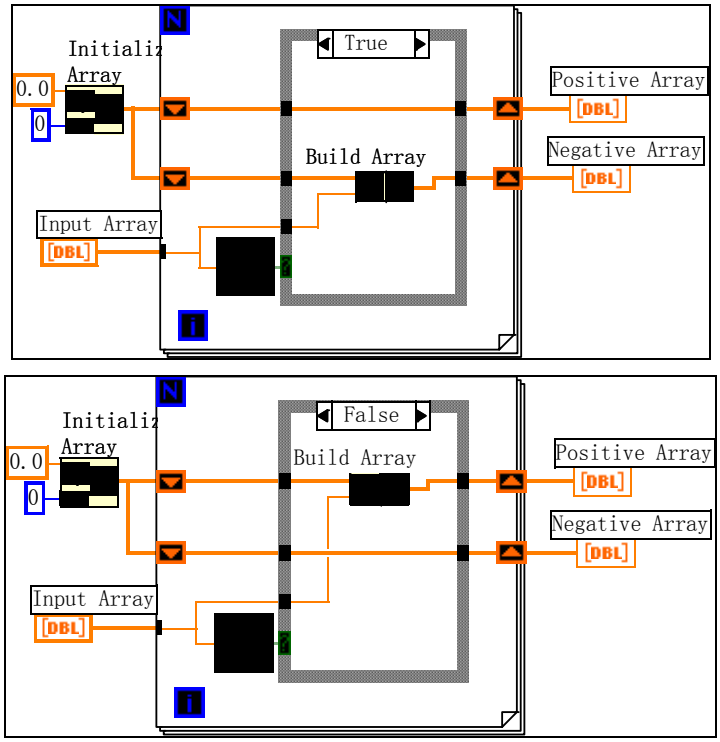


图 3－7 练习 3－2 的流程图

注意，Input Array 引出的连线与 For 循环外的粗线不同，表示这是一个数组，而循环内部的细线则表示这是一个数组元素。 数组元素在每个循环期间将自动编号。

- 用自动索引功能设置 For 循环的计数器

注意，计数器端子还没有连线。当您对某个进入 For 循环的数组使用自动索引功能时，循环就将根据数组的大小执行相应的次数，这样就无需连接某个值到计数器的端口。如果对一个以上的数组使用自动索引功能，或者在使用自动索引功能之外还需要设置计数器时，实际的循环次数将是其中最小的数。

3. 执行该 VI。在输入的八个数中，可以看到 4 个属于正数数组，另外 4 个属于负数数组。

4. 从流程图中将一个值为 5 的常数对象连接到 For 循环的计数器端子。执行该 VI。可以看到尽管输入数组仍然有八个

元素，但是 3 个位于正数数组，另外 2 个位于负数数组。这说明，如果设置了 N 并开启了自动索引功能，那么实际循环的次数将取较小的数。

5. 关闭该 VI，不要保存任何修改。

注：练习 3－2 的算法说明

下面是一段伪代码，解释上面的算法，假定输入数组为 A（已赋值），B（正数）、C（负数）。Sbr 、Scr 分别是与 B 数组、C 数组对应的右寄存器数组，Sbl 、Scl 分别是与 B 数组、C 数组对应的左寄存器数组，size 运算为测数组实际大小，ins 运算为将一个数插入数组中最左边的空位。

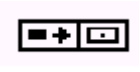
B=0	初始化
C=0	
K=size(A(.))	测A数组大小
For i=0 to k-1	
p=A(i)	取第I个元素值
if p>=0 then	
Ins p,Sbr	将p值插入右寄存器
Else	
Ins p,Scr	
end if	
Sbl=Sbr	右寄存器值送给左寄存器
Scl=Scr	
Next i	
B=Sbr	右寄存器值送到正数组
C=Scr	
Print B	显示
Print C	
End	

练习 3－2 结束。

3．3 数组功能函数

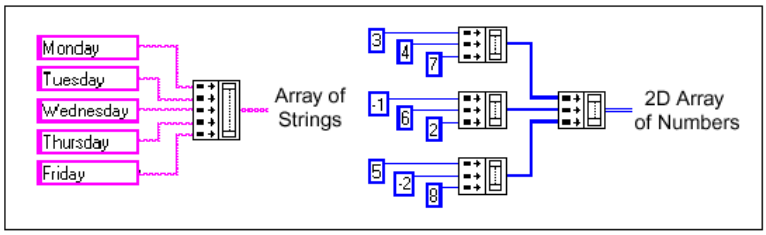
LabVIEW 提供了很多用于操作数组的功能函数，位于 **Functions»Array** 中。其中包括 Replace Array Element,、Search 1D Array、Sort 1D Array、 Reverse 1D Array 和 Multiply Array Elements 等等。

- 创建数组——Build Array 函数（Functions»Array），用于根据标量值或者其他的数组创建一个数组。



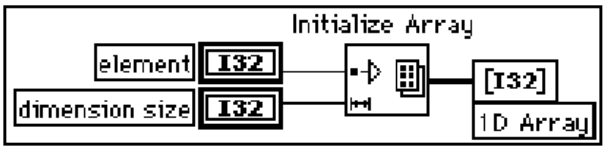
开始时，Build Array 函数具有一个标量输入端子。您可以根据需要向该功能函数中加入任意数量的输入，输入可以是标量或者数组。如果要添加其他的输入，用鼠标单击函数的左侧，在弹出菜单中选择 **Add Element Input** 或者 **Add Array Input**。还可以用变形工具来增大节点的面积（把移位工具放置在某个对象的边角就会变成变形光标）。也可以使用变形光标或者选择 **Remove Input** 来删除输入。

下图显示了利用流程图中的常数对象的值创建和初始化数组的两种方法。左侧的方法是，将 5 个字符串常数放入一个一维字符串数组中。右侧的方法是，将三组数值常数放入三个一维数值数组，再将这三个数组组成一个二维数组。这样最后产生的是一个 3x3 的数组，三列分别是 3, 4, 7; –1, 6, 2; 5, –2, 8.。



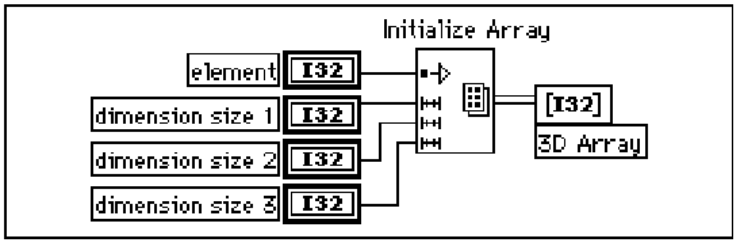
还可以通过结合其他的含有标量元素的数组来创建数组。例如，假设有两个数组，三个标量元素，可把它们组成一个新的数组，顺序是：数组 1，标量 1，标量 2，数组 2，标量 3。

- 初始化数组（Initialize Array）——用于创建所有元素值都相等的数组。下图  中，该功能函数创建了一个一维数组。



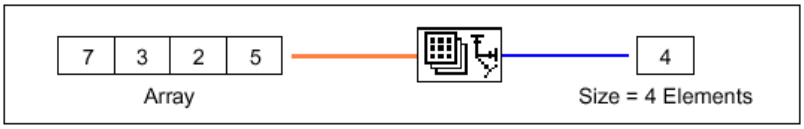
元素输入端子决定每个元素的数据类型和数值，维长度输入端子决定数组的长度，例如，假设元素类型是长整型，值为 5，维长度为 100，那么创建的数组是一个一维的、由 100 个值为 5 的长整型元素组成的数组。也可以从前面板控制端子、流程图常数或者程序其他部分的计算结果得到输入。

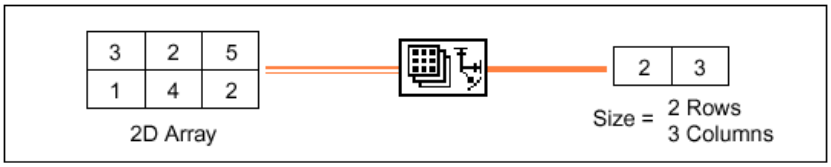
创建和初始化一个多维数组的方法是，用鼠标右键单击函数的右下侧，在弹出菜单中选择 **Add Dimension**。还可以使用变形光标来增大初始化数组节点的面积，为每个增加的维添加一个维长度输入端子。也可以通过缩小节点的方法来删除维，即从函数的弹出菜单中选择 **Remove Dimension**，或者使用变形光标。下面的示意图显示了怎样初始化一个三维数组。



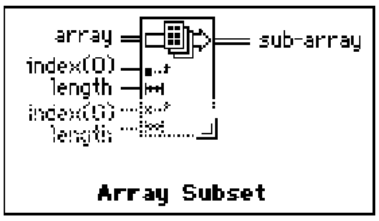
如果所有的维长度输入都是 0，该函数会创建一个具有指定数据类型和维数的空数组。

- 数组大小——Array Size 函数，返回输入数组中的元素个数。

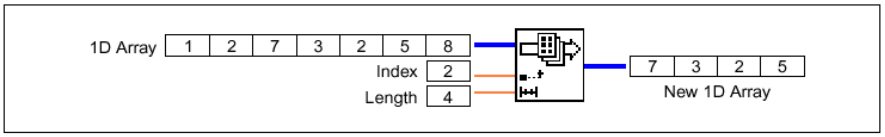


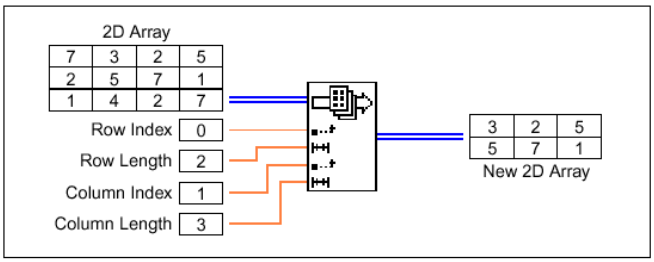


- 数组子集（Array Subset）——选取数组或者矩阵的某个部分。

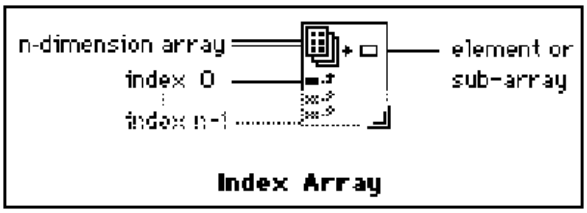


该函数可以返回从某个指针开始的部分数组，并包括了长度元素。下图显示了一些数组子集的例子，注意，数组索引从 0 开始。





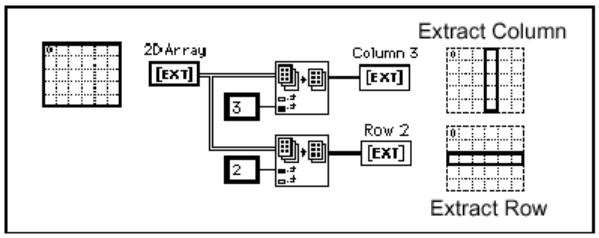
- 索引数组（Index Array）——用于访问数组中的某个元素。



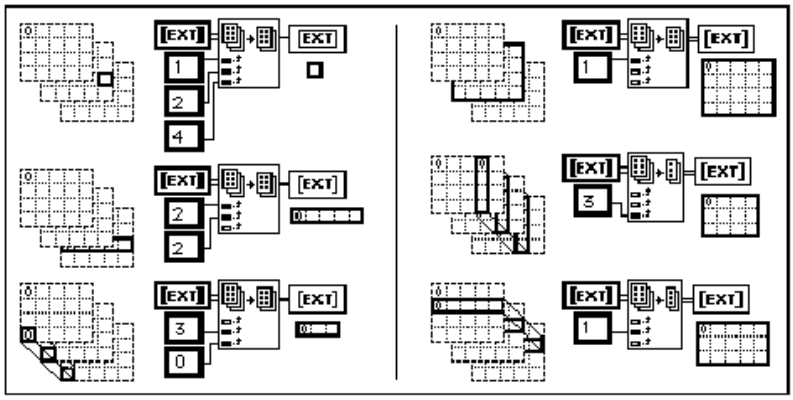
下图显示了一个索引函数的例子，它用于访问数组中的第三个元素。注意，因为第一个元素的索引为 0，所以第三个元素的索引是 2。

将一个二维数组与 Index Array 函数相连，Index Array 就会含 2 个索引端子。将一个三维数组与 Index Array 函数相连，Index Array 就会含 3 个索引端子。余类推。可以使用的索引端的符号是一个黑方快，被禁止使用的索引端（**Disable Indexing**）是一个空心的小方框。当给一个被禁止使用的索引端连接上一个 Constant 或 Control 是它会自动变为黑方快，即变为可以索引，相反原来一个可以使用的索引端上连接的 Constant 或 Control 被删去时，索引端符号会自动变为空心的小方框，即变为禁止使用。

也可以按照任何维的组合提取子数组，下面的示意图显示了怎样从一个二维数组中提取一个一维的行或者列数组。



还可以从一个三维数组中提取一个二维数组，方法是禁止两个索引端子，或者通过禁止一个索引端子提取一个一维数组。下图显示了从三维数组提取数组的各种方法。



下面的规则对使用剪切数组进行了规定：

输出对象的维数必须等于被禁止的索引端口的数目。例如

0 个索引端口被禁止＝标量元素

1 个索引端口被禁止＝二维元素

2 个索引端口被禁止＝三维元素

启动的端子所连接的数值必须指定输出元素。

这样，您就可以理解，上图中左下方的例子的作用是，利用 0 列和 3 行的所有元素产生一个一维数组，而右上方的例子的作用是利用第一帧中的所有元素产生一个二维数组。新的第 0 个元素是与原有元素最近的元素。

练习　3－3　使用创建数组功能函数

目的：使用创建数组函数，把一些元素和输出组织成一个更大的数组。

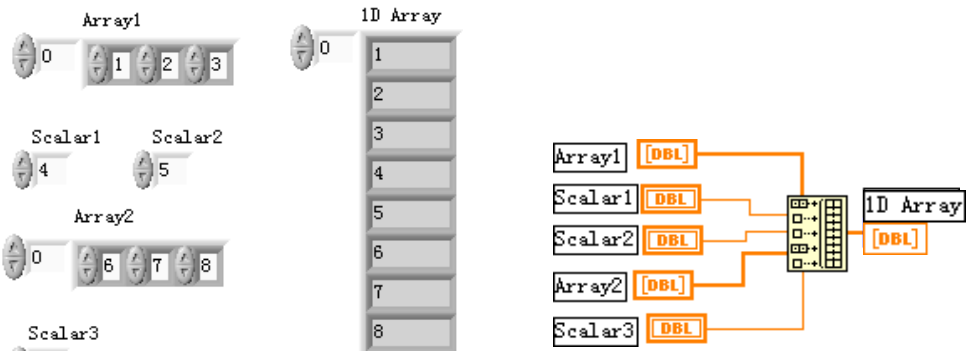


图 3 － 8　练习 3 － 3 的面板和框图

1. 按照图 3 － 8 创建一个前面板。

2. 从 Controls»Numeric 模板中选择一个数字控制对象放置在前面板中，设置它的标签为 scalar 1。

3. 复制并粘贴该数字显示对象，创建两个新的对象，并分别设置它们的标签为 scalar 2 和 scalar 3。

4. 创建一个数字控制对象的数组，设置它的标签为 array 1。复制并粘贴它，创建一个新的数组，设置它的标签为 array 2。

5. 在 array 1、scalar 1、 scalar 2、scalar 3、array 2 中输入数值 1 到 9。

6. 创建流程图。选择 Functions»Array，在流程图中放置一个 Build Array 功能函数。用定位工具增大函数额面积，以容纳 5 个输入。

7. 把数组和标量与 Build Array 连接起来。创建输出的一维数组，它由 array 1、scalar 1、 scalar 2、array 2、scalar 3 中的元素所组成，如图所示。

8. 执行该 VI。可以看到 array 1、scalar 1、 scalar 2、scalar 3、array 2 中的数值出现在同一个一维数组中。

9. 保存该 VI 为 LabVIEW\Activity 目录下的 Build Array.vi。

练习 3 － 3　结束。

3．4　什么是多态化（Polymorphism）？

多态化是指一种函数功能，即可以协调不同格式、维数或者显示的输入数据。大多数 LabVIEW 的函数都是多态化的。例如，图 3 － 9 给出了 Add 函数的一些多态化组合。

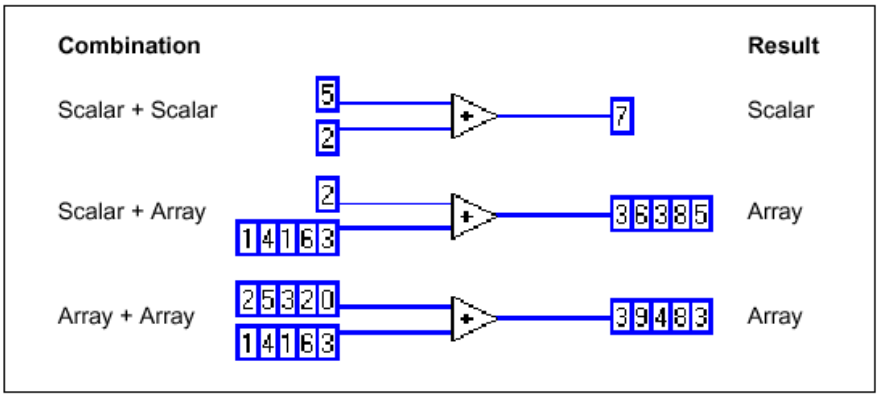


图 3 － 9　多态化组合的例子

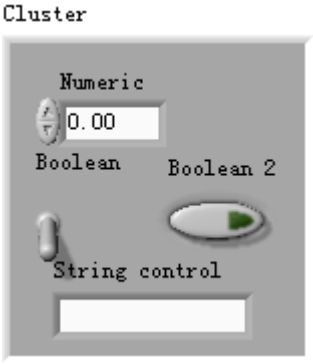
第一个组合中，两个标量相加，结果还是一个标量。第二个组合中，该标量与数组中的每个元素相加，结果是一个数组。数组是数据的集合。第三个组合中，一个数组的每个元素被加到另一个数组的对应元素中。您还可以使用其他的组合，例如数值簇或者簇数组。

可以把这些准则应用到其他的 G 语言函数或者数据类型。G 语言函数对于各种情况都具有多态性功能。有些函数接受数值和布尔输入，而有些函数接受其他任何数据格式的组合。如果您想了解更多关于多态化的知识，请参阅 **Online Reference»Function and VI Reference**。

3．5　簇

3．5．1　创建簇控制和显示

在前面板上放置一个簇壳（Cluster shell）就创建了一个簇。然后你可以将前面板上的任何对象放在簇中。例如数组，你也可以直接从 Control 工具板上直接拖取对象堆放到簇中。一个簇中的对象必须全部是 Control，或全是 Indicator，不能在同一个簇中组合 Control 与 Indicator，因为簇本身的属性必须是其中之一。一个簇将是 Control 或 Indicator，取决于其内的第一个对象的状态。如果需要可以使用工具重置簇的大小。右图所示是一个含 4 个 Control 的簇。也可以在流程图上用类似的方法创建簇常数。



如果你要求簇严格地符合簇内对象的大小，可在簇的边界上弹出快速菜单选择自动定义大小（Autosizing）

簇的序（Order）

簇的元素有一个序，它与簇内元素的位置无关。簇内第一个元素的序为 0，第二个是 1，等等。如果你删除了一个元素，序号将自动调整。如果你想将一个簇与另一个簇连接，这两个簇的序和类型必须同一。

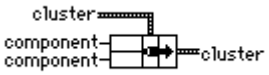
如果想改变簇内元素的序，可在快速菜单中选择 **Claster Order** ，这时会出现一个窗口，在该窗口内可以修改序。

3．5．2　使用簇与子 VI 传递数据

一个 VI 的连接窗口最大有 28 个端子，如果你不希望使用全部 28 个端子传递数据，这既烦琐又易出错。通过把控制或显示对象捆绑成一个簇的方法，仅使用一个端子就可以实现该功能。

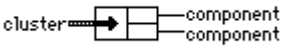
- 捆绑（Bundle）数据 

Bundle 功能将分散的元件集合为一个新的簇，或允许你重置一个已有的簇中的元素。可以用位置工具拖曳其图标的右下角以增加输入端子的个数。最终簇的序是取决于被捆绑的输入的顺序。右图中 Bundle 图标中部的 Claster 端子用于用新元素重置原簇中的元素。



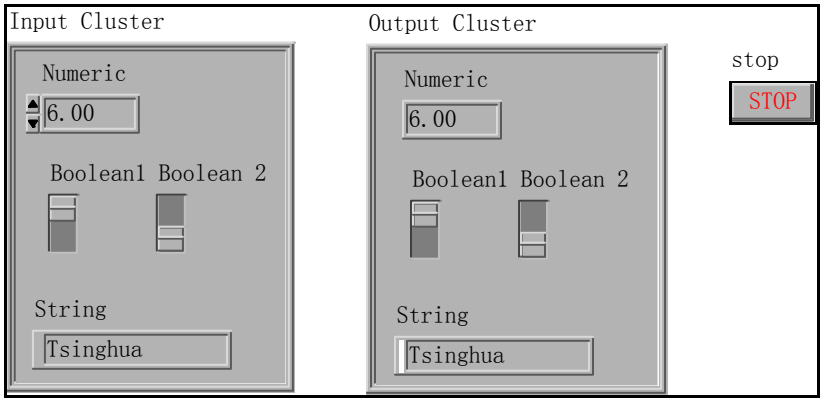
- 分解（Unbundle）簇 

Unbundle 功能是 Bundle 的逆过程，它将一个簇分解为若干分离的元件。如果你要对一个簇分解，就必须知道它的元素的个数。LabVIEW 还提供一种可以根据元素的名字来捆绑或分解簇的方法，稍后介绍。



练习 3－4　簇

目的：学习创建簇、分解簇，再捆绑簇并且在另一个簇中显示其内容。



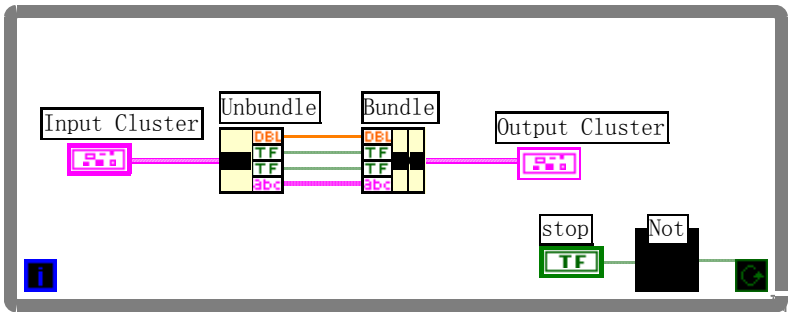


图 3－10　练习 3－4 的面板和框图

1．打开前面板，创建一个簇壳（Array & Cluster palette），标签改为 Input Cluster，拖曳至适当大小。

2．在这个簇壳中放置一个数字 Control，两个布尔开关，和一个串 Control。

3．仿照以上步骤，创建 Output Cluster 如上。注意将各 Control 改为相应的 indicator。

4．用快速菜单查看两个簇的序是否一致，若有差别，改之。
5．在前面板上设置一个[STOP]按钮。注意其缺省值为 FALSE，不要改变它的状态。
6．建立如上面所示的流程图。注意在[STOP]按钮与循环条件端子之间接入了一个 NOT 函数，因为按钮缺省值为 FALSE，经 NOT 函数后变为 TRUE，这就意味着当按钮状态不变时，循环继续执行，相反一旦按钮动作，则循环终止。
7．返回前面板并运行 VI。在输入簇中输入不同的值观察输出。
8．关闭并保存程序。Cluster Exercise.vi

练习 3－4 结束

3．5．3 用名称捆绑与分解簇

有时你并不需要汇集或分解整个簇，而仅仅需要对其一、两个元素操作。这时你可以用名称来捆绑与分解簇。在 Cluster 工具模板中除了 Bundle 及 Unbundle 功能外，还提供有 Bundle By Name 和 Unbundle By Name 功能。它们允许根据元素的名称（而不是其位置）来查询元素。与 Bundle 不同，使用 Bundle By Name 可以访问你需要的元素，但不能创建新簇；它只能重置一个已经存在的簇的元素，同时你必须给 Bundle By Name 图标中间的输入端子一个输入以申明要替换其元素的簇。Unbundle 可返回指定名称的簇元素，不必考虑簇的序和大小。例如，如果你想重置上例中 Boolean 2 的值，就可以使用 Bundle By Name 功能而不必担心簇的序和大小。与此类似如果你要访问串的值，可以使用 Unbundle By Name 功能。

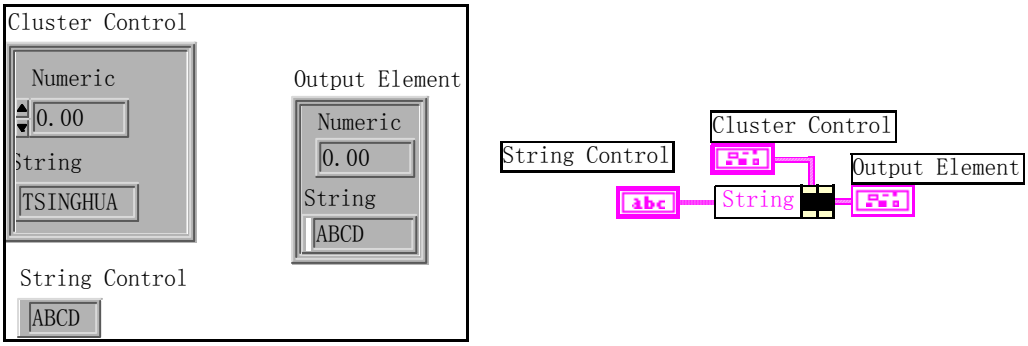
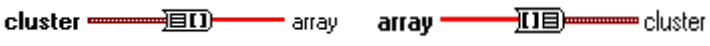


图 3－11 用名称操作簇

在上面的例子中，Cluster Control 中有两个元素，一个是数据类型（名称是 Numeric），另一个是字符串型（名称是 String），另一个控制是字符串“ABCD”，框图如右所示，运行该程序，即可将簇内的字符串值重置。

3．5．4 数组和簇的互换

有时你会发现，将数组变为簇（或者相反）很方便。尤其是因为 LabVIEW 包括的关于数组的操作功能多于簇。例如，前面板上有一个多按钮的簇，你希望颠倒这些按钮值的序。好了，Reverse 1D Array 功能正好可用，但是它仅可用于数组。这没关系，你可以使用功能 Cluster to Array 将簇转换为数组，使用 Reverse 1D Array 切换开关的值，最后再利用 Array to Claster 变换回簇。



3．6 Waveform 数据类型

在数据采集和信号分析中经常要遇到波形数据，在 LabVIEW 6i 中增加了 Waveform 数据类型，使得波形的描述更加简洁。Waveform 数据类型包含了波形的数据(Y)、起始时刻(t0)和步长ΔX，使用 Waveform 模板的 Build Waveform 函数可以建立一个波形。许多用于数据采集和波形分析的 VI 和函数的缺省状态都接受或返回 Waveform 数据类型。当你将一个 Waveform 数据类型连接到 Waveform Graph 或 Chart 时，会自动画出相应的曲线。Waveform 数据类型是根据原有的数据类型进一步“打包”组合而成，这种打包也不可避免地带来一些负作用，有时还需要对 Waveform 数据类型“解包”。有关这一数据类型的函数或 VI 在 Functions» Waveform 和 Analyze 之中。

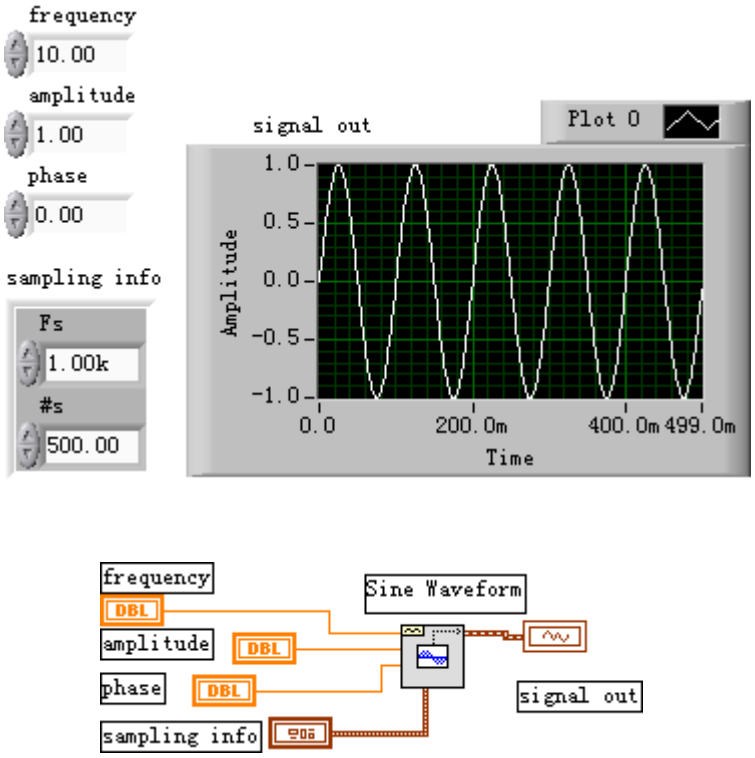


图 3 － 1 2　使用 Waveform 的波形发生例子

图 3 － 1 2 是一个使用 Waveform 函数产生正弦波的例子。其中仅仅调用了 Sine Waveform 一个函数，只要将有关参数指定，就可产生正弦波。Sine Waveform 实际上是一个子 VI，点击其图标，就可看到下层的程序，还是比较复杂的。在 LabVIEW 6i 以前的版本中用户就需要那样去编程。

第四章　图形显示

４．１　概述

图形显示对于虚拟仪器面板设计是一个重要的内容。LabVIEW 为此提供了丰富的功能。在前面几章我们已经接触了这个问题，现在较系统地介绍一下。

我们不从图形的实现方法上去讨论问题,那是计算机图形学的课题。但我们需要从用户的可能的需求角度探求一下，如果你需要做虚拟仪器方面的开发，那么可能遇到些什么图形问题。LabVIEW 在这方面所做的工作是非常值得借鉴的。

在 LabVIEW 的图形显示功能中 Graph 和 Chart 是两个基本的概念。一般说来 Chart 是将数据源（例如采集得到的数据）在某一坐标系中，实时、逐点地显示出来，它可以反映被测物理量的变化趋势，例如显示一个实时变化的波形或曲线，传统的模拟示波器、波形记录仪就是这样。而 Graph 则是对已采集数据进行事后处理的结果。它先将被采集数据存放在一个数组之中，然后根据需要组织成所需的图形显示出来。它的缺点是没有实时显示，但是它的表现形式要丰富得多。例如采集了一个波形后，经处理可以显示出其频谱图。现在，数字示波器也可以具备类似 Graph 的显示功能。

LabVIEW 的 Graph 子模板中有许多可供选用的控件，其中常用的见下表：

	Chart	Graph
Waveform（波形）	*	*
XY		*
Intensity（强度图）	*	*
Digital（数字图）		*
3D Surface（三维曲面）		*
3D Parametric（三维参变量）		*
3D Curve（三维曲线）		*

由表中可以看出，Chart 方式尽管能实时、直接地显示结果，但其表现形式有限，而 Graph 方式表现形式要远为丰富，但这是以牺牲实时为代价的。在 LabVIEW　6 i 版本中还包含有极坐标等其他图形（Plot），本章不讨论。

４．２　Graph 控件

各种图形都提供了相应的控件，以 Graph 为例介绍。图 4 － 1 所示为它的控件。所有这些控件都包含在图形快速菜单的 **Visible Items** 选项下。

曲线图例可用来设置曲线的各种属性，包括线型（实线、虚线、点划线等）、线粗细、颜色以及数据点的形状等。

图形模板可用来对曲线进行操作，包括移动、对感兴趣的区域放大和缩小等。

光标图例可用来设置光标、移动光标，帮助你用光标直接从曲线上读取感兴趣的数据。

刻度图例用来设置坐标刻度的数据格式、类型（普通坐标或对数坐标），坐标轴名称以及刻度栅格的颜色等。

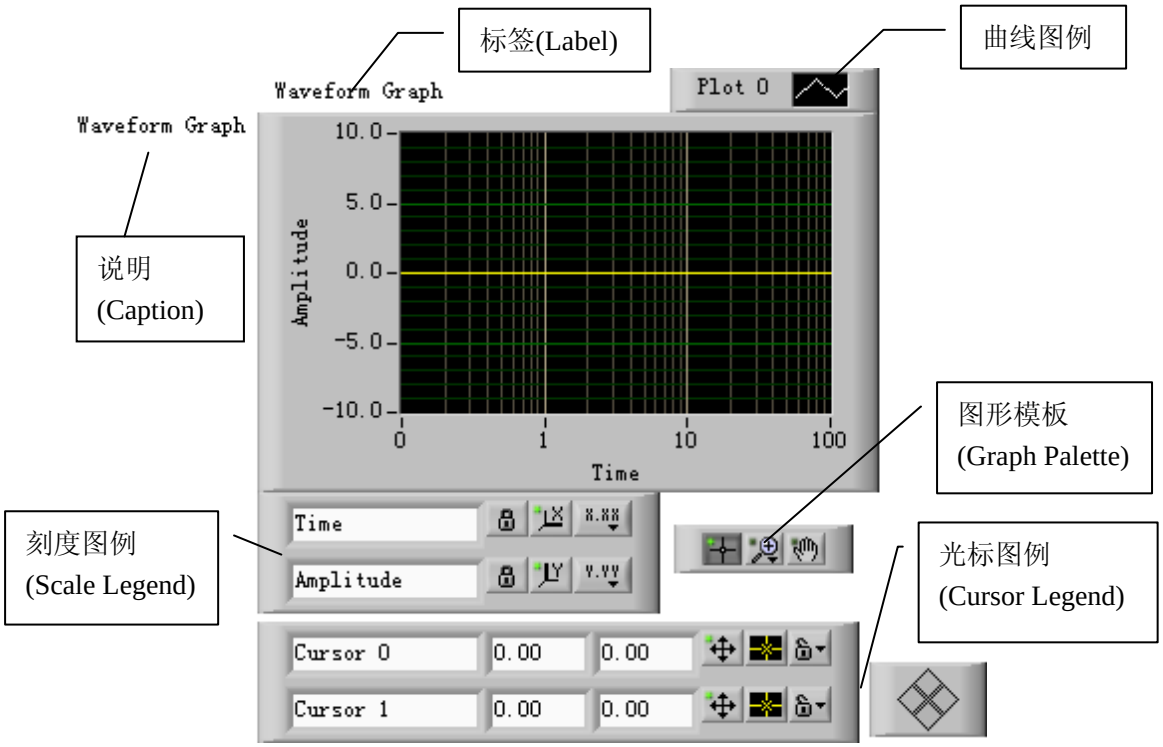


图 4-1　Graph 的图形控件

４．３　Chart 的独有控件

Chart 的数据并没有事先存在一个数组中，它是实时显示的，为了能够看到先前的数据，Chart 控件内部含有一个显示缓冲器，其中保留了一些历史数据。这个缓冲器按照先进先出的原则管理，其最大容量是 1024 个数据点。

- 滚动条(Scrollbar)

  它直接对应于显示缓冲器，通过它可以前后观察缓冲器内任何位置的数据。

- 数据显示(Digital Display)

  选中它，可以在图形右上角出现一个数字显示器，这样可以在画出曲线的同时显示当前最新的一个数据值。

- 刷新模式(Update Mode)

  Chart 提供了三种画面的刷新模式，分别是

  - Strip Chart Mode（条壮图）：它与纸带式图表记录仪类似。曲线从左到右连续绘制，当新的数据点到达右部边界时，先前的数据点逐次左移。
  - Scope Chart Mode（示波器模式）：它与示波器类似。曲线从左到右连续绘制，当新的数据点到达右部边界时，清屏刷新，从左边开始新的绘制。它的速度较快。
  - Sweep Chart Mode（扫描模式）：与示波器模式的不同在于当新的数据点到达右部边界时，不清屏，而是在最左边出现一条垂直扫描线，以它为分界线，将原有曲线逐点向右推，同时在左边画出新的数据点。如此循环下去。

- 堆叠式图区（Stack Plots）

在相同的纵坐标下，由于各种测量信号的差异，将几条曲线显示在同一个图区有困难时，可以组织出一种纵坐标相同，而有各自横坐标的堆叠式图区。

打开\LabVIEW\Examples\Graphs\chart.lib 目录的 Charts.vi，那里提供了有关 Chart 的各种形态的例子，以及堆叠式图区的例子

练习 4－1 Chart 和 Graph 的比较

目的：创建一个 VI，用 Chart 和 Graph 分别显示 40 个随机数产生的曲线，比较程序的差别。

前面板及流程图如下

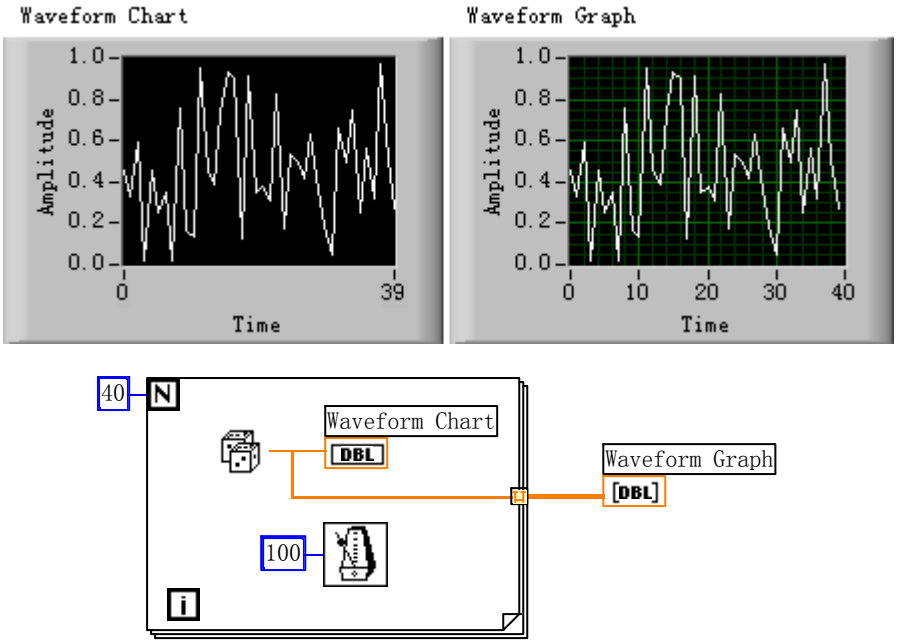


图 4－2 Chart 和 Graph 的比较

显示的运行结果是一样的。但实现方法和过程不同。在流程图中可以看出，Chart 产生在循环内，每得到一个数据点，就立刻显示一个。而 Graph 在循环之外，40 个数都产生之后，跳出循环，然后一次显示出整个数据曲线。从运行过程可以清楚地看到这一点。

值得注意的还有 For 循环执行 40 次，产生的 40 个数据存储在一个数组中，这个数组创建于 For 循环的边界上（使用自动索引功能）。在 For 循环结束之后，该数组就将被传送到外面的 Graph。仔细看流程图，穿过循环边界的连线在内、外两侧粗细不同，内侧表示浮点数，外侧表示数组。

练习 4－1 结束

4．4 XY 图形控件（XY Graph）

波形图（Waveform Graph）有一个特征，其 X 是测量点序号、时间间隔等，Y 是测量数据值。但是它并不适合描述一般的 Y 值随 X 值变化曲线。适合于这种情况的控件是 XY Graph。我们通过一个构成利萨育图形的例子来看一下它的使用。我们知道如果控制 XY 方向的两个数组分别按正弦规律变化（假设其幅值、频率都相同），如果它们的相位相同，则利萨育图形是一条 45 度的斜线，当它们之间相位差 90 度时为圆，其他相位差是椭圆。

练习 4 － 2　利用 XY Graph 构成利萨育图形。

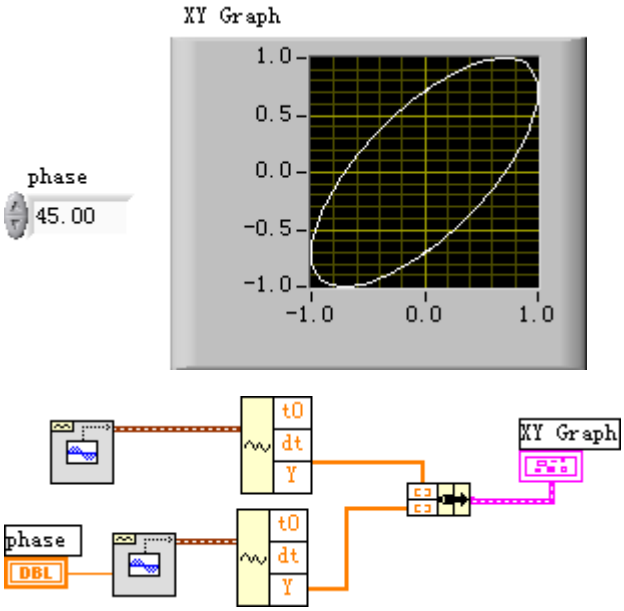


图 4 － 3　利用 XY Graph 构成利萨育图形

面板和框图如图 4 － 3 所示。面板上除了一个 XY Graph 外，还有一个相位差输入控件。在框图中使用了两个 Sine Waveform.vi，第一个所有输入参数（包括频率、幅值、相位等）都使用缺省值，所以其初始相位为 0 。第二个将其初始相位作为一个控件引到面板上。它们的输出是包括 t0、dt 和 Y 值的簇，但是对于 XY Graph 只需要其中的 Y 数组，因此使用波形函数中的 Get Waveform Components 函数分别提取出各自的 Y 数组，然后再将他们捆绑在一起，连接到 XY Graph 就可以了。当相位置为 45 度时，运行程序，得到如图所示的椭圆。

练习 4 － 2　结束

4．5　强度图形控件（Intensity Graph）

强度图形控件提供了一种在二维平面上表现三维数据的方法。例如可以用屏幕色彩的亮度来反映来反映一个二维数组元素值的大小。图 4 － 3 就是这样的一个例子。注意图中的 x、y 轴刻度对应的是数组行、列的序号。

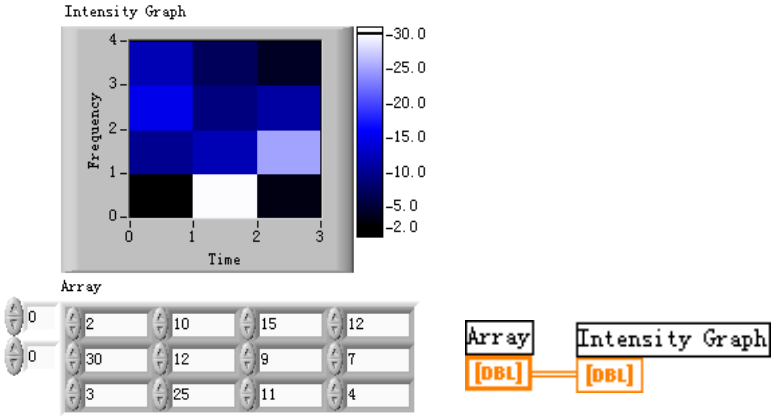


图 4 － 4　用强度图形表现一个二维数组及其元素值的大小

4．6　数字波形图控件（Digital Waveform Graph）

该控件的例子见图 4 － 5 。面板上有一个输入控制，共输入了 7 个十进制数，还有一个二进制显示对象，显示了这些十进制数对应的二进制数，最右边是一个 Digital Waveform Graph。注意这个图中数据应当从纵方向读出，在横坐标上的刻度是数据的序号（ 0 到 6 ），其中最后一个数的序号是 6 ，纵坐标从下向上读是 11111111，第一个数的序号是 0 ，其值从上向下读是 00000001，而第二个数（序号 1 ）是 00000010。

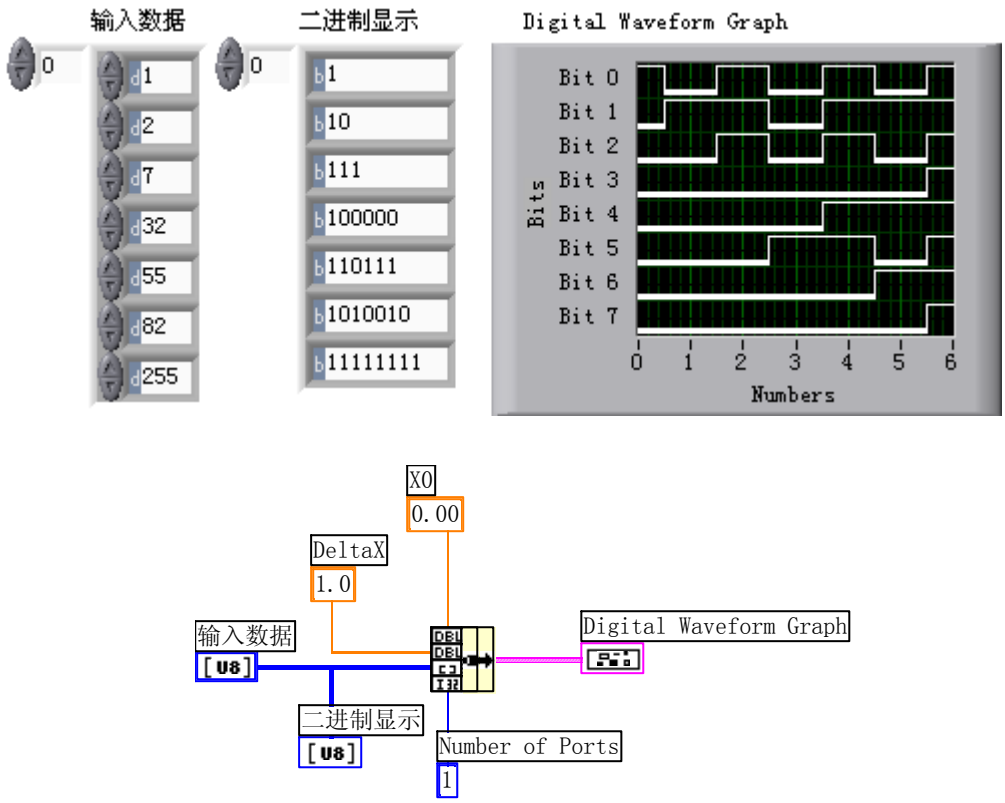


图 4 － 5　数字波形图控件的例图

该程序的框图中值得注意的问题有以下几点：

- 十进制数可以直接送给 Digital Waveform Graph，不必事先转化为二进制数。
- 在送给 Digital Waveform Graph 之前，需要经过一个捆绑（bundle 函数）。
- 捆绑的顺序是 x0、deltx 、输入数据，最后是 Number of　Ports。这里的 Number of Ports 将反映二进制的位数或字长，为 1 时是 8 位，为 2 时变为 1 6 位，余类推。

4．7　3D 图形显示控件（3D Graph）

第四章　字符串和文件 I/O

4．1　字符串

字符串是 ASCII 字符的集合。如同其他语言一样，LabVIEW 也提供了各种处理字符串的功能，如果想获得字符串的例子，可查看 Examples\General\strings.llb。下面扼要介绍一些内容。

- 创建字符串控制对象和显示对象

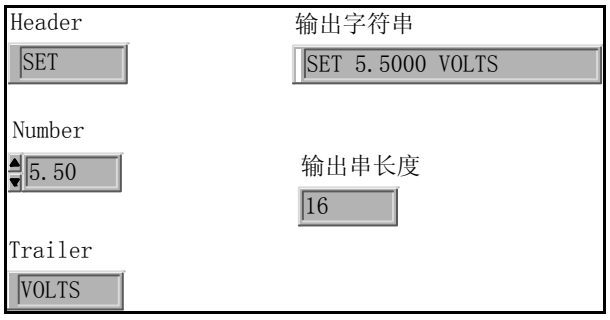
可以在 Controls»String & Table 中看到右边所显示字符串控制对象和显示对象。还可以使用操作工具或者标签工具输入或者改变字符串控制对象中的文本。用移位工具拖曳字符串控制对象和显示对象的边角可以增大它们的面积。

练习　4－1　组合字符串

目的：使用一些字符串功能函数将一个数值转换成字符串，并把该字符串和其他一些字符串连接起来组成一个新的输出字符串。

前面板

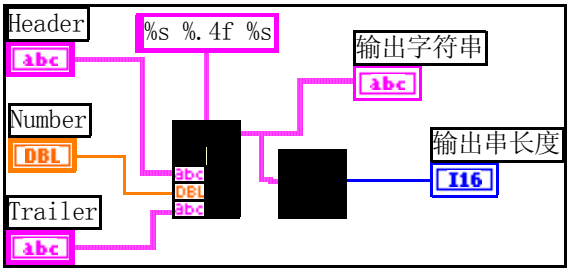
打开一个新的前面板，按照下图向其中添加对象。



其中的两个字符串控制对象和数值控制对象可以合并成一个输出字符串并显示在字符串显示器中。数值显示器显示出字符串的长度。

本练习中输出字符串是一个 GPIB（IEEE 488）命令字符串，它可用来与和串口仪器（RS-232 或者 RS-422）进行通信。

流程图



- Format Into String 函数（**Functions»String**）——在本练习中，它用于对

数值和字符串进行格式化，使它们成为一个输出字符串。用变形工具可以添加三个加和输入。

- String Length 函数（**Functions»String**）——在本练习中，它用于返回一个字符串的字节数

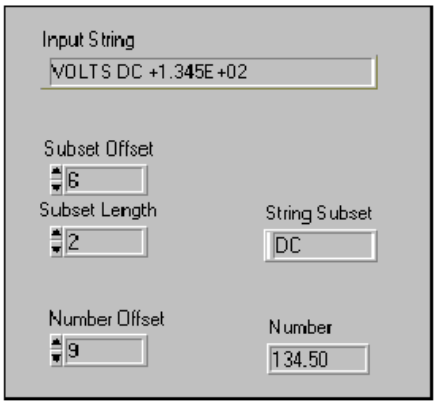
- 执行该 VI。注意，Format Into String 功能函数将两个字符串控制对象和数值控制对象组合成一个输出字符串。
- 把该 VI 保存为 Build String.vi，在下一个练习中还将用到这个 VI。
- 字符串格式的设定：选中 Format Into String 函数，单击右键，在快速菜单中选择 **Edit Format String**，可分别对个输入的各部分格式做设定。

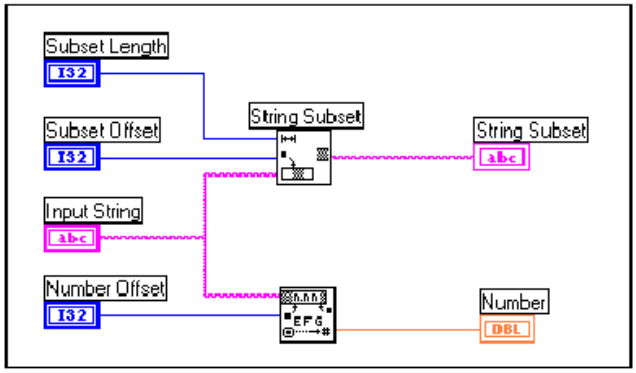
练习 4－1 结束。

练习 4－2 字符串子集和数值的提取

目的：创建一个字符串的子集，其中含有某个数值的字符串显示，再将它转换成数值。

打开 Examples\General\strings.llb 中的 Parse String.vi。用默认输入值执行该 VI。注意，*DC* 的字符串子集被用于输入字符串。还要注意，字符串的数值部分被提取出来，并转换为数值。您可以尝试使用不同的控制数值（记住数组式的字符串是从 0 开始进行编号），或者您可以返回到流程图，查看怎样从输入字符串中提取出其中的元素。





String Subset 函数（**Functions»String**）——在本练习中，它用于返回偏移地址开始的子字符串以及字节数。第一个偏移地址是 0。

很多情况下，必须把字符串转换成数值，例如需要将从仪器中得到的数据字符串转换成数值。

Scan From String 函数（**Functions»String**）——在这个例子中，它用于扫描字符串，并将有效的数值（0 到 9,正负，e，E 和分号）转换成数值。如果连接了一个格式字符串，它将根据字符串指定的格式进行转换，否则将进行默认格式的转换。该函数从偏移地址的 **string** 处开始扫描。第一个字符的偏移地址是 0。这个函数在已知头长度（本例中是 VOLTS DC）时或者字符串只含有有效字符时很有用。

选择 **File»Close**，关闭该 VI。注意不要保存它。

练习 4－2 结束。

4．2 文件的输入/输出（I/O）

文件 I/O 功能函数是一组功能强大、伸缩性强的文件处理工具。它们不仅可以读写数据，还可以移动、重命名文件与目录。创建电子表格格式的、由可读的 ASCII 文本组成的文件，以及为了提高读写速度和压缩率采用二进制的格式写入数据。

可以采用下面三种文件格式存储或者获得数据：

- ASCII 字节流——如果希望让其他的软件（譬如字处理程序或者电子表格程序）也可以访问数据，就需要将数据存储为 ASCII 格式。为此，您需要把所有数据都转换为 ASCII 字符串。
- 数据记录文件——这种文件采用的是只有 G 语言可以访问的二进制格式。数据记录文件类似于数据库文件，因为它可以把不同的数据类型存储到同一个文件记录中。
- 二进制字节流——这种文件的格式是最紧凑、最快速地存储文件的格式。您必须把数据转换成二进制字符串的格式，还必须清楚地知道在对文件读写数据时采用的是哪种数据格式。

因为 ASCII 字节流格式是最常用的数据文件格式，所以本节着重介绍这种格式。如果您想获得关于文件 I/O 的例子，请参考 Examples\File。

４．２．１　文件 I/O 功能函数

大多数的文件 I/O 操作都包括三个基本的步骤：打开一个已有的文件或者新建一个文件；对文件进行读写；关闭文件。LabVIEW 在 **Functions» File I/O** 中提供了很多有用的工具 VI。本节主要介绍 9 个高级工具 VI，这些工具 VI 可以把错误检查和错误处理等功能与文件 I/O 功能函数结合起来。

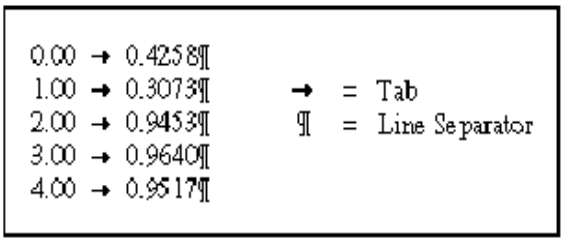
以下 5 个功能从左到右对应于上面 5 个图标。

- Write To Spreadsheet File VI——用于将由单精度数值组成的一维或者二维数组转换成文本字符串，再将它写入一个新建文件或者已有文件。该 VI 先打开或者新建文件，之后再关闭文件。它可以用于创建能够被大多数电子表格软件读取的文本文件。
- Read From Spreadsheet File VI——用于从某个文件的特定位置开始读取指定个数的行或者列内容，再将数据转换成二维、单精度数组。该 VI 先打开文件，之后再关闭文件。它可以用于读取用文本格式存储的电子表格文件。
- Write Characters To File VI——用于将一个字符串写入一个新建文件或者已有文件。该 VI 打开这个文件、写入数据，再关闭文件。
- Read Characters From File VI——用于从某个文件的特定位置开始读取指定个数的字符。该 VI 先打开文件，之后再关闭文件。
- Read Lines From File VI——用于从某个文件的特定位置开始读取指定个数的行内容。该 VI 先打开文件，之后再关闭文件。

如果想查看其他的文件 I/O 功能函数，请选择 **Function»File I/O» Binary File VIs** 或者 **Function»File I/O»Advanced File Functions**。

４．２．２　将数据写入电子表格文件

将数据存储到文件的最常见应用之一是设置文本文件的格式以便在电子表格文件中打开它。大多数电子表格文件用 Tab 键分割各列，而用 EOL（段尾）分隔各行，如下图所示。



用一个电子表格程序（如 Excel）打开该文件可以看到下面这个表格。

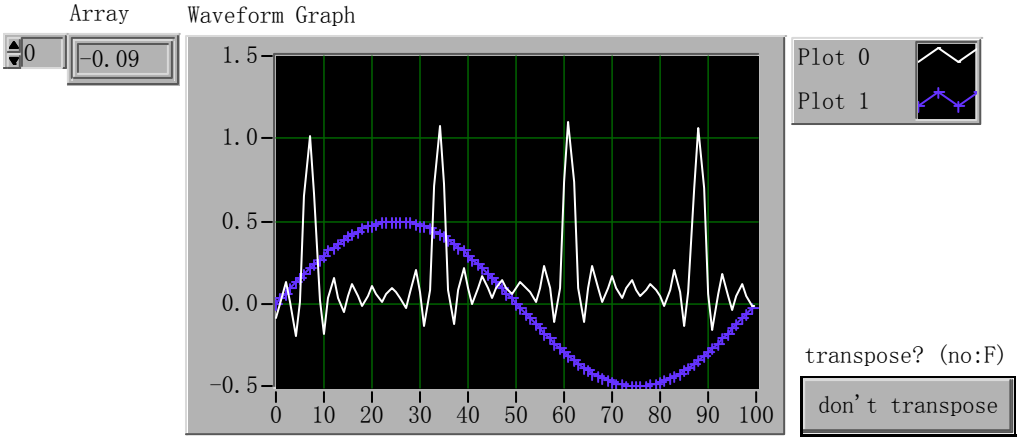
	A	B	C
1	0	0.4258	
2	1	0.3073	
3	2	0.9453	
4	3	0.964	
5	4	0.9517	
6			

练习 4－3 将数据写入电子表格文件

目的：修改一个已有的 VI 以使用文件 I/O 功能函数，以便可以将数据以 ASCII 格式保存到一个新的文件。然后就可以用一个电子表格程序打开该文件。

前面板

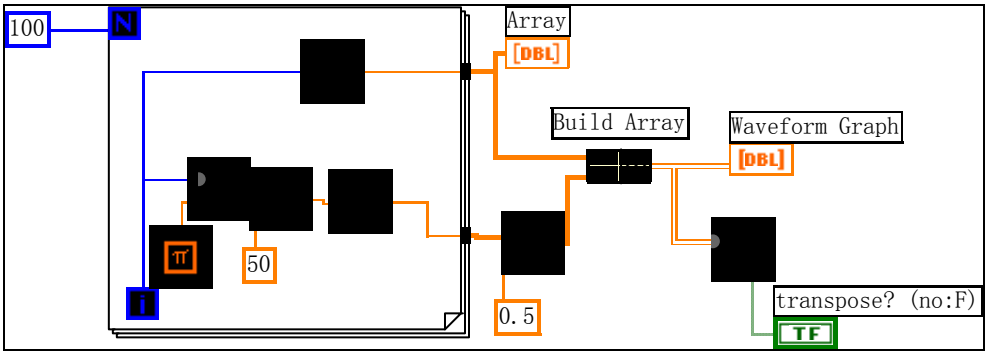
打开前面练习中创建的 Graph Waveform Arrays.vi。当调用这个 VI 时，该 VI 将产生两个数据数组，并将它们绘制在一个图区中。您需要对该 VI 进行修改，从而把两个数组写入一个文件，格式是每列含有一个数组。



流程图

打开 Graph Waveform Arrays.vi 的流程图，按照下图在流程图的右下角添加功能函数。

- Write To Spreadsheet File VI （Functions»File I/O）用于将二维数组转换成电子表格字符串，再将它写入一个文件。如果没有指定路径名称，将会弹出一个文件对话框，提示输入文件名。该 VI 将把一维或者二维数组写入文件。这个例子中，因为用的是二维数组，所以无需连接一维输入端子。



- Boolean 常数 （Functions»Boolean）用于控制是否在写入数据之前转换成二维数组。在这个例子中需要对数据进行转换。因为电子表格文件的每列都含有一个数据数组，所以必须先转换成二维数组。
- 返回前面板，执行该 VI。数据数组产生以后，会出现一个文件对话框提示输入新建文件的文件名。输入文件名，并单击 OK。
- 可以尝试选择转换与不转换两种情况运行程序，查看结果差别。
- 保存该 VI 为 Waveform Arrays to File.vi，并关闭该 VI。

现在可以用电子表格软件或者文本编辑器打开或者编辑刚才创建的文件。可以看到两列表格，每列含有 100 个元素。

在这个例子中，直到所有数组都被采集以后，数据才可以被转换或者写入文件。如果需要更大的数据缓存和希望在数据产生后把它们写入到硬盘，就需要使用另外一个文件 I/O VI。

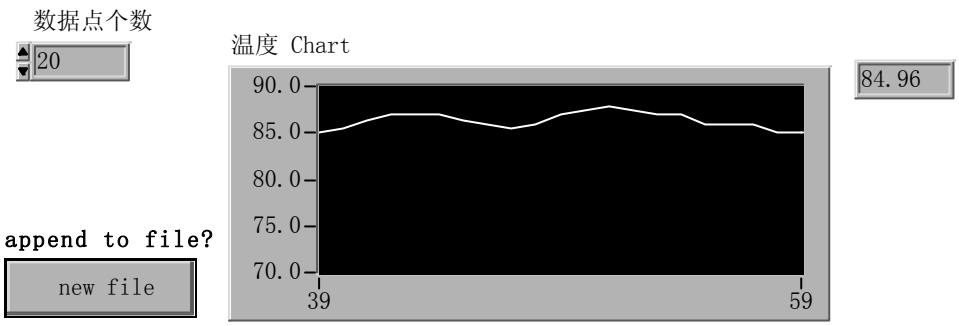
练习 4－3 结束。

练习 4－4 向文件添加数据

目的：创建一个 VI，可以把温度数据以 ASCII 格式添加到某个文件中。该 VI 使用 For 循环产生温度数据，并将它们存储到一个文件中。在每个循环期间，您都要把数据转换成字符串，添加一个逗号作为分隔符，并将字符串添加到文件中。

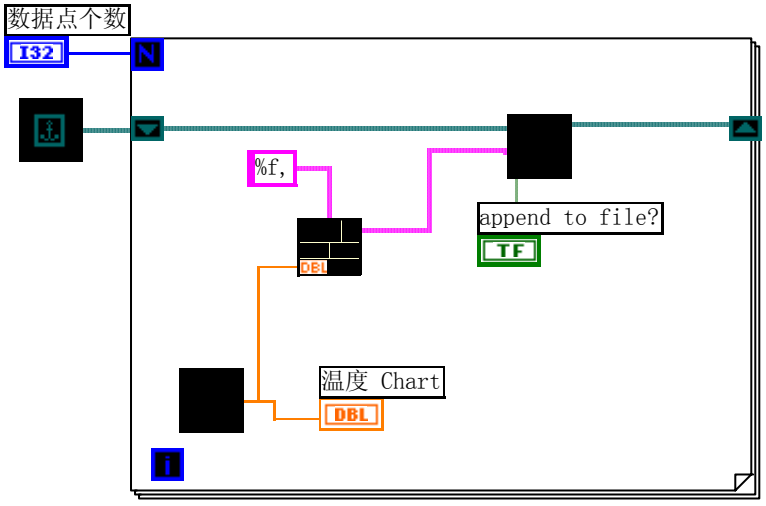
前面板

打开一个新的前面板，并按照下图放置对象。



前面板中包括一个数字式显示器和一个波形图。“数据点个数”控制对象指定了需要采集和写入文件的温度数据的数量。波形图表则用于显示温度曲线。将波形图表的 Y 轴范围设置为 70.0 到 90.0，X 轴范围设置为 0 到 20。

流程图



- 打开流程图，添加 For 循环并增大它的面积。该 VI 将产生由“数据点个数”控制对象指定的个数的温度数据。
- 在循环中加一个移位寄存器，方法是用鼠标右键单击循环边界，在快捷菜单中选择移位寄存器。这个移位寄存器中将含有文件的路径名。
- 完成对象的连线。
- Empty Path 常数（**Functions»File I\O»File Constants**）——用于初始化移位寄存器，以保证需要对文件写入数据时路径都是空的。会出现一个文件对话框提示输入文件名。

- Digital Thermometer VI（**Functions»Select a VI…**）——返回一个模拟温度测量值（仿真）。

- Format Into String 函数（**Functions»String**）——将温度数据转换成字符串，并且在数据后面增加一个逗号。

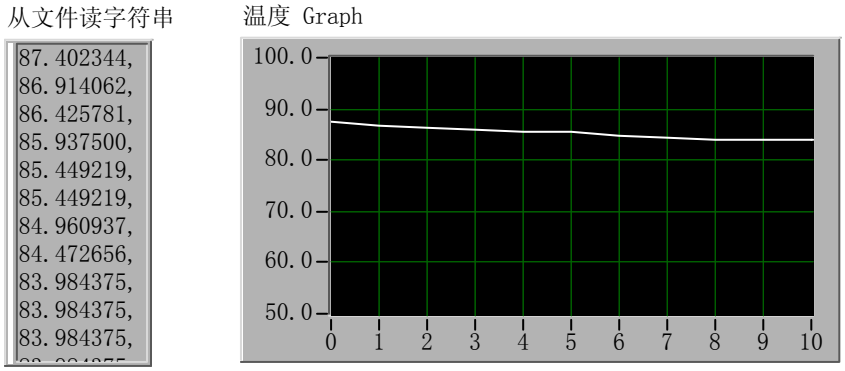
- Write Characters To File VI（**(Functions»File I/O)**——用于向文件写入字符串。
- Boolean 常数（**Functions»Boolean**）用于将 Write Characters To File VI 的 append to file?输入为 TRUE，这样在循环执行时新的温度数据就会加入到选中的文件中。用操作工具单击这个常数可以将它设置为 TRUE。
- 返回前面板，把“数据点个数”设置为 20，执行该 VI。这时会出现一个文件对话框，提示输入文件名。输入文件名以后，VI 就会在每个温度数据产生时，将它写入到该文件中。
- 把该 VI 保存为 LabVIEW\Activity 目录下的 Write Temperature to File.vi。
- 使用任意一个字处理软件，例如 Write for Windows，Teach Text for Macintosh， 或者 UNIX 平台下的某个文本编辑器，打开该数据文件查看其内容。您可以看到文件的内容是 20 个用逗号分隔开的数值（准确到小数点后三位）。

练习 4－4 结束。

练习 4－5 从文件读取数据

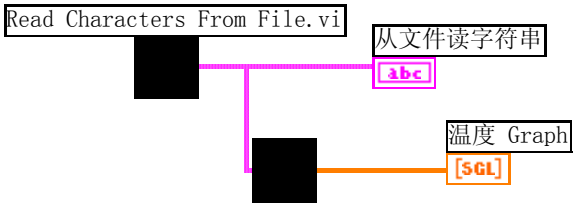
目的：创建一个 VI，可以从上一个练习中创建的例子中读取数据，并把这些数据显示在一个波形图形中。必须按照数据保存的格式来读取它，因为原来是用字符串数据类型，把数据

保存为 ASCII 格式，那么就必须用一个文件 I/O 函数把数据作为字符串读出。
前面板
打开一个新的前面板，并按照下图放置对象。



前面板中包括一个字符串显示对象和一个波形图。“从文件读字符串“显示对象将从上个练习创建的文件中读出用逗号分隔开的温度数据。波形图则用于显示温度曲线。

流程图



- Read Characters From File VI（**Functions»File I/O**）——用于从文件中读取数据，以及输入字符串中的信息。如果没有指定路径名称，将出现一个文件对话框提示您输入文件名。在这个例子中，无需判断需要读取的字符的个数，因为文件的字符数比默认的 512 要少。要从文件中读取数据，必须知道数据的存储方式。如果知道了文件的长度，就可以使用 Read Characters From File VI 读取指定个数的字符。

- Extract Numbers VI (Examples\General\strings.llb)——用于提取由逗号、分行符号、非数值字符等分隔开的数据组成的 ASCII 字符串，并将它们转换成数值数组。

- 返回前面板，执行该 VI。将出现一个文件对话框，在其中选择刚才保存的数据文件，您可以看到图形中显示的数据与 Write Temperature to File VI 例子中显示的一样。
- 保存该 VI 为 Temperature from File.vi，并关闭它。

练习 4－5 结束。

4．3 数据记录文件（datalog file）

上面提供的例子说明了处理 ASCII 字符格式存储的数据的文件的简单方法。在需要创建供其他软件（如电子表格软件）访问的文件时这种方法很有用。另外一种是称为数据记录文

件（datalog file）的数据格式。它与数据库文件有类似之处，文件是由记录组成的，一个文件的所有记录有相同的结构和长度。访问该文件是可以以记录为单位，并且可直接访问文件中的任意一个记录。记录本身的数据结构可由用户自己定义，一个记录内可容纳不同的数据类型，它就像一个簇一样。

如果要用 VI 获得数据，您可能不想把数据写入到 ASCII 文件中，因为把数据和字符串之间相互转换非常花费时间。例如，把一个二维的字符转换成一个具有电子表格格式的字符串（具有标题和时间标记）也是一个非常复杂的操作。如果不需要把文件存储成可供别的软件访问的格式，您可以把数据输出到一个数据记录文件。使用这种格式时，把数据写入到文件的操作变得非常简单，这也使得读写操作的速度更快。它还可以简化数据采集的工作，因为您可以把初始的数据块作为一个日志或者记录读取，而无需了解其中含有多少数据。G 语言会记录数据的数量，用于对每个数据记录文件的记录。

Write Datalog File 示例（位于 Examples\File\datalog.llb）创建了一个新的数据记录文件，并把指定数目的数据写入该文件。每个记录都是一个由一个字符串和一个单精度数据数组构成的簇。

要读取一个数据记录文件，您采用的格式必须与对该文件写入数据时所用格式相同。Read Datalog File 示例将从 Write Datalog File 示例创建的数据记录文件中一次读取一个记录。读出的每个记录都是一个由一个字符串和一个单精度数据数组构成的簇。

第六章　数据采集

６．１　概述

在计算机广泛应用的今天，数据采集的重要性

６．１．１　采样定理与抗混叠滤波器

６．１．２　数据采集系统的构成

６．１．３　模入信号类型与连接方式

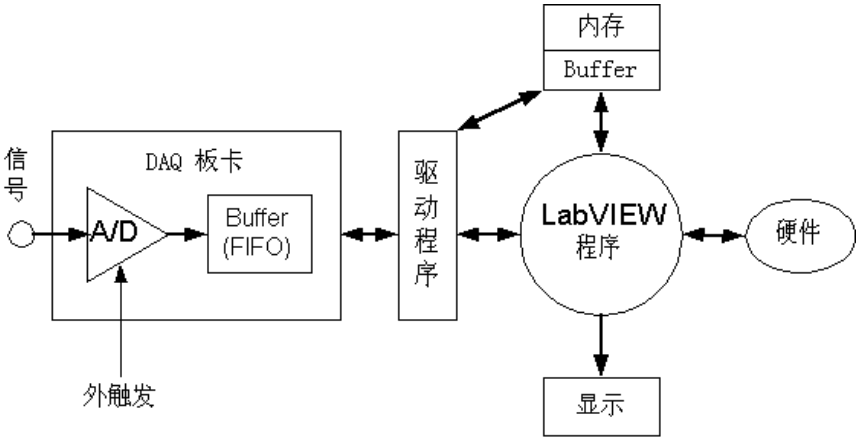
６．１．４　信号调理

数据采集问题的复杂程度评估

概念与名词

- **AC**: Alternating Current，交流电流，泛指交流信号。
- **DC**: Direct Current，直流电流，泛指直流信号。
- **ADC**: Analog-to-Digital Conversion，模数变换，有时也表示为 A/D。
- **DAQ**: Data Acquisition，数据采集。
- **DMA**: Direct Memory Access，直接内存访问。它允许将采集的数据直接送给计算机的内存，数据传输速率较高。
- **GPIB**: General Purpose Interface Bus，也称为 IEEE 488.2 总线。它是一种应用最广泛的仪器总线。
- **SCXI**: Signal Conditioning extensions for Instrumention,信号调理器。
- **VISA**: Virtual Instrument Standard Architecture，虚拟仪器软件体系结构。它是控制 GPIB、VXI、RS-232 和其他类型仪器的接口库。

５．２ 数据采集结构



上图表示了数据采集的结构。在数据采集之前，程序将对 DAQ 板卡初始化，板卡上和内存中的 Buffer 是数据采集存储的中间环节。需要注意的两个问题是：是否使用 Buffer？是否使用外触发启动、停止或同步一个操作。

５．２．１　缓冲（Buffers）

这里的缓冲指的是 PC 内存的一个区域（不是 DAQ 卡上的 FIFO 缓冲），它用来临时存放数据。例如，你需要采集每秒采集几千个数据，在一秒内显示或图形化所有数据是困难的。

但是将采集卡的数据先送到 Buffer，你就可以先将它们快速存储起来，稍后再重新找回它们显示或分析。需要注意的是 Buffer 与 DAQ 操作的速度及容量有关。如果你的卡有 DMA 性能，模拟输入操作就有一个通向计算机内存的高速硬件通道，这就意味着所采集的数据可以直接送到计算机的内存。

不使用 Buffer 意味着对所采集的每一个数据你都必须及时处理（图形化、分析等），因为这里没有一个场合可以保持你着手处理的数据之前的若干数据点。

下列情况需要使用 Buffer I/O：

- 需要采集或产生许多样本，其速率超过了实际显示、存储到硬件，或实时分析的速度。
- 需要连续采集或产生 AC 数据（>10 样本 / 秒），并且要同时分析或显示某些数据。
- 采样周期必须准确、均匀地通过数据样本。

下列情况可以不使用 Buffer I/O：

- 数据组短小，例如每秒只从两个通道之一采集一个数据点。
- 需要缩减存储器的开支。

5．2．2　触发（Triggering）

触发涉及初始化、终止或同步 DAQ 事件的任何方法。触发器通常是一个数字或模拟信号，其状态可确定动作的发生。软件触发最容易，你可以直接用软件，例如使用布尔面板控制去启动/停止数据采集。硬件触发让板卡上的电路管理触发器，控制了 DAQ 事件的时间分配，有很高的精确度。硬件触发可进一步分为外部触发和内部触发。当某一模入通道发生一个指定的电压电平时，让卡输出一个数字脉冲，这是内部触发的例子。采集卡等待一个外部仪器发出的数字脉冲到来后初始化采集卡，这是外部触发的例子。许多仪器提供数字输出（常称为“trigger out”）用于触发特定的装置或仪器，在这里，就是 DAQ 卡。

下列情况使用软件触发：

- 用户需要对所有 DAQ 操作有明确的控制，并且
- 事件定时不需要非常准确。

下列情况使用硬件触发：

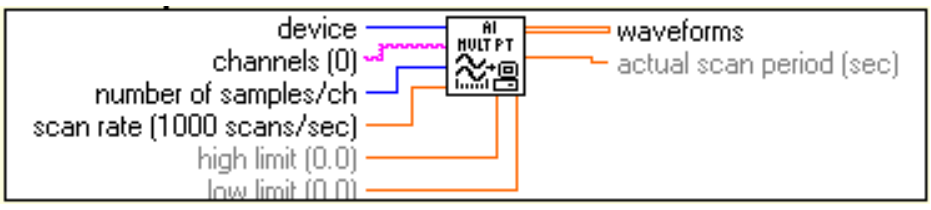
- DAQ 事件定时需要非常准确。
- 用户需要削减软件开支。
- DAQ 事件需要与外部装置同步。

下面我们可以看到怎样使用 DAQ 的 VI 程序设置有 Buffer 及无 Buffer 的 I/O 操作，以及设置触发的类型。

5．3　模拟 I/O（Analog I/O）

5．3．1　基本概念

在研究 DAQ VI 之前需要了解如下的几个定义。我们以下图的 AI Acquire Waveform.vi 为例说明。



device——设备号。在 NI DAQ 设置工具中设定。该参数告诉 LabVIEW 你使用什么卡，它可以使 DAQ VI 自身独立于卡的类型，也就是说，如果你稍后使用了另一种卡，并且赋予它同样的设备号，你的 VI 程序可正常工作而无须修改。

samples——表示一个 A/D 转换：它是一个点、一个与测量发生时的实际模拟量对应的数据。

channel——指定数据样本的物理源。例如，一个卡有 1 6 个模拟输入通道，你就可以同时采集 1 6 组数据点。在 LabVIEW VI 中，一个通道或一组通道都用一个字符串来指定。例如：

通道	通道串
通道 5	5
通道 0 到 4	0:4
通道 1，8，以及 10 到 13	1,8,10:13

scan——是在多通道采样时，由一个单独的通道得到的样本。

waveform——是由一个通道得到的一组样本，采集若干周期或一定时间。通常，但不是必须，数据点之间的时间对给定的 waveform 是一个常数。

初学者经常对 scan 和 waveform 感到迷惑。scan 是相对于若干通道的一组样本（对每一个通道的一个瞬时得到一个样本），waveform 是相对于时间的一组样本（由一个通道得到）。

high limit 和 low limit——你期望的对信号的限制。输入信号变化的缺省值是 10V 到-10V，可以设置 DAQ 系统的增益，例如，对大多数卡，如果你将其设为 5 到-5V，则增益为 2。如果你将其设为 1 到-1V，则增益为 10。所以如果你知道你的输入信号与缺省值不同，可以改写这些值。可以用下面的公式确当所使用的增益：

增益＝采集卡输入范围 / （|High Limit|-|Low limit|）

许多采集卡只支持某些预先确定的增益值。如果你设置一个理论上的增益是得不到支持的，LabVIEW 会自动将其调整到最近的预置值。典型的采集卡所支持的增益值有 0.5,1,2,5,10,20,50,100。

taskID——一个 3 2 位的整数（I32 类型）。某些 DAQ VI 用来识别指定的加于其上的 I/O 操作。许多 DAQ VI 需要接受一个 taskID in 并且返回一个 taskID out 给下一个 VI。你不能通过 taskID 给每一个 VI 提供采集卡的信息，如采样率、电压限制等。但一个初始化的 VI 可以把这些信息传给 taskID out，它可以信息告诉别的需要设置的 VI。后面将有例子说明。

5．3．2 简单 Analog I/O

这是 LabVIEW 提供的一组标准的、简单易用的 DAQ VI。

◆ Analog Input

从左到右， 4 个 VI 的功能为：

- 从指定通道获得一个样本。
- 从由通道字符串规定的一组通道每通道获得一个样本。这些样本返回到一个样本数组，顺序由通道号决定。

- 按指定的采样率由一个通道得到一个波形（一组覆盖一个周期的样本），这些样本返回到一个 wareform 数组。
- 从由通道字符串规定的每个通道获得一个波形。这些样本返回到一个波形的 2 维数组，顺序由通道号和采样周期决定。通道数据的每个点占 1 列，时间增量由行决定。

◆ Analog Output

从左到右， 4 个 VI 的功能为：

- 在指定输出通道设置一个规定电压。
- 在指定输出通道设置一组规定电压。这些电压在输出通道一直保持恒定，直到其自身改变或装置复为位。
- 在指定输出通道产生一个波形，波形的点（以电压为单位）是预先由波形数组提供的。更新速率（Update rate）规定了两个点之间的时间。
- 与上类似，多波形，每通道一个，可同时发生。每个波形存放在 2 维数组的 1 列。

练习 5 － 1

目的：采集一个直流电压信号

1．准备一个直流电源（例如 0.5V）作为信号源连接到 DAQ 卡的 0 通道模入端。

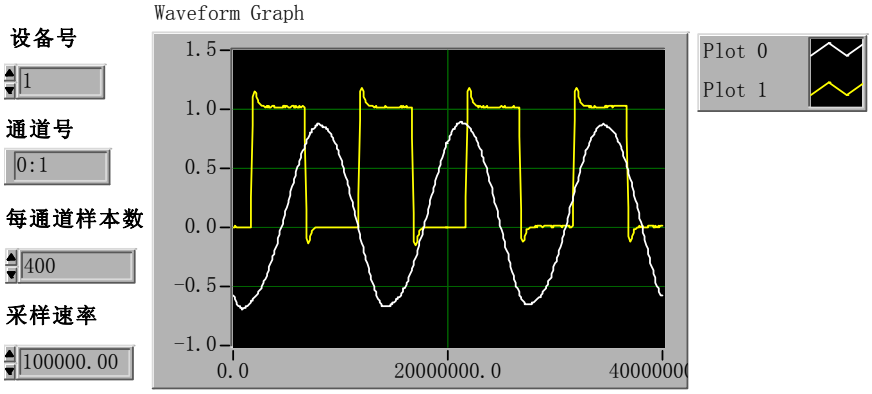
2．构造前面板和框图如上面所示。

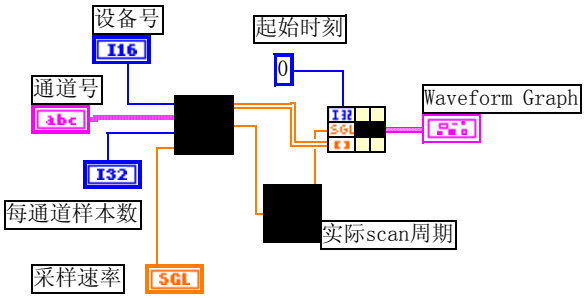
3．运行程序。可得到 Meter 指示 0.5V。

练习 5 － 1 结束

练习 5 － 2

目的：多通道数据采集



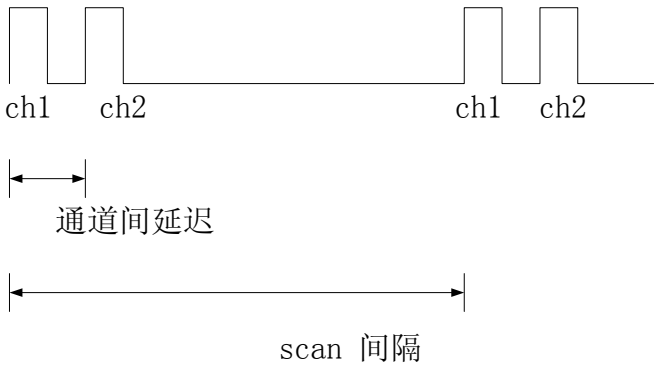


1．准备一个方波信号源和一个正弦波信号源。分别连接到模入通道 0 和 1 。
2．设置前面板与框图如上。
3．在前面板上的 graph 上用快捷菜单选择 **Transpose Array**。因为 **AI Sample Channel** 功能返回的 2D 数组是每列反映一个通道的电压，而在一般情况下 graph 图形是行对列，所以需要对数组做一次转置，以使 Y 轴表示电压值。
4．在框图上将起始时刻、实际采样周期和采样数据使用 **Bundle** 功能捆绑成一个数组，送给了 graph。注意捆绑的顺序是 I32、SGL 和数组，如果变为 I32、数组和 SGL，可能出错。
5．设置 scan 速率、通道号、每通道样本数如前面板所示。
6．运行该程序。
7．保存为 Acquire Multiple Channels.vi。
8．该程序是无缓冲、软件触发的。

练习 5 － 2　结束

多通道 I/O 的一个重要限制必须说明。如果对它设置一个高 scan 速率，并且从每个通道实时观察数据（不是由数组观察），你或许会看到通道之间的相位延迟。这是为什么？大多数卡在一个时刻只能执行一个 A/D 变换。这就是为什么它被称为扫描：输入端口的数据被依次数字化，每时刻一个通道。延迟，或通道间的延迟，发生在每个通道样本之间。默认的情况是通道间的延迟足够小，但是它与卡关系密切。

对直流或低频信号，这个相位延迟一般不是问题。通道间的延迟远小于扫描周期，看起来每个通道的采样是同时的。例如，通道间的延迟是微秒级的，而采样速率是 1 scan/sec。但是对高频信号，延迟就必须引起注意，如果你关心信号的同步，测量就可能出现问题。



5．3．3　中级 Analog I/O

上面介绍的简单 Analog I/O 的基本局限是执行 DAQ 任务的庸余。例如，你每一次调用

AI Sample Channel，都必须为特定类型的测量设置硬件，告诉它采样率等。显然，如果你经常要采集大量的样本，你未必需要在每一次重复时都去设置测量。

中级 Analog I/O 有更好的功能与灵活性，可以更有效地开发你的应用。它的特点包括控制内部采样率，使用外部触发，执行连续外部触发等。下面我们将仔细描述它的各种 VI，你应该注意其大量输入、输出端子中的部分内容。有效地使用这些 VI 只需要关注你需要的端子。在大多数情况下，你不需要为在 help 中解释的端子的选项烦恼。

◆ Analog Input

AI Config 对指定的通道设置模入操作，包括硬件、计算机内 buffer 的分配。常用的端子有：

- Device——采集卡的设备号。
- Channel——指定模入通道号的串数组。
- Intput limit——指定输入信号的范围达到调节硬件增益的目的。
- Buffer size——单位是 scan，控制用于采集数据的 AI Config 占用计算机内存的大小。
- Interchannel delay——对扫描间隔设置通道间的偏差。

AI Start 启动带缓冲的模入操作。它控制数据采集速率，采集点的数目，及使用任何硬件触发的选择。它的两个重要输入是：

- Scan rate(scan/sec)——对每个通道采集的每秒扫描次数。
- Number of scans to acquire——对通道列表的扫描次数。？？

AI Read——从被 AI Config 分配的缓冲读取数据。它能够控制由缓冲读取的点数，读取数据在缓冲中的位置，以及是否返回二进制数或标度的电压数。它的输出是一个 2 维数组，其中每一列数据对应于通道列表中的一个通道。

AI Single Scan——返回一个扫描数据。它的电压数据输出是由通道列表中的每个通道读出的电压数据。使用这个 VI 仅与 AI Config 有关联，不需要 AI Start 和 AI Read。

AI Clear——清除模入操作、计算机中分配的缓冲、释放所有 DAQ 卡的资源，例如计数器。

当你设置一个模入应用时，首先使用的 VI 总是 AI Config。AI Config 会产生一个 taskID 和 Error cluster（出错信息簇）。所有别的模入 VI 接受这个 taskID 以识别操作的设备和通道，并且在操作完成后输出一个 taskID。因为 taskID 是一个输入并向另一个模入 VI 输出，所以该参数形成了 DAQ VI 之间的一个关联数据。

AO Config 对指定的通道设置模出操作，包括硬件、计算机内 buffer 的分配。常用的端子有：

- Device——采集卡的设备号。

- Channel——指定模出通道号的串数组。
- Limit settings——指定输出信号的范围。
- taskID——用于所有后来的模出 VI 以规定操作的设备和通道。

AO Write 以电压数据的方式写数据到模出数据缓冲区。它是一个 2 维数组，其中每一列数据对应于通道列表中的一个通道。

AO Start 启动带缓冲的模出操作。Update rate(scan/sec)是每秒发生的更新数的个数。如果你将 0 写入 Number of buffer iteerations 端子，则卡将连续输出给缓冲，直到运行 AO Clear 功能。

AO Wait 在返回之前一直等待直到波形发生任务完成。它的电压数据输出是由通道列表中的每个通道读出的电压数据。使用这个 VI 仅与 AO Config 有关联，不需要 AO Start 和 AO Read。

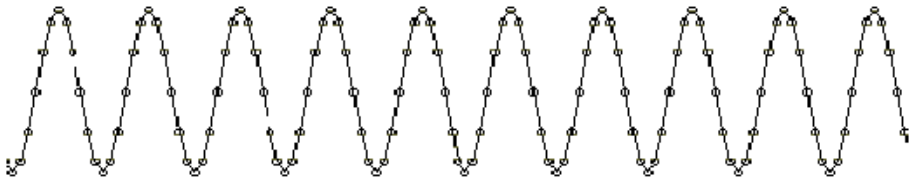
AO Clear——清除模出操作、计算机中分配的缓冲、释放所有 DAQ 卡的资源，例如计数器。

当你设置一个模出应用时，首先使用的 VI 总是 AO Config。AO Config 会产生一个 taskID 和 Error cluster（出错信息簇）。所有别的模出 VI 接受这个 taskID 以识别操作的设备和通道，并且在操作完成后输出一个 taskID。因为 taskID 是一个输出并向另一个模出 VI 输出，所以该参数形成了 DAQ VI 之间的一个关联数据。

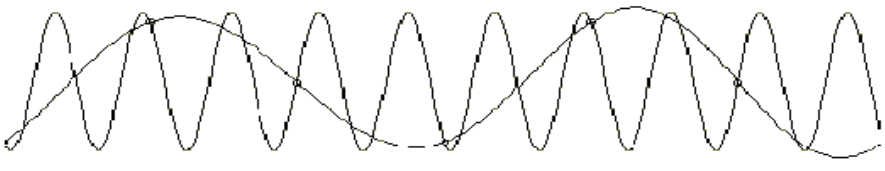
５．４ 采样注意事项

５．４．１ 采样频率的选择

对输入信号的采样率是最重要的参数之一。采样率决定了模数转换（A/D）的频率。较高的采样率意味着在给定时间内采集更多的点，所以可以更好地还原原始信号。而采样率过低则可能会导致信号畸变。下图显示了一个信号分别用充分的采样率和过低的采样率进行采样的结果。采样率过低的结果是还原信号的频率看上去与原始信号不同。这种信号畸变叫做混频（alias）。



充分采样率时的信号



过低采样率的采样结果

根据奈奎斯特定理，为了防止发生混频，最低采样频率必须是信号频率的两倍。对于某个给定的采样率，能够正确显示信号而不发生畸变的最大频率叫做奈奎斯特频率，它是采样频率的一半。如果信号频率高于奈奎斯特频率，信号将在直流和奈奎斯特频率之间畸变。混

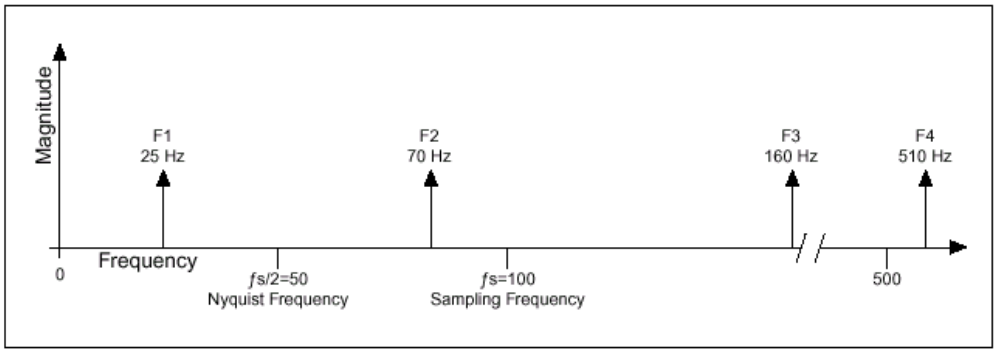
频偏差（alias frequency）是输入信号的频率和最靠近的采样率整数倍的差的绝对值。下图显示了这种现象。假设采样频率 fs 是 100HZ,，再假设输入信号还含有频率为 25 Hz，70 Hz，160 Hz 和 510 Hz 的成分，采样的结果会怎样呢？低于奈奎斯特频率（fs/2=50 Hz）的信号可以被正确采样。而频率高于奈奎斯特的信号采样时会发生畸变。例如，F1（25 Hz）显示正确，而在分别位于 30 Hz、40 Hz 和 10 Hz 的 F2、F3 和 F4 都发生了频率畸变。计算混频偏差时需要用到下面这个等式：

混频偏差＝ABS（采样频率的最近整数倍－输入频率），其中 ABS 表示“绝对值”，例如：

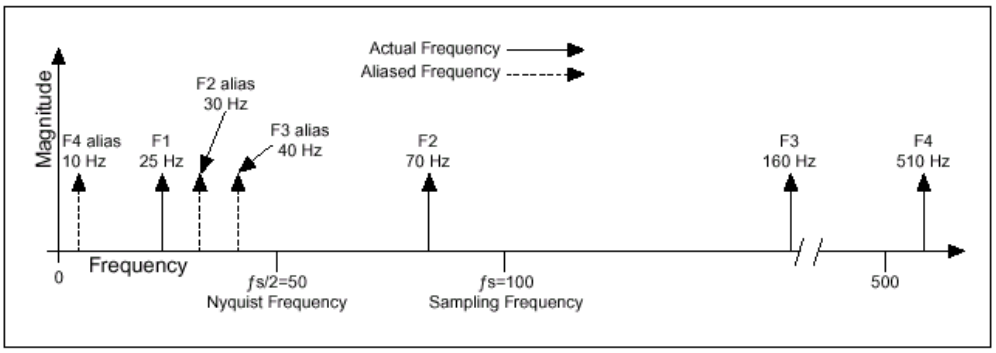
混频偏差 F2 = |100 - 70| = 30 Hz

混频偏差 F3 = |2*100 - 160| = 40 Hz

混频偏差 F4 = |5*100 - 510| = 10 Hz

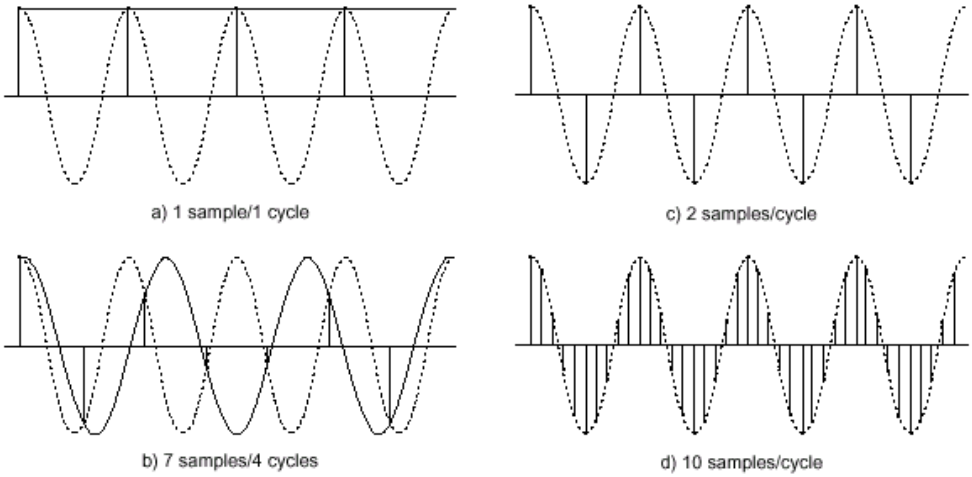


实际信号的频率组成



采样后信号的频率组成和混频偏差

采样率应当设成多少？您可能会首先考虑用 DAQ 板支持的最大频率。但是，长期使用很高的采样率可能会导致没有足够的内存或者硬盘存储数据。下图显示了采用不同的采样频率的效果。在例 a 中，对一个频率为 f 的正弦波形进行采样，每秒采样数与每秒周期数相等，也就是一个周期采样一次，还原的波形出现了畸变，成了一个直流信号。如果把采样率增大到每个周期采样四次，如例 b 所示，波形的频率提高了，频率畸变比原始信号要小（3 个周期）。例 b 中的采样率是 7/4f。如果把采样率增加到 2f，那么转换后的波形具有正确的频率（与周期数相同），并可以还原成原始波形，如例 c 所示。对于时域下的处理，可能需要您提高采样率以接近于原始信号。通过把采样率提高到足够大，例如 fs=10f，或者每周期采样 10 次，就可以正确地复原波形，如例 d 所示。



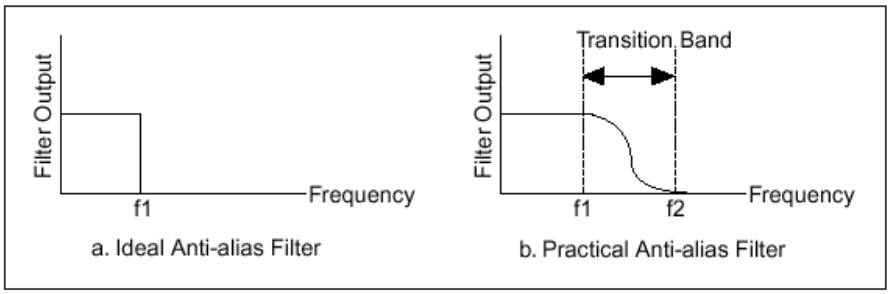
各种采样率的效果

５．４．２　使用抗混频滤波器

根据上面的讨论我们已经了解到，采样率必须大于被采样信号的频率的两倍。换句话说，信号的最高稳定频率必须小于或者等于采样频率的一半。但是在实际应用中，怎样才能保证这一点呢？即使已经确定被测的信号有一个最大的频率值，杂散信号（例如来自于输电线路或者当地广播电台的干扰）可能会带来比奈奎斯特频率高的频率。这些频率很可能会混杂在需要的频率范围中，导致错误的结果。

为了保证输入信号的频率全部在给定范围内，需要在采样器和 ADC 之间安装一个低通滤波器（可以通过低频信号但是削弱高频信号的滤波器）。因为它通过对高频信号（高于奈奎斯特信号频率）进行削弱，减少了混频信号的干扰，所以这个滤波器被称为抗混频滤波器。这个阶段数据仍然处于模拟状态，所以抗混频滤波器是一个模拟滤波器。

一个理想抗混频滤波器如下图所示：



抗混频滤波器

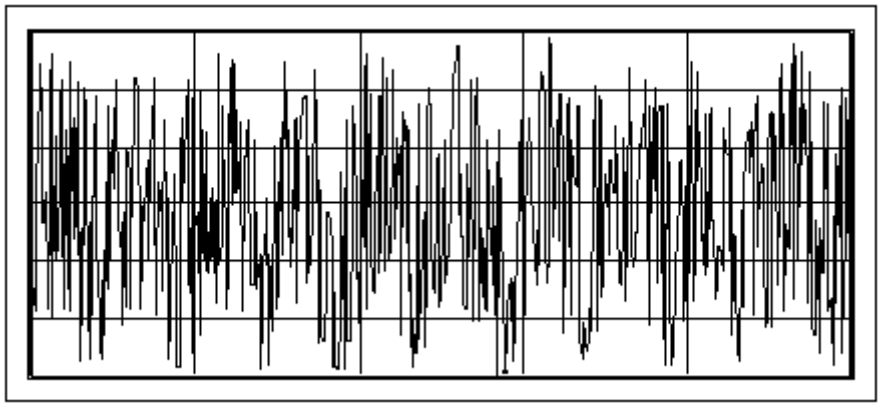
它通过了所有需要的输入频率（低于 f1），并过滤了所有不需要的频率（高于 f1）。但是，这样的滤波器实际上并不可能实现。实际应用中的抗混频滤波器如图 b 所示。它们通过所有低于 f1 的频率，并过滤所有高于 f2 的频率。f1 和 f2 之间的区域被称为过渡带（transition band），其中输入信号逐步减弱。尽管您只希望通过所有频率低于 f1 的信号，但是过渡带中的信号仍然可能会导致混频。所以，在实际应用中，采样频率应当大于过渡带的最高频率的两倍。因而采样频率就将比输入频率的两倍还要大。这是采用频率大于输入频率最大值的两倍的原因之一。

第六章 信号处理与分析

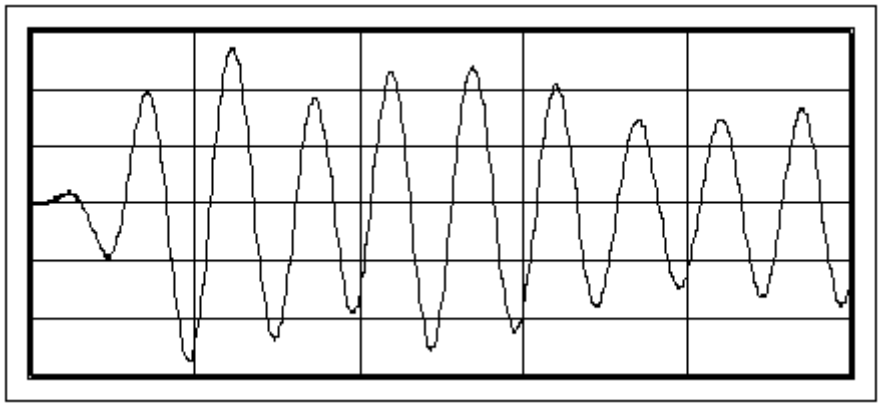
6．1 概述

数字信号在我们周围无所不在。因为数字信号具有高保真、低噪声和便于信号处理的优点，所以得到了广泛的应用，例如电话公司使用数字信号传输语音，广播、电视和高保真音响系统也都在逐渐数字化。太空中的卫星将测得数据以数字信号的形式发送到地面接收站。对遥远星球和外部空间拍摄的照片也是采用数字方法处理，去除干扰，获得有用的信息。经济数据、人口普查结果、股票市场价格都可以采用数字信号的形式获得。因为数字信号处理具有这么多优点，在用计算机对模拟信号进行处理之前也常把它们先转换成数字信号。本章将介绍数字信号处理的基本知识，并介绍由上百个数字信号处理和分析的 VI 构成的 LabVIEW 分析软件库。

目前，对于实时分析系统，高速浮点运算和数字信号处理已经变得越来越重要。这些系统被广泛应用到生物医学数据处理、语音识别、数字音频和图像处理等各种领域。数据分析的重要性在于，无法从刚刚采集的数据立刻得到有用的信息，如下图所示。必须消除噪音干扰、纠正设备故障而破坏的数据，或者补偿环境影响，如温度和湿度等。



通过分析和处理数字信号，可以从噪声中分离出有用的信息，并用比原始数据更全面的表格显示这些信息。下图显示的是经过处理的数据曲线。



用于测量的虚拟仪器(VI)

用于测量的虚拟仪器(VI)执行的典型的测量任务有：

- 计算信号中存在的总的谐波失真。
- 决定系统的脉冲响应或传递函数。
- 估计系统的动态响应参数，例如上升时间、超调量等等。
- 计算信号的幅频特性和相频特性。
- 估计信号中含有的交流成分和直流成分。

在过去，这些计算工作需要通过特定的实验工作台来进行，而用于测量的虚拟仪器可以使这些测量工作通过 LabVIEW 程序语言在台式机上进行。这些用于测量的虚拟仪器是建立在数据采集和数字信号处理的基础之上，有如下的特性：

- 输入的时域信号被假定为实数值。
- 输出数据中包含大小、相位，并且用合适的单位进行了刻度，可用来直接进行图形的绘制。
- 计算出来的频谱是单边的（single_sided），范围从直流分量到 Nyquist 频率(二分之一取样频率)。（即没有负频率出现）
- 需要时可以使用窗函数，窗是经过刻度地，因此每个窗提供相同的频谱幅度峰值，可以精确地限制信号的幅值。

一般情况下，可以将数据采集 VI 的输出直接连接到测量 VI 的输入端。测量 VI 的输出又可以连接到绘图 VI 以得到可视的显示。

有些测量 VI 用来进行时域到频域的转换，例如计算幅频特性和相频特性、功率谱、网路的传递函数等等。另一些测量 VI 可以刻度时域窗和对功率和频率进行估算。

本章我们将介绍测量 VI 中常用的一些数字信号处理函数。

LabVIEW 的流程图编程方法和分析 VI 库的扩展工具箱使得分析软件的开发变得更加简单。LabVIEW 分析 VI 通过一些可以互相连接的 VI，提供了最先进的数据分析技术。你不必像在普通编程语言中那样关心分析步骤的具体细节，而可以集中注意力解决信号处理与分析方面的问题。LabVIEW 6i 版本中，有两个子模板涉及信号处理和数学，分别是 **Analyze** 子模板和 **Methematics** 子模板。这里主要涉及前者。

进入 **Functions** 模板 **Analyze》Signal Processing** 子模板。

1	2	3
4	5	6

其中共有 6 个分析 VI 库。其中包括：

①. Signal Generation（信号发生）：用于产生数字特性曲线和波形。

②. Time Domain（时域分析）：用于进行频域转换、频域分析等。

③. Frequency Domain（频域分析）：

④. Measurement（测量函数）：用于执行各种测量功能，例如单边 FFT、频谱、比例加窗以及泄漏频谱、能量的估算。

⑤. Digital Filters（数字滤波器）：用于执行 IIR、FIR 和非线性滤波功能。

⑥．Windowing（窗函数）：用于对数据加窗。

在后面几节中，你将学习如何使用分析库中的 VI 创建函数发生器和简单实用的频谱分析仪，如何使用数字滤波器，窗函数的作用以及不同类型窗函数的优点，怎样执行简单的曲线拟合功能，以及其他一些内容。可以在 `labview\examples\analysis` 目录中找到一些演示程序。

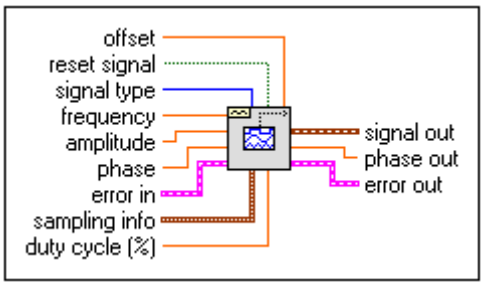
6．2　信号的产生

本节将介绍怎样产生标准频率的信号，以及怎样创建模拟函数发生器。参考例子见 examples\analysis\sigxmpl.llb。

你还将学习怎样使用分析库中的信号发生 VI 产生各种类型的信号。信号产生的应用主要有：

- 当无法获得实际信号时，（例如没有 DAQ 板卡来获得实际信号或者受限制无法访问实际信号），信号发生功能可以产生模拟信号测试程序。
- 产生用于 D/A 转换的信号

在 LabVIEW 6i 中提供了波形函数，为制作函数发生器提供了方便。以 Waveform>>Waveform Generation 中的基本函数发生器（Basic Function Generator.vi）为例，其图标如下：



其功能是建立一个输出波形，该波形类型有：正弦波、三角波、锯齿波和方波。这个 VI 会记住产生的前一波形的时间标志并且由此点开始使时间标志连续增长。它的输入参数有波形类型、样本数、起始相位、波形频率（单位：Hz）

参数说明：

offset：波形的直流偏移量，缺省值为 0.0。数据类型 DBL

reset signal：将波形相位重置为相位控制值且将时间标志置为 0。缺省值为 FALSE.

signal type：产生的波形的类型，缺省值为正弦波。

frequency ：波形频率（单位 Hz），缺省值为 10。

amplitude ：波形幅值，也称为峰值电压，缺省值为 1.0。

phase ：波形的初始相位（单位 度）缺省值为 0.0.

error in ：在该 VI 运行之前描述错误环境。缺省值为 no error. 如果一个错误已经发生，该 VI 在 error out 端返回错误代码。该 VI 仅在无错误时正常运行。 错误簇包含如下参数。

status ：缺省值为 FALSE，发生错误时变为 TRUE。

code ：错误代码，缺省值为 0。

source ：在大多数情况下是产生错误的 VI 或函数的名称，缺省值为一个空串。

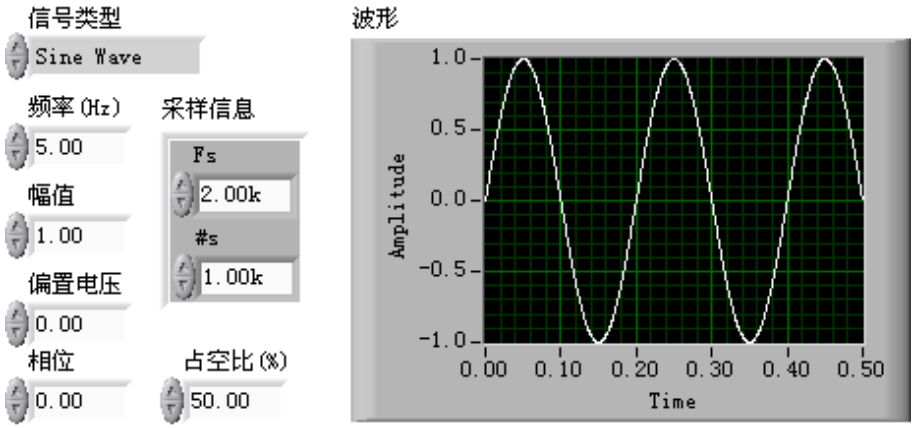
sampling info ：一个包括采样信息的簇。共有 Fs 和#s 两个参数。

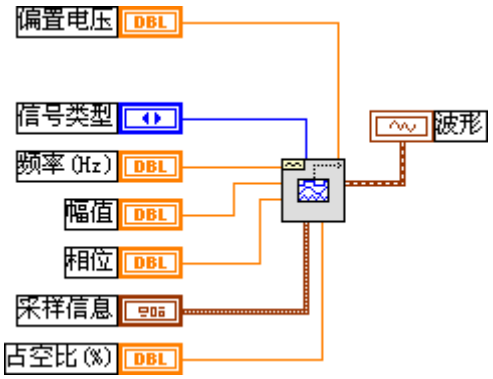
Fs ：采样率，单位是样本数/秒，缺省值为 1000。

#s ：波形的样本数，缺省值为 1000。

duty cycle (%)：占空比，对方波信号是反映一个周期内高低电平所占的比例，缺省值为 50%。
signal out：信号输出端
phase out ：波形的相位，单位：度。
error out ：错误信息。如果 error in 指示一个错误，error out 包含同样的错误信息。否则，它描述该 VI 引起的错误状态。

使用该 VI 制作的函数发生器如下，由框图可以看出，其中没有附加任何其他部件。





6． 3 标准频率

在模拟状态下，信号频率用 Hz 或者每秒周期数为单位。但是在数字系统中，通常使用数字频率，它是模拟频率和采样频率的比值，表达式如下：

数字频率＝模拟频率/采样频率

这种数字频率被称为标准频率，单位是周期数/采样点。

有些信号发生 VI 使用输入频率控制量 f，它的单位和标准频率的单位相同：周期数/每个采样点，范围从 0 到 1，对应实际频率中的 0 到采样频率 fs 的全部频率。它还以 1.0 为周期，从而令标准频率中的 1.1 与 0.1 相等。例如某个信号的采样频率是奈奎斯特频率（fs/2），就表示每半个周期采样一次（也就是每个周期采样两次）。与之对应的标准频率是 1/2 周期数/采样点，也就是 0.5 周期数/采样点。标准频率的倒数 1/f 表示一个周期内采样的次数。

如果你所使用的 VI 需要以标准频率作为输入，就必须把频率单位转换为标准单位：周期数/采样点。

6．4　数字信号处理

6．4．1　FFT 变换

信号的时域显示（采样点的幅值）可以通过离散傅立叶变换（DFT）的方法转换为频域显示。为了快速计算 DFT，通常采用一种快速傅立叶变换(FFT)的方法。当信号的采样点数是 2 的幂时，就可以采用这种方法。

FFT 的输出都是双边的，它同时显示了正负频率的信息。通过只使用一半 FFT 输出采样点转换成单边 FFT。FFT 的采样点之间的频率间隔是 fs/N，这里 fs 是采样频率。

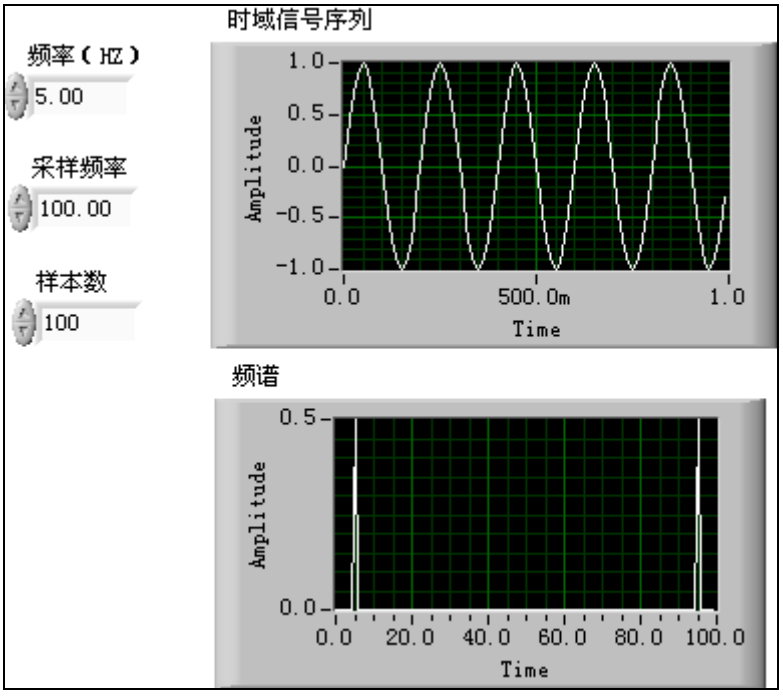
Analyze 库中有两个可以进行 FFT 的 VI，分别是 Real FFT VI 和 Complex FFT VI。

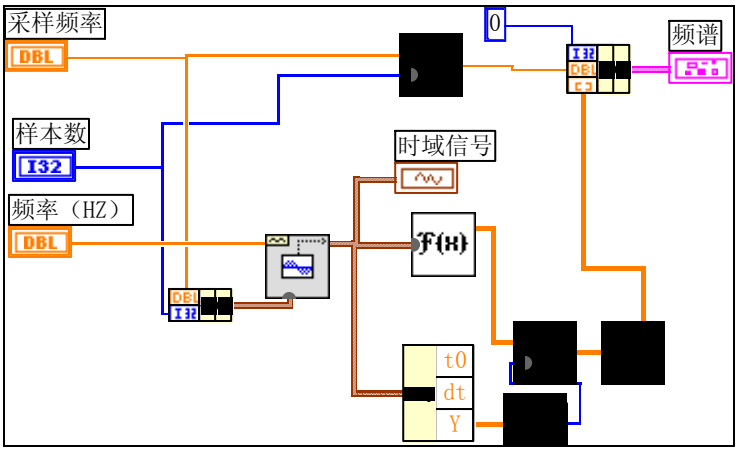
这两个 VI 之间的区别在于，前者用于计算实数信号的 FFT，而后者用于计算复数信号的 FFT。它们的输出都是复数。

大多数实际采集的信号都是实数，因此对于多数应用都使用 Real FFT VI 。当然也可以通过设置信号的虚部为 0，使用 Complex FFT VI 。使用 Complex FFT VI 的一个实例是信号含有实部和虚部。这种信号通常出现在数据通信中，因为这时需要用复指数调制波形。

计算每个 FFT 显示的频率分量的能量的方法是对频率分量的幅值平方。高级分析库中 Power Spectrum VI 可以自动计算能量频谱。Power Spectrum VI 的输出单位是 $Vrms^2$ 。但是能量频谱不能提供任何相位信息。

FFT 和能量频谱可以用于测量静止或者动态信号的频率信息。FFT 提供了信号在整个采样期间的平均频率信息。因此，FFT 主要用于固定信号的分析（即信号在采样期间的频率变化不大）或者只需要求取每个频率分量的平均能量。





2．流程图中的 Array Size 函数用来根据样本数转换 FFT 的输出，得到频率分量的正确幅值。

3．把该 VI 保存为 LabVIEW\Activity 目录中的 FFT_2sided.vi。

4．选择频率（Hz）=10，采样率= 100，样本数= 100。执行该 VI。注意这时的时域图和频谱图。因为采样率=样本数= 100 ，所以时域图中的正弦波的周期数与选择的频率相等，即可以显示 10 个周期。（如果把频率改成 5，那么就会显示 5 个周期）

双边 FFT

5．检查频谱图可以看到有两个波峰，一个位于 10Hz,另一个位于 90Hz，90Hz 处的波峰实际上是 10Hz 处的波峰的负值。因为图形同时显示了正负频率，所以被称为双边 FFT。

6．先后令频率=10、20（Hz），执行该 VI。注意每种情况下频谱图中波峰位置的移动。观察频率等于 10 和 20 时的时域波形。注意哪种情况下的波形显示更好，并解释原因。

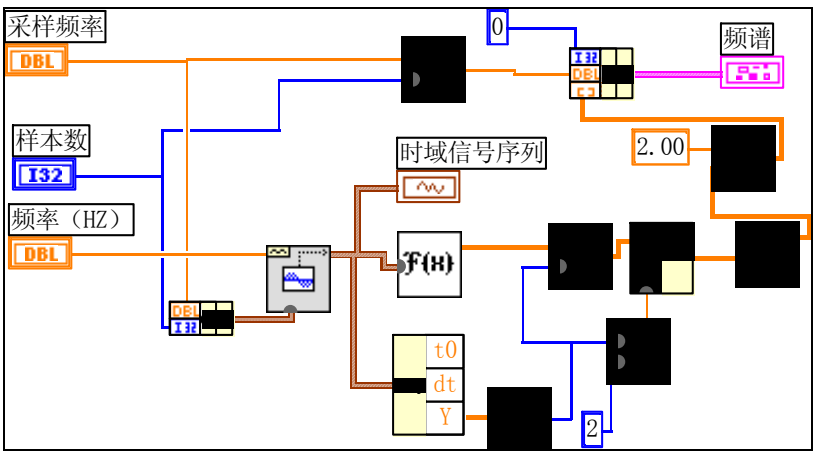
7．因为 fs = 100 Hz，所有只能采样频率低于 50Hz 的信号（奈奎斯特频率＝fs/2）。把频率修改为 48Hz，可以看到频谱图的波峰位于± 48 Hz。

8．把频率改为 52HZ，观察这时产生的图形与第 5 步产生的图形的区别。因为 52 大于奈奎斯特频率，所以混频偏差等于|100 – 52| = 48 Hz。

9．把频率改成 30 和 70Hz，执行该 VI。观察这两种情况下图形是否相同，并解释原因。

单边 FFT

10.按照下图修改流程图。上面已经知道因为 FFT 含有正负频率的信息，所以可以 FFT 具有重复信息。现在这样修改之后只显示一半的 FFT 采样点（正频率部分）。这样的方法叫做单边 FFT。单边 FFT 只显示正频部分。注意要把正频分量的幅值乘以 2 才能得到正确的幅值。但是，直流分量保持不变。（若程序中考虑含直流分量的情况，应当增加一个分支或 case 结构。



11．设置频率（Hz） = 30，采样率= 100，样本数= 100， 运行该 VI。

12．保存该 VI 为 LabVIEW\Activity 目录下的 FFT_1sided.vi。

13．把频率改为 70Hz，执行该 VI，观察这时产生的图形与第 9 步产生的图形的区别。

练习 6-1 结束。

6．4．2 窗函数

计算机只能处理有限长度的信号，原信号 x(t)要以 T(采样时间或采样长度)截断，即有限化。有限化也称为加“矩形窗”或“不加窗”。矩形窗将信号突然截断，这在频域造成很宽的附加频率成分，这些附加频率成分在原信号 x(t)中其实是不存在的。一般将这一问题称为有限化带来的泄露问题。泄露使得原来集中在 f_0 上的能量分散到全部频率轴上。泄露带来许多问题：如①使频率曲线产生许多“皱纹”(Ripple)，较大的皱纹可能与小的共振峰值混淆；②如信号为两幅值一大一小频率很接近的正弦波合成，幅值较小的一个信号可能被淹没。③f_0 附近曲线过于平缓，无法准确确定 f_0 的值。

为了减少泄露，人们尝试用过渡较为缓慢的、非矩形的窗口函数。常用的窗函数如下表所示。

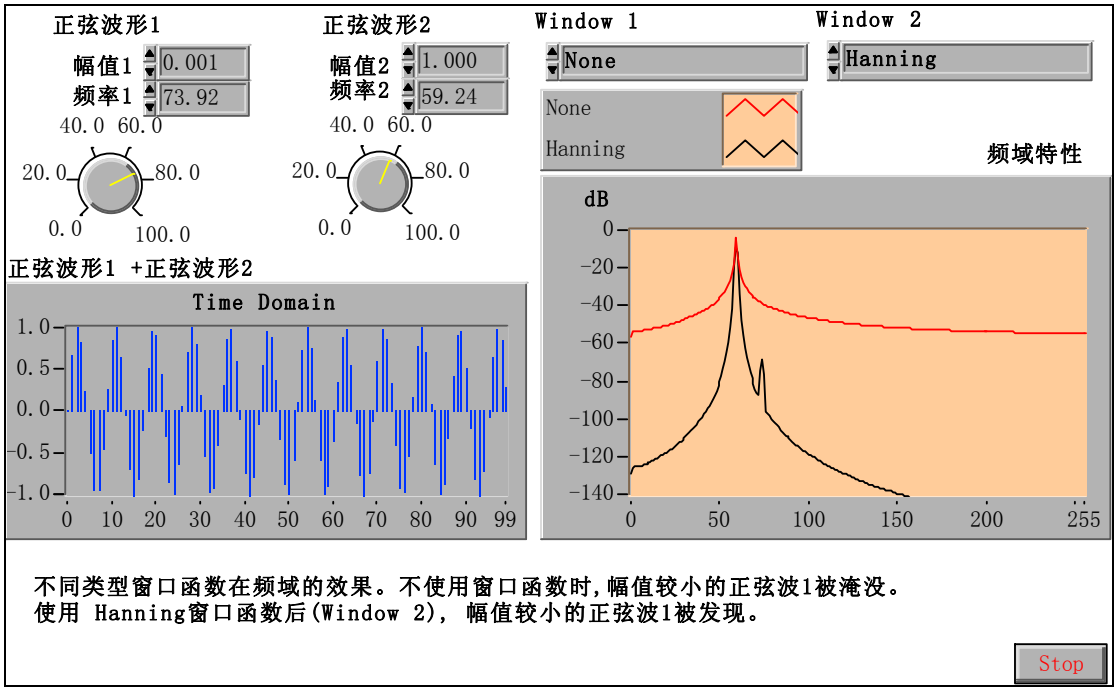
窗	定 义	应 用
矩形窗（无窗）	W[n]=1.0	区分频域和振幅接近的信号瞬时信号宽度小于窗
指数形窗	W[n]=exp[n*lnf/N-1] f=终值	瞬时信号宽度大于窗
海宁窗	W[n]=0.5cos(2nπ/N)	瞬时信号宽度大于窗普通目的的应用
海明窗	W[n]=0.54-0.46cos(2nπ/N)	声音处理
平顶窗	W[n]=0.2810639-0.5208972cos(2nπ/N) +0.1980399cos(2nπ/N)	分析无精确参照物且要求精确测量的信号
Kaiser-Bessel 窗	$W[n]=I^{o}(\beta)$	区分频率接近而形状不同的信号
三角形窗	W[n]=1-｜（2n-N）/N｜	无特殊应用

在实际应用中如何选择窗函数一般说来是要仔细分析信号的特征以及最终你希望达到的目的，并经反复调试。窗函数有利有弊，使用不当还会带来坏处。使用窗函数的原因很多，例如：

- 规定测量的持续时间。
- 减少频谱泄漏。
- 从频率接近的信号中分离出幅值不同的信号。

下面的例子（详见 LabVIEW 6i 中的 Search Examples > Fundamentals Examples >Analysis Examples > Signal Processing > Windows Examples > Window Comparison）是从频率接近的信号中分离出幅值不同的信号，正弦波 1 与正弦波 2 频率较接近，但幅值相差 1000 倍，相加后产生的信号变换到频域，如果在 FFT 之前不加窗，则频域

特性中幅值较小的信号被淹没。加 Hanning 窗后两个频率成分都被检出。



6．4．3　谐波失真与频谱分析

当一个含有单一频率(比如 f_1)的信号 x(t)通过一个非线性系统时，系统的输出不仅包含输入信号的频率（f_1），而且包含谐波分量（$f_2=2f_1, f_3=3f_1, f_4=4f_1$ 等等），谐波的数量以及它们对应的幅值大小取决于系统的非线性程度。电网中的谐波是一个值得关注的问题。

下面的一个非线性系统的例子是输出 y(t)是输入 x(t)的立方。假如输入信号：

$$x(t) = \cos(wt)$$

则输出：

$$x^3(t) = 0.5\cos(wt) + 0.25[\cos(wt) + \cos(3wt)]$$

因此，输出不仅含有基波频率 w，而且还有三次谐波的频率 3w。

谐波失真的总量

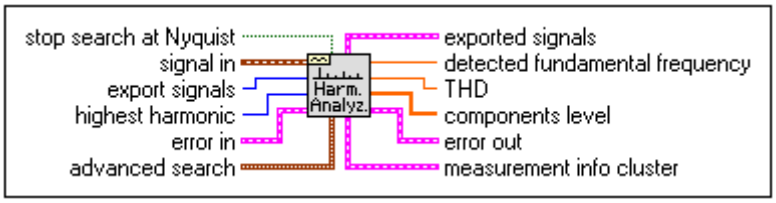
为了决定一个系统引入非线性失真的大小，需要得到系统引入的谐波分量的幅值和基波的幅值的关系。谐波失真是谐波分量的幅值和基波幅值的相对量。假如基波的幅值是 A_1，而二次谐波的幅值是 A_2，三次谐波的幅值是 A_3，四次谐波的幅值是 A_4。。。。。N 次谐波的幅值是 A_N，总的谐波失真（THD）为：

$$THD = \frac{\sqrt{A_2^2 + A_3^2 + \ldots A_N^2}}{A_1}$$

用百分数表示的谐波失真（%THD）为：

$$\%\ THD = \frac{100 * \sqrt{A_2^2 + A_3^2 + \ldots A_N^2}}{A_1}$$

LabVIEW 6i 提供的谐波分析器与以前的版本有一些变化，下面先介绍它



该 VI 对输入信号进行完整的谐波分析，包括测定基波和谐波，返回基波频率和所有的谐波幅度电平，以及总的谐波失真度（THD）。其部分参数含义如下：

stop search at Nyquist：如果设置为 TRUE (缺省值 T)，则只包含低于 Nyquist 频率（采样频率的一半）的谐波。如果设置为 FALSE，该 VI 将继续搜索 Nyquist 范围之外的频率。

signal in ： 输入信号。

export signals ： 选择输出到信号指示器的信号。有如下几种选择：

none——对快速计算；

input signal——定时将输入信号反映到输出端；

fundamental signal——在输出端反映基波；

residual signal——在输出端反映除基波之外的剩余信号；

harmonics only——在输出端反映谐波时域信号及其频谱。

highest harmonic ： 控制最高谐波成分，包括用于谐波分析的基波。例如，对于 3 次谐波分析，该控制将设置测量基波、2 次谐波和 3 次谐波。.

error in ： 在该 VI 运行之前描述错误环境。缺省值为 no error. 如果一个错误发生，该 VI 在 error out 端返回错误代码。该 VI 仅在无错误时正常运行。 错误簇包含如下参数。

status ： 缺省值为 FALSE，发生错误时变为 TRUE。

code ： 错误代码，缺省值为 0。

source ： 在大多数情况下是产生错误的 VI 或函数的名称，缺省值为一个空串。

advanced search ： 控制频域搜索区域，中心频率及频带宽度。该功能用来确定信号的基波。

approx. fund. freq. (optional) ——用来搜索基波的中心频率的估算值。如果设置缺省值为-1.0,，则选择幅值最大的频率成分为基波。

search (+/- % of Fsampl.) ——用来搜索基波频率频带宽度，是采样率的百分比。

exported signals ： 包含输出的时域信号及其频谱供选择。

detected fundamental frequency ： 探测在频域搜索得到的基波。用 advanced search 设置频率搜索范围。所有谐波测量为基波的整数倍。

THD ： 总谐波失真度。它定义为谐波 RMS 之和与基波幅值之比。为了折算为百分数，需要乘以 100。

components level ： 测量谐波幅值的电平（单位 伏），是一个数组。该数组索引包括 0 (DC), 1 (基波), 2 (2 次谐波),... n (n 次谐波), 直到最高谐波成分。

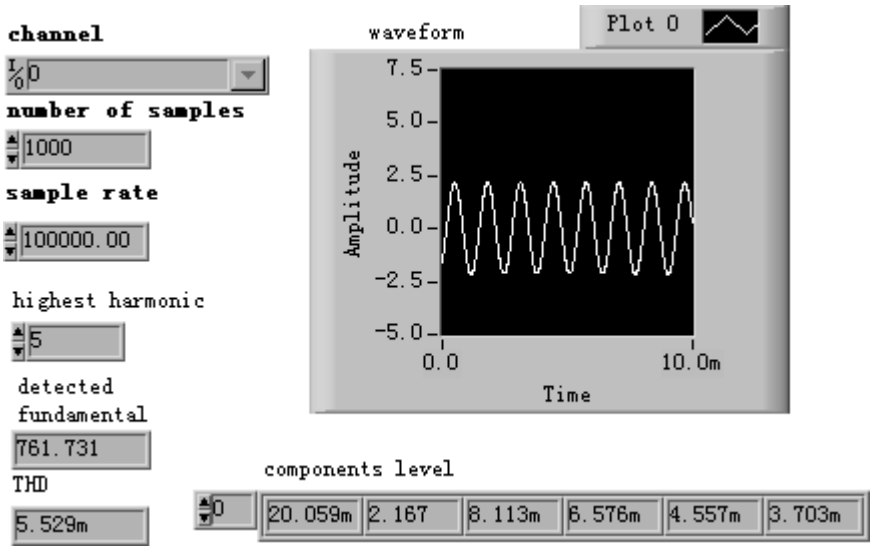
measurement info cluster ： 任何处理期间遭遇的预告

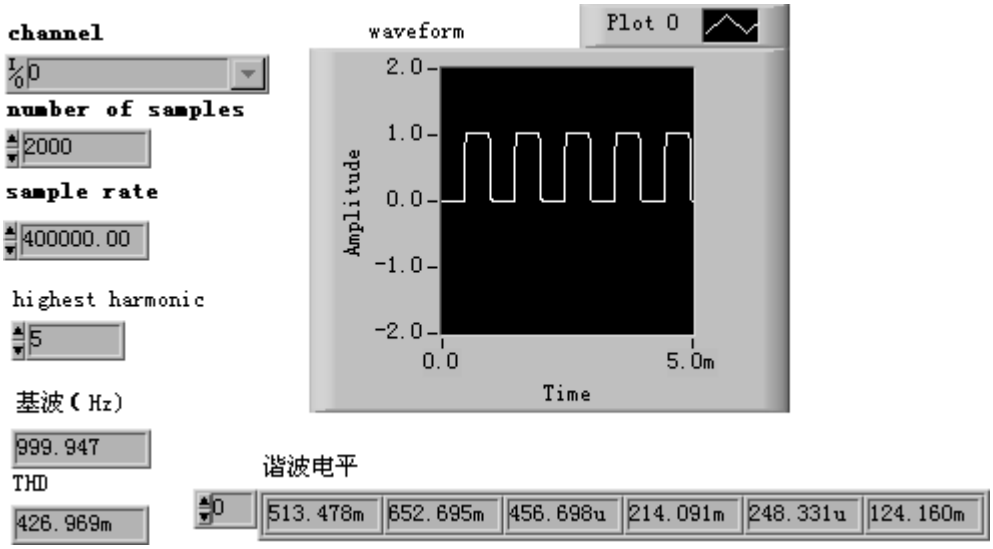
uncertainty 备用；

Warning ： 如果处理期间警告发生为 TRUE。

comments ： 当 Warning 为 TRUE 时的消息内容。

下面是一个谐波分析的例子。由通道 0 输入一个模拟信号，经 DAQ 后进行谐波分析，先后分析了两个信号，首先是一个 761Hz 的正弦信号，第二个信号是一个 1000Hz 的。分析仅限于不高于 5 次的谐波。分析结果见两个前面板。对一个实际的正弦信号，谐波失真总量（THD）与基波电平相比，可以忽略。对方波 THD 就较大了。





谐波分析应用的一个例子

6．3．1　数字滤波

模拟滤波器设计是电子设计中最重要的部分之一。尽管很多参考书都提供了简单可靠的模拟滤波器示例，但是滤波器的设计通常还是需要专家来完成，因为这项工作需要较高深的数学知识和对系统与滤波器之间的关系有深入的了解。

现代的数字采样和信号处理技术已经可以取代模拟滤波器，尤其在一些需要灵活性和编程能力的领域中，例如音频、通讯、地球物理和医疗监控技术。

与模拟滤波器相比，数字滤波器具有下列优点：

- 可以用软件编程
- 稳定性高，可预测
- 不会因温度、湿度的影响产生误差，不需要精度组件
- 很高的性能价格比

在 LabVIEW 中可以用数字滤波器控制滤波器顺序、截止频率、脉冲个数和阻带衰减等参数。

本节所涉及到的数字滤波器都符合虚拟仪器的使用方法。它们可以处理所有的设计问题、计算、内存管理，并在内部执行实际的数字滤波功能。这样您无需成为一个数字滤波器或者数字滤波的专家就可以对数据进行处理。

采样理论指出，只要采样频率是信号最高频率的两倍以上就可以根据离散的、等分的样本还原一个时域连续的信号。假设对信号以△t 为时间间隔进行采样，并且不丢失任何信息，参数 △t 是采样间隔。

可以根据采样间隔计算出采样频率

$$f_s = \frac{1}{\Delta t},$$

根据上面的公式和采样理论可以知道，信号系统的最高频率可以表示为：

$$f_{Nyq} = \frac{f_s}{2}.$$

系统所能处理的最高频率是恩奎斯特频率。这同样适用于数字滤波器。例如，如果采样间隔是 0.001 秒，那么采样频率是

$$f_s = 1{,}000 \text{ Hz},$$

系统所能处理的最高频率是

$$f_{Nyq} = 500 \text{ Hz}.$$

下面几种滤波操作都基于滤波器设计技术：

- 平滑窗口
- 无限冲激响应（IIR）或者递归数字滤波器
- 有限冲激响应（FIR）或者非递归数字滤波器
- 非线性滤波器

很多情况下通带的增益在均值附近稍微发生变化是容许的。通带的这种变化被称为通带波动（*passband ripple*），也就是实际增益与理想增益之间的差值。在实际使用中阻带衰减（*stopband attenuatio*n）也不可能无限接近 0，您必须指定一个符合需要的衰减值。通带波动和阻带衰减都使用分贝或者 dB 为单位，定义是：

$$\text{dB} = 20*\log_{10}(A_o(f)/A_i(f))$$

其中 $\log_{10}$ 表示基值 10 的对数，而 $A_i(f)$ and $A_0(f)$ 分别是频率在滤波前后的幅值。例如，对于 - 0.02 dB 的通带波动，表达式是：

$$-0.02 = 20*\log_{10}(A_o(f)/A_i(f))$$
$$A_o(f)/A_i(f) = 10^{-0.001} = 0.9977$$

这表明输入输出的幅值非常接近。

如果阻带衰减为－60 dB ，那么可以得到：

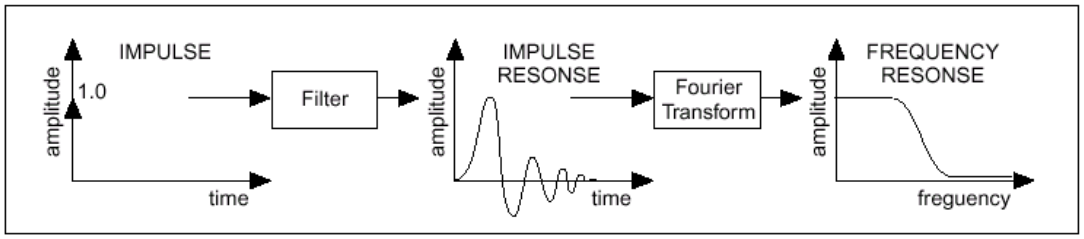
$$-60 = 20*\log_{10}(A_o(f)/A_i(f))$$
$$A_o(f)/A_i(f) = 10^{-3} = 0.001$$

这表明输出幅值是输入幅值的 1/1000。

衰减值通常用不带负号的分贝为单位，但是默认为负值。

IIR 和 FIR 滤波器

另外一种滤波器分类方法是根据它们的冲激响应的类型。滤波器对于输入的冲激信号（x[0] = 1 且对于所有 I<>0，x[i] = 0）的响应叫做滤波器的冲激响应（impulse response），如下图所示。冲激响应的傅立叶变换被称为滤波器的频率响应（frequency response）。根据滤波器的频率响应可以求出滤波器在不同频率下的输出。换句话说，根据它可以求出滤波器在不同频率时的增益值。对于理想滤波器，通频带的增益应当为 1，阻带的增益应当为 0。所以，通频带的所有频率都被输出，而阻带的所有频率都不被输出。



如果滤波器的冲激响应在一定时间之后衰减为 0，那么这个滤波器被称为有限冲激响应（FIR）滤波器。但是，如果冲激响应一直保持，那么这个滤波器被称为无限冲激响应滤波器（IIR）。冲激响应是否有限（即滤波器是 IIR 还是 FIR）取决于滤波器的输出的计算方法。

IIR 滤波器和 FIR 滤波器之间最基本的差别是，对于 IIR 滤波器，输出只取决于当前和以前的输入值，而对于 FIR 滤波器，输出不仅取决于当前和以前的输入值，还取决于以前的输出值。简单地说，FIR 滤波器需要使用递归算法。

IIR 滤波器的缺点是它的相位响应是非线形的。在不需要相位信息的情况下，例如简单的信号监控，那么 IIR 滤波器就符合需要。而对于那些需要线形相位响应的情况，应当使用 FIR 滤波器。但是，IIR 滤波器的递归性增大了它的设计与执行的难度。

因为滤波器的初始状态是 0（负指数是 0），所以在到达稳态之前会出现与滤波器阶数相对应的过渡过程。对于低通和高通滤波器，过渡过程或者延迟的持续时间等于滤波器的阶数。

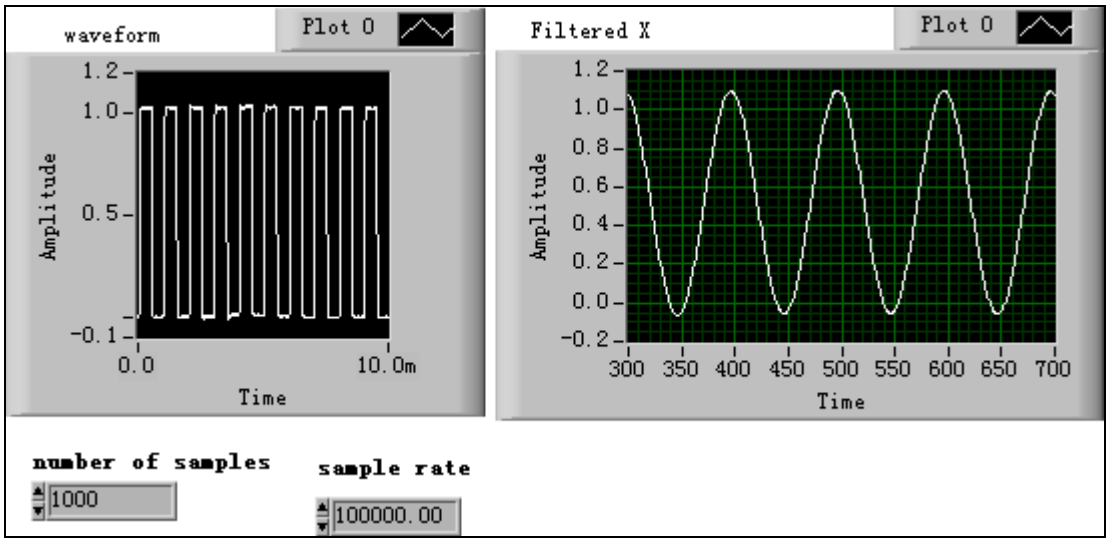
可以通过启动静止内存消除连续调用中的过渡过程，方法是将 VI 的 init/cont 控制对象设置为 TURE（连续滤波）。

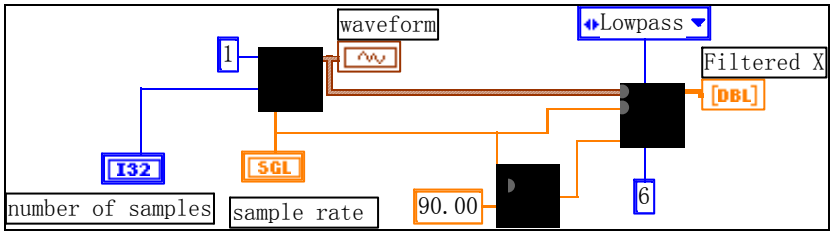
对数字滤波器的详细讨论不是本书的内容，读者可参阅有关数字信号处理的书籍，下面我们具一个简单的例子说明在 LabVIEW 中如何使用数字滤波器。

练习 6-2　使用数字滤波器

目的：使用一个低通数字滤波器对实际采集的方波信号滤波。

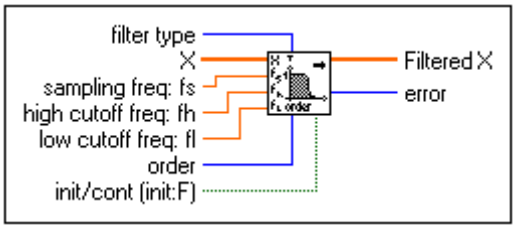
1．创建前面板和流程图如下所示。





2．注意流程图。其中使用了一个数字滤波器模块（**Functions** 模板：**Analyze**）**Signal Processing**） **Filters** 下的 Butterworth Filter.vi）。先介绍一下这个 VI。

Butterworth 滤波器



filter type ：按下列值指定滤波器类型

0:　Lowpass 低通

1:　Highpass 高通

2:　Bandpass 带通

3:　Bandstop 带阻

X ：需要滤波的信号序列

sampling freq fs ：产生 X 序列时的采样频率，必须大于 0。缺省值是 1.0。如果它小于

等于 0 则输出序列 Filtered X 为空并返回一个错误。

high cutoff freq fh ： 高端截止频率。当滤波器类型为 0 (lowpass) 或 1 (highpass)时忽略该参数。

low cutoff freq fl：低端截止频率。它必须满足 Nyquist 准则，即

$$0 \leqq f_i < 0.5 f_s$$

如果该条件不满足则输出序列 Filtered X 为空并返回一个错误。f_i 的缺省值是 0.125。

order ： 大于 0，缺省值是 2。

init/cont ： 内部状态的初始化控制。当其为 FALSE (default), 初态为 0，当 init/cont 为 TRUE, 滤波器初态为上一次调用该 VI 的最后状态。为了对一个大数据量的序列进行滤波，可以将其分割为较小的块，设置这个状态为 FALSE 处理第一块数据，然后改设置为 TRUE 继续对对其余的数据块滤波。

Filtered X ： 滤波样本的输出数组。

3．在了解了这个滤波器的功能之后再来看上面的流程图。

这里 DAQ 部分将一个外部的 1KHz 的方波采集进来，采样频率是 100KHz，采到的方波一方面显示其波形，同时又送到滤波器的入口。滤波器类型设置为 Lowpass，其采样频率端直接连接到前面的采样频率控制端，因而也是 100KHz。另外，将采样频率除以 90 后作为低端截止频率，应该也是合理的，滤波器的阶数选为 6。这样的一个 VI 运行结果如前面板所示。

还需要指出的是原方波不以 X 轴对称，有直流分量，经这个低通滤波器后，直流分量还应当存在，曲线显示的确如此。

练习 6-2 结束。

6．3．2 曲线拟合

曲线拟合（curve fitting）技术用于从一组数据中提取曲线参数或者系数，以得到这组数据的函数表达式。

通常，对于每种指定类型的曲线拟合，如果没有特殊说明，都存在两种 VI 可以使用。一种只返回数据，用于对数据的进一步操作，另一种不仅返回系数，还可以得到对应的拟合曲线和均方差（MSE）。

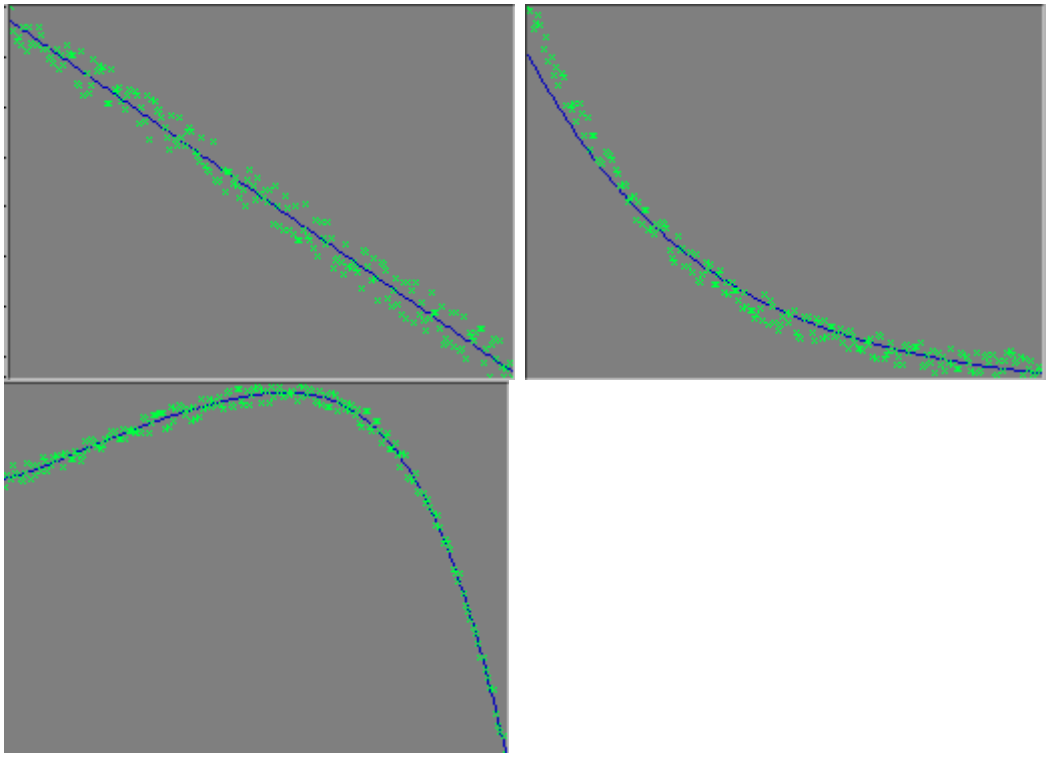
LabVIEW 的分析软件库提供了多种线性和非线性的曲线拟合算法，例如线性拟合、指数拟合、通用多项式拟合、非线性 Levenberg-Marquardt 拟合等。

曲线拟合的实际应用很广泛。例如：

- 消除测量噪声
- 填充丢失的采样点（例如，如果一个或者多个采样点丢失或者记录不正确）
- 插值（对采样点之间的数据的估计；例如在采样点之间的时间差距不够大时）
- 外推(对采样范围之外的数据进行估计,例如在需要在试验以后或者以后的数值时）
- 数据的差分（例如在需要知道采样点之间的偏移时，可以用一个多项式拟合离散数据，而得到的多项式可能不同）
- 数据的合成（例如在需要找出曲线下面的区域，同时又只知道这个曲线的若干个离散采样点的时候）
- 求解某个基于离散数据的对象的速度轨迹（一阶导数）和加速度轨迹（二阶导数）

下面是使用 LabVIEW 提供的算法得到的三种拟合的例子：线形拟合（左上）、指数拟

合（右上）、多项式拟合（左下）。



一般说来，采集得到的数据大都需要经过适当的处理，其中包括滤波、曲线拟合等。详细内容请参考有关资料。