第 5 章

复合数据类型

在第 3 章中，学习了 LabVIEW 中的基本数据类型，包括数值、字符串、布尔量和枚举等。本章将介绍 LabVIEW 特有的所谓复合数据类型，主要有数组、簇、波形、DDT 和变体。

5.1 数组

数组是相同类型元素的集合。在 LabVIEW 中，数组元素的索引号从 0 开始；数组可以是一维的，或多维的。与 C 语言不同的是，在 LabVIEW 中，创建数组时，不用事先指定数组的大小，即数组的长度，可以根据所编写 VI 的实际需求而改变。[1-4]

5.1.1 数组的创建

在 LabVIEW 中创建一个数组，有以下 5 个步骤。

第 1 步，先创建数组框架。数组框架有两种（如图 5.1 和图 5.2 所示）：一种用于建立数组输入控件和显示控件框架，找到它的路径是“控件”选板→“新式”→“数组、矩阵与簇”→“数组”子选板；另一种用于建立数组常量框架，找到它的路径为“函数”选板→“编程”→“数组”子选板。LabVIEW 默认初建的数组框架是一维的。

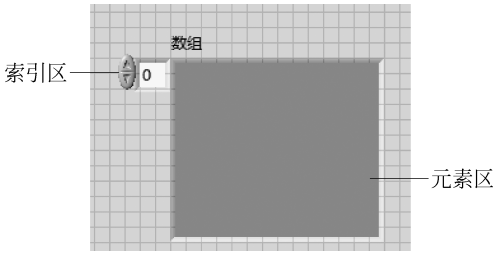


图 5.1 在前面板创建数组输入控件和数组显示控件框架

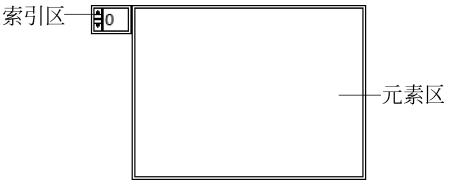


图 5.2 在程序框图面板创建数组常量框架

第 2 步，向数组框架中添加“元素”，以确定数组元素的具体数据类型。

第 3 步，以拖动方式操作，确定数组元素的可视大小，如图 5.3 所示；通过拖曳鼠标，可同时显示数组中的多个元素，具体如图 5.4 所示。

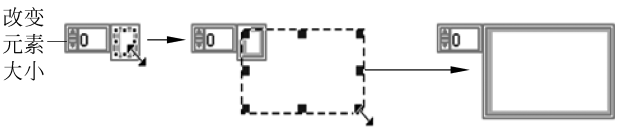


图 5.3　改变数组元素的可视大小(以数组常量为例)

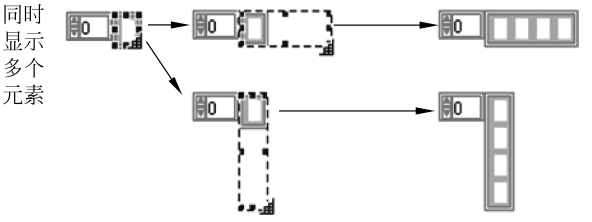
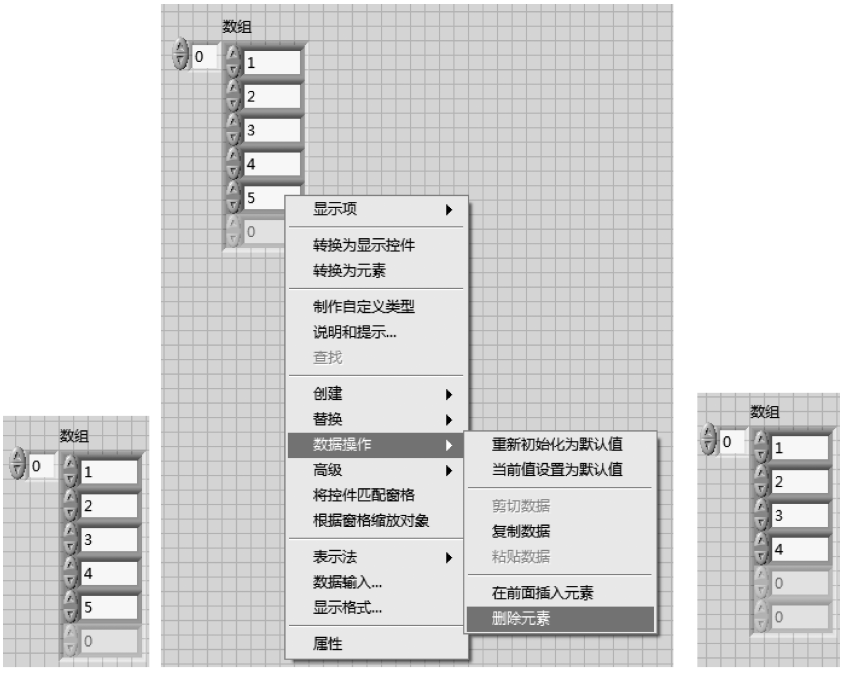


图 5.4　同时显示多个元素(以数组常量为例)

第 4 步,给数组中的元素赋值。

第 5 步,增加数组的维度。有两种实现方法:一种是用鼠标选中数组,右击鼠标,弹出快捷菜单,选择“添加维度”或“删除维度”;另一种是将鼠标移至数组左上角区域,通过拖曳,可以改变数组的维数。

按照上述步骤创建好的一个数组输入控件如图 5.5(a)所示。有时,需要将数组中的某个元素删除,操作步骤如下:将鼠标放在要删除的元素(例如,元素 5)处,右击鼠标,选择“数据操作”中的“删除元素”(如图 5.5(b)所示)。删除元素 5 后的数组如图 5.5(c)所示。



(a)　(b)　(c)

图 5.5　删除数组中的元素

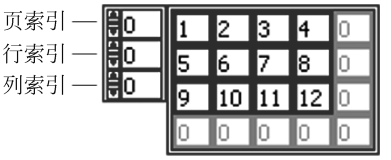
5.1.2　数组的索引

如图 5.1 和图 5.2 所示，数组框架的左上角区域提供的是数组的索引，即该区域也称索引区域；数组框架中索引区域以外的主要区域是元素区域。在 LabVIEW 中，查看数组索引的规则是：索引区域显示的值，对应的是元素区域所显示的左上角元素的索引值。对于一维数组，数组的索引是行或者列；对于二维数组，数组的索引是行和列；而对于三维数组，数组的索引为页、行和列。数组各维的索引都是从 0 开始排序的。

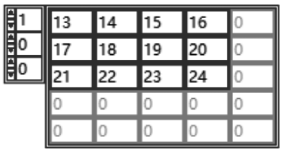
下面通过例 5.1，学习如何查看 LabVIEW 中数组的索引。

【例 5.1】 数组的索引。

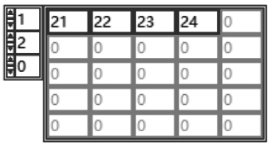
如图 5.6 所示的数组，是一个两页的三维数组。从图 5.6 可以看出，在该数组中，第 0 页上元素“1”的索引是(0,0,0)，而第 1 页上元素“13”的索引是(1,0,0)。而当索引区域变为(1,2,0)时，数组的显示如图 5.6(c)所示。



(a) 索引为(0，0，0)



(b) 索引为(1，0，0)



(c) 索引为(1，2，0)

图 5.6　三维数组举例

通过例 5.1，可加深对数组索引区域显示的值的理解，即，它永远是数组元素区域所显示的左上角元素的索引值。

5.1.3　数组函数

LabVIEW 中提供有一些数组函数，它们都在“函数”选板→“编程”→“数组”子选板上。表 5.1 列出了 5 个常用的数组函数，下面通过几个示例对这几个数组函数做具体介绍。

表 5.1 数组函数

序号	名 称	图标和连接端口	功 能 说 明
1	数组大小	数组 → 大小	提供该数组各维的长度
2	索引数组	n维数组、索引|0、索引|n-1 → 元素或子数组	返回 n 维数组在索引位置的元素或子数组
3	数组子集	n维数组、索引|0(0)、长度0(剩余)、索引|n-1(0)、长度n-1(剩余) → 子数组	返回数组的一部分，从索引处开始，包含长度(值)个元素
4	初始化数组	元素、维数大小0、…、维数大小n-1 → 初始化的数组	创建一个 n 维数组，其中的每个元素都被初始化为元素的值
5	创建数组	数组、元素、元素、元素 → 添加的数组	将若干个输入数组和元素组合成一个新的数组

【例 5.2】 “数组大小”函数。

例 5.2 的 VI 的程序框图和前面板，如图 5.7 所示。它完成的是将一个三维数组常量连至“数组大小”函数，然后将此函数的输出结果提供给“大小”显示控件。运行此 VI，从前面板上“大小”输出控件的显示结果可以看出，这个数组的大小为 2 页、3 行和 4 列。

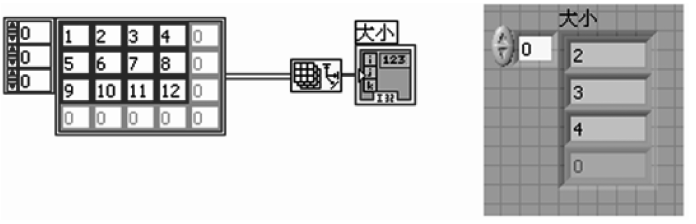


图 5.7 “数组大小”函数使用示例

【例 5.3】 “索引数组”函数。

例 5.3 的 VI 的程序框图和前面板，如图 5.8 所示。它所实现的是将一个 5 行 3 列的二维数组常量连至“索引数组”函数。摆放位置在上的被调用的“索引数组”函数，索引的是原

二维数组第 1 行的元素，输出结果是原二维数组的一个子数组，且是一个一维数组。而摆放位置在下的被调用的“索引数组”函数，其索引的是原二维数组中第 1 行第 2 列的那个元素，输出的是一个数值常量。

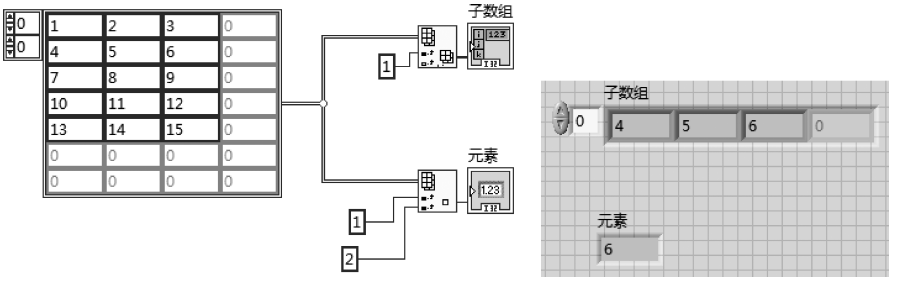


图 5.8　“索引数组”函数使用示例

【例 5.4】　“数组子集”函数。

例 5.4 的 VI 的程序框图和前面板，如图 5.9 所示。它完成的，是将一个 5 行 3 列的二维数组常量连至“数组子集”函数。其中，“数组子集”函数索引的是原二维数组从第 1 行开始、长度为 3 的一个子二维数组，具体输出的子二维数组有 3 行 3 列。

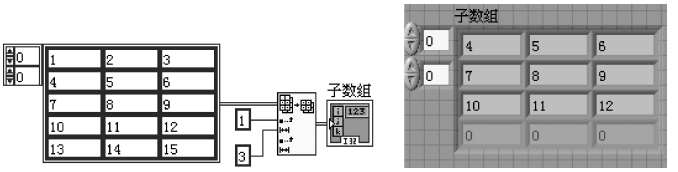


图 5.9　“数组子集”函数使用示例

【例 5.5】　“删除数组元素”函数。

例 5.5 的 VI 的程序框图和前面板，如图 5.10 所示。输入的数组是一维的，共有 5 个元素，分别是(1,2,3,4,5)。该 VI 调用了“删除数组元素”函数，将输入数组中索引号为 2、长度为 1 的元素删除掉了，即元素“3”被删掉了，结果如图 5.10 所示。

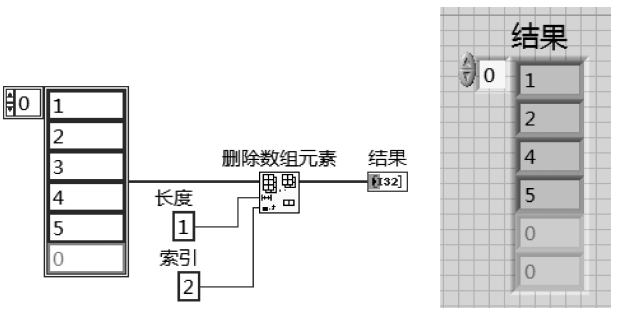


图 5.10　“删除数组元素”函数使用示例

【例 5.6】 “初始化数组”函数。

例 5.6 的 VI 的程序框图和前面板，如图 5.11 所示。其中，第 1 个“初始化数组”函数（摆放位置在上的）创建了一个长度（大小）为 5 的一维数组，且其中的每个元素都是 1；第 2 个“初始化数组”函数创建了一个 5 行 3 列的二维数组，且其每个元素都是 2。

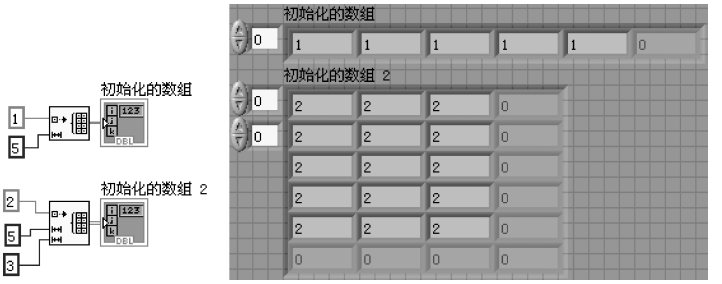


图 5.11 “初始化数组”函数使用示例

【例 5.7】 “创建数组”函数。

在图 5.12 所示 VI 的程序框图面板上，基于两个一维数组常量，利用“创建数组”函数生成了两个新数组。其中，摆放位置在上的“创建数组”函数的“连接输入”选项是勾选的，可实现将两个一维数组串接起来，生成一个新的一维数组。而摆放位置在下的“创建数组”函数的“连接输入”选项是未选择的，其实现的是将两个一维数组作为元素，生成另一个新的二维数组，并以原最长的一维数组的大小作为新建的二维数组相应维的大小，且对缺少的部位进行自动补 0。

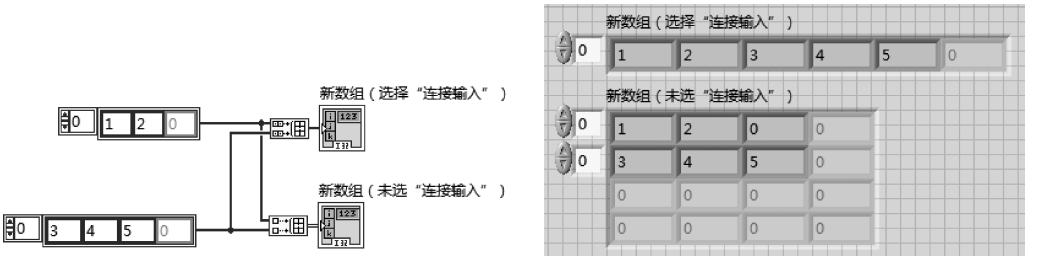


图 5.12 “创建数组”函数使用示例

常见问题 22：“创建数组”函数的“连接输入”选项。

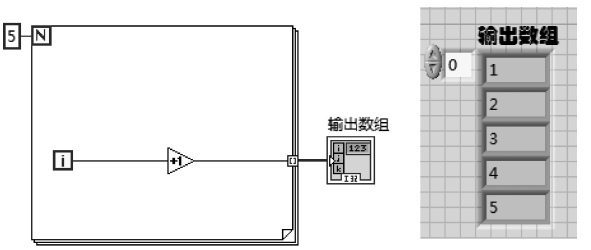
在使用“创建数组”函数时，需要注意其快捷菜单中的“连接输入”选项，此选项勾选时，按顺序拼接所有输入，形成一个新的输出数组，该输出数组的维数与所连接的最大维数输入数组的维数相同；此选项未选择时，则要求所有输入数组的维数必须相同，且经该函数操作后，输出的新数组会比输入数组高一维。

5.1.4　利用循环结构创建数组

在第 4 章，曾经学过循环结构对数组有自动索引的功能。有读者就此也许会想到，是否可以利用循环结构的自动索引功能来创建数组呢？答案是肯定的，而且利用这种思路来创建数组，在以后的编程中会经常用到。

【例 5.8】 利用循环结构创建一个一维数组，数组元素从 1 到 5。

例 5.8 的 VI 的程序框图和前面板，如图 5.13 所示。可见，该 VI 调用了一个 For 循环，其循环总数接线端接入常量 5，并将其计数接线端 i 进行＋1 的运算，然后将其运算结果连至右边框上，设置自动索引打开，即 For 循环右边框上的隧道呈空心状态，然后将经隧道传出的值提供给一个一维数组显示控件。运行此 VI，结果如图 5.13(b)所示。



(a)　　(b)

图 5.13　利用循环结构创建数组

5.1.5　函数的多态化功能

在 LabVIEW 中，函数大多数都是多态化的。所谓多态化，是指一种函数功能，它可以协调不同格式、维数的输入数据，如图 5.14 所示。这里以加法函数为例，它可以接收两个标量相加，结果还是标量；还可以接收标量与数组的相加，具体是将这个标量分别加到数组的各个元素上；另外，还可以接收两个数组的相加，具体是分别将这两个数组对应的元素进行相加。

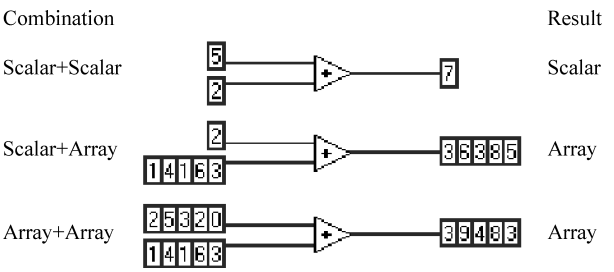


图 5.14　函数的多态化举例

很多函数都具备多态化的功能，这是 LabVIEW 为方便用户使用而设计构建的，读者可以先建立起这样的基本概念。

5.2 簇

簇是多个元素的集合。与数组不同的是，簇的元素可以是不同类型的，类似于 C 语言的结构。利用簇可以在编写 VI 的过程中将分布在程序框图上不同位置的数据元素组合起来，这样可以减少连线的拥挤程度；另外，在建立子 VI 时，利用簇将不同类型的元素组合起来，还可以减少子 VI 接线端的数量。在实际应用中，当要对一个所编写的测量仪器 VI 的若干个不同性质的参数进行配置时，就可以使用簇来实现[1-4]。

5.2.1 簇的创建

LabVIEW 中簇的创建方法与创建数组相类似，共有如下 3 个步骤。

第 1 步，首先要创建簇框架，如图 5.15 所示。同数组一样，簇框架也有两种：一种是簇输入控件和簇显示控件框架，位于“控件”选板→“新式”→“数组、矩阵与簇”子选板上；另一种是簇常量框架，位于“函数”选板→“编程”→“簇、类与变体”子选板上。

第 2 步，向簇框架中添加元素，如图 5.15 所示。

第 3 步，通过拖曳确定簇的可视大小，如图 5.16 所示。

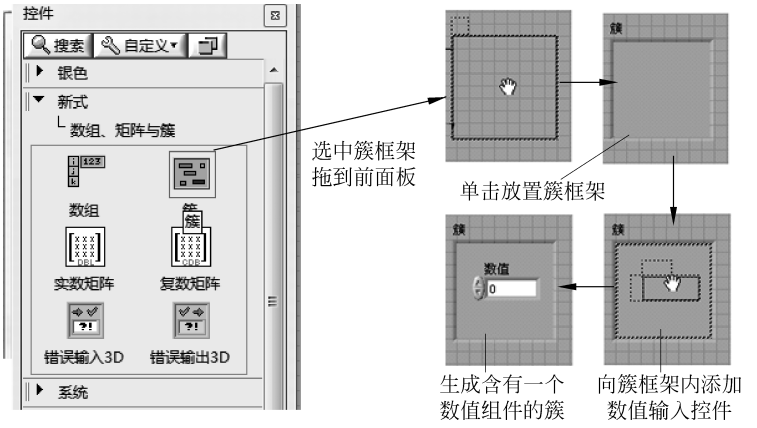


图 5.15 在前面板上创建簇

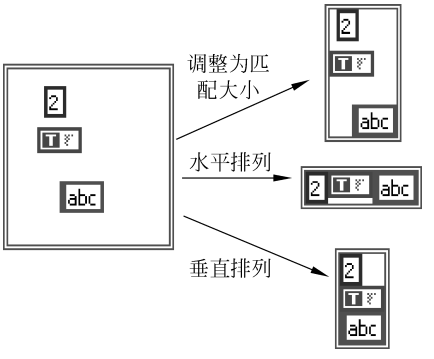


图 5.16 在程序框图面板上创建簇常量并改变其可视大小

5.2.2　簇的顺序

在簇中，元素有一定的排列顺序，该顺序就是创建该簇时添加元素的顺序。簇元素的排列顺序很重要，是完成很多操作的依据。

簇中元素的顺序是可以改变的。具体的操作是，在簇框架上右击鼠标，弹出快捷菜单，选择“重新排序簇中控件”，则打开了簇元素顺序的编辑状态。如图 5.17 所示，簇的每个元素上都有两个序号，左侧的为新序号，右侧的为旧序号。第一次，单击簇元素之一，改变其序号；随后，对其他的元素重复上述过程，直到改好所有元素的顺序为止，单击上方工具栏中的“确认”按钮，保存此次对簇元素排序所做的修改。

图 5.17　簇元素的顺序

5.2.3　簇函数

表 5.2 中列出了有关簇的主要函数，分别是“捆绑”“解除捆绑”“按名称捆绑”和“按名称解除捆绑”等函数。

表 5.2　簇函数

序号	名　　称	图标和连接端口	功 能 说 明
1	捆绑	捆绑 [Bundle] 簇 元素0 元素1 … 元素n-1 输出簇	(1) 将所有输入元素打包成簇 (2) 替换成新簇
2	解除捆绑	解除捆绑 [Unbundle] 簇 元素0 元素1 … 元素n-1	将簇中的元素分解出来
3	按名称捆绑	按名称捆绑 [Bundle By Name] 输入簇 元素0 … 元素m-1 名称 0 名称 -1 输出簇	(1) 按标签替换“输入簇”中的元素；替换结果从“输出簇”提供出来 (2) “输入簇”必须接入，且要求其至少 1 个元素有标签

续表

序号	名　　称	图标和连接端口	功 能 说 明
4	按名称解除捆绑	按名称解除捆绑 [Unbundle By Name] 已命名簇 名称 0 名称 m-1 元素0 … 元素m-1	(1) 将输入簇中的元素按标签解除捆绑 (2) 在函数输出端,只能获得拥有标签的簇元素

【例 5.9】 “捆绑”函数。

例 5.9 的 VI 的程序框图和前面板,如图 5.18 所示。从图 5.18 中的程序框图可见,该 VI 利用“捆绑”函数将 3 个常量(字符串常量“abc”、数值常量“1”和布尔常量“True”)打包成一个簇,其结果经前面板的“输出簇”控件显示出来。

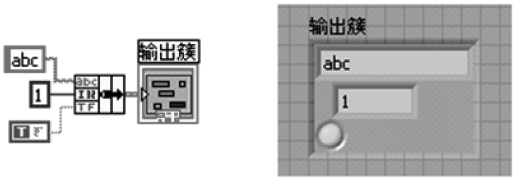


图 5.18　捆绑函数应用示例 1

“捆绑”函数的另一个功能是替换成新簇,图 5.19 所示的 VI 展示了这一用法。已知一个簇,其中的元素为字符串常量“ABC”、数值常量“2”和布尔常量“False”,将这个簇提供给“捆绑”函数,该函数就会自动识别输入簇中各元素的数据类型,并在输入端口上给出标示,例如,“捆绑”函数的第一个连线输入端子上有“abc”的标示,表示簇中的第一个元素为字符串常量。然后,将一个新字符串常量“abc”连至“捆绑”函数的第 1 个输入端子上,把布尔常量“True”连至第 3 个输入端子上,再将“捆绑”函数的输出结果赋给“输出簇”控件。运行此 VI,可以看到,初始簇中的字符串常量元素即大写的“ABC”被小写的“abc”所替换,同时,布尔常量元素也由“False”变为了“True”。

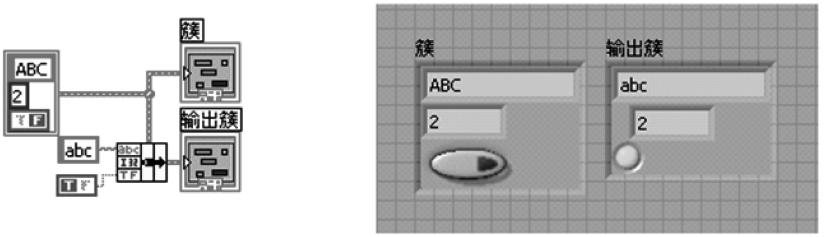


图 5.19　捆绑函数应用示例 2

【例 5.10】 “解除捆绑”函数。

例 5.10 给出了“解除捆绑”函数的使用示例,实现它功能的 VI 的程序框图和前面板如

图 5.20 所示。从程序框图可见，一个簇常量连至“解除捆绑”函数的输入端，该函数对该簇进行解包，并会自动辨识出其中各元素的数据类型，之后，将各元素连至相应的输出控件，在前面板显示出来。

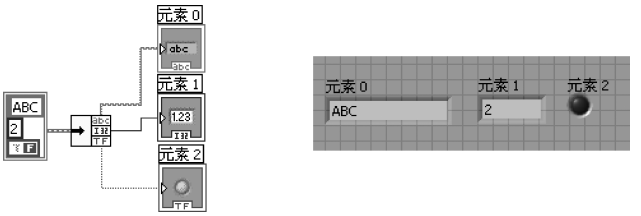


图 5.20　解除捆绑函数应用示例

“按名称捆绑”函数，相当于“捆绑”函数的替换成新簇的功能。使用该函数时，要求“输入簇”必须接入，且至少其中的 1 个元素要有标签。下面通过例 5.11 学习该函数的使用。

【例 5.11】 “按名称捆绑”函数。

例 5.11 给出了“按名称捆绑”函数使用示例的 VI，其程序框图和前面板如图 5.21 所示。从该 VI 的程序框图可见，一个簇常量连至“按名称捆绑”函数，该函数会自动辨识出输入簇中有标签的元素；将新元素连至“按名称捆绑”函数的输入端口上，替换生成的新簇就会通过输出簇控件在前面板显示出来。运行此 VI 可以看出，新元素(“abc”和“true”)已经替换了原簇常量中的相应元素(“ABC”和“false”)。

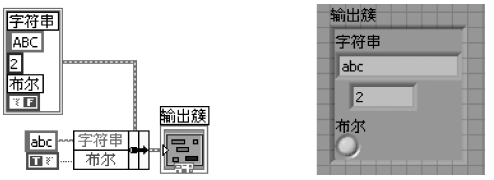


图 5.21　“按名称捆绑”函数应用示例

常见问题 23：如何为簇中的元素添加标签？

可执行如下操作：选中簇中的某个元素，右击鼠标，弹出快捷菜单，从“显示项”子菜单选中“标签”，然后输入一个名称，即可为该元素添加标签。

另外，在初建的“按名称捆绑”函数上，只有一个输入端子，例如，对例 5.11 而言，只有“字符串”输入端子，对此，可以将鼠标移至“字符串”输入端子下边沿，通过向下拖曳鼠标，便可生成更多所需的输入端子。

“按名称解除捆绑”函数的功能是将输入簇中的元素按标签解除捆绑。在该函数的输出端，只能获得带有标签的簇元素。下面将通过例 5.12 学习该函数的使用。

【例 5.12】 “按名称解除捆绑”函数。

例 5.12 给出了“按名称解除捆绑”函数使用示例的 VI,它的程序框图和前面板如图 5.22 所示。在它的程序框图上,是将一个簇常量提供给“按名称解除捆绑”函数,该函数会自动辨识出输入簇中带有标签的元素,然后,再将解包出的元素连至相应的显示控件上。

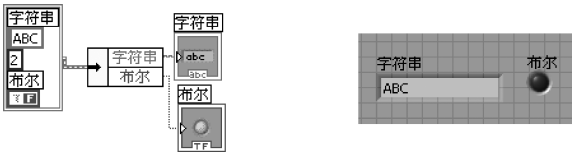


图 5.22 “按名称解除捆绑”函数使用示例

与“按名称捆绑”函数一样,“按名称解除捆绑”函数初建时也只有一个输出端子。单击其标签域,可弹出带有标签的簇元素列表;为看到这些带有不同标签的簇元素,必须对其分别建立相应的显示控件。

5.2.4 错误簇

簇的一个典型应用就是错误簇,如图 5.23 所示。错误簇中的元素有 3 个,分别是“状态”(status)、“代码”(code)和“源”(source)。其中,“状态”是布尔类型的数据,T 表示有错误,F 表示无错误;“代码”是 I32 类型的数据,0 表示无错误;“源”是字符串类型的数据,用以对错误信息进行描述。

图 5.23 错误簇的输入控件和输出控件

在随后将介绍的 LabVIEW 提供的很多函数中都配置有错误簇。如图 5.24 所示,该 VI 是利用数据采集 DAQmx VI 实现模拟输入有限个数据样本。在该 VI 中,下方的连线即为错误簇。利用错误簇的输入和输出,可将用到的各个函数连接起来。此种程序结构将会在后续很多应用场合经常出现,例如,数据采集、文件 IO、串口通信和声音采集等。

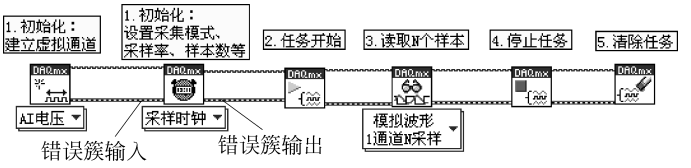


图 5.24 错误簇使用举例

当 VI 出现异常状态时，可以利用错误簇中提供的信息查找出错原因。所以，利用错误簇，可以让所编写的 VI 的运行更加稳定。另外，也可以利用错误簇输入和输出的连接，来规定 VI 中代码的执行顺序。

5.3　波形

波形是一种非常实用的数据类型，利用 LabVIEW 实现数据采集及信号处理时，会经常用到波形这种数据类型。

5.3.1　什么是波形

波形，可以理解为是一种特殊的簇。在 LabVIEW 中，波形含有 4 个组成部分，分别是 t0、dt、“数组 Y”和“属性”。其中，t0 为时间标识，表示波形数据的时间起点；dt 为双精度浮点类型，表示波形数据中相邻数据点之间的时间间隔，以秒为单位；Y 是双精度浮点数组，它按时间顺序给出整个波形的所有数据点；“属性”是变体类型，用于携带任意的属性信息（关于变体数据类型的介绍，请见第 5.5 节）。波形控件位于“控件”选板→“新式”→“I/O”子选板上。

t0 是时间标识。时间标识又称时间戳，是 LabVIEW 中记录时间的专用数据类型。找到时间标识常量的路径是“函数”选板→“编程”→“定时”→“时间标识常量”。而找到时间标识的输入控件和显示控件的路径，则是“控件”选板→“新式”→“数值”子选板。时间标识常量及控件，如图 5.25 所示。

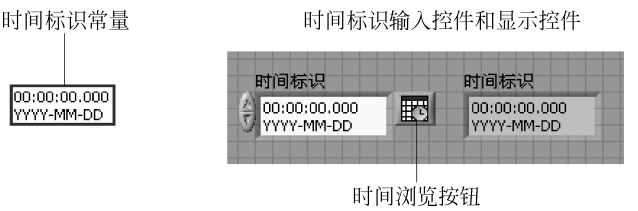


图 5.25　时间标识常量及控件

5.3.2　波形函数

表 5.3 列出了几种典型的波形函数，它们位于“函数”选板→“编程”→“波形”子选板上。其中，在默认情况下，“创建波形”函数只有“波形”和“波形成分”即 Y 输入端子；拖曳该函数图标的边框，可增加 dt、t0 和 attributes（变体类型）输入端子；如果“波形”端子接入了已有的波形数据，则该函数会根据经“波形成分”接入的参数去修改波形数据并输出。

“获取波形成分”函数的功能是将波形数据解包。默认情况下，该函数图标只有 Y 输出端子；拖曳该函数图标的边框，可增加 dt、t0 和 attributes（变体类型）输出端子；也可以单击其输出端子，在弹出的菜单中，选择希望从该输出端子输出波形的哪个成分（数组 Y、dt 或者 t0 等）。

表 5.3　波形函数

序号	名　称	图标和连接端口	功 能 说 明
1	创建波形	创建波形 [Build Waveform] 波形 … 波形成分 波形	创建波形或修改已有波形
2	获取波形成分	获取波形成分 [Get Waveform Components] 波形 t0 Y 波形成分 … 波形成分	将波形数据解包
3	设置波形属性	设置波形属性 [Set Waveform Attribute] 波形 名称 值 错误输入(无错误) 波形输出 替换 错误输出	为输入的波形数据添加“名称”和“值”的属性
4	获取波形属性	获取波形属性 [Get Waveform Attribute] 波形 名称 默认值(空变体) 错误输入(无错误) 波形副本 名称 值 错误输出	获取波形中名为“名称”的属性

下面通过例 5.13～例 5.16，学习“创建波形”和“获取波形成分”两个波形函数的使用。

【例 5.13】　生成一段随机信号，并将其随时间变化的波形在前面板上显示出来。

例 5.13 的 VI 的程序框图和前面板，分别如图 5.26 和图 5.27 所示，它的功能是先利用 For 循环生成一个一维数组，该数组的元素为随机数，数组长度是 100。随后，将该数组赋给“创建波形”函数的 Y 数组的输入端子，并为“创建波形”函数的 dt 输入端子赋一个常量 1，表示数组中两两相邻元素之间的时间间隔为 1s。最后，将生成的波形提供给波形图控件和波形显示控件。利用波形图控件，可以直观地看到所生成的这段随机信号随时间变化的情况；而利用波形显示控件，则可以看到所产生的随机信号波形的具体信息。

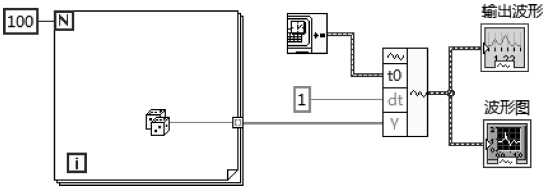


图 5.26　例 5.13 VI 的程序框图

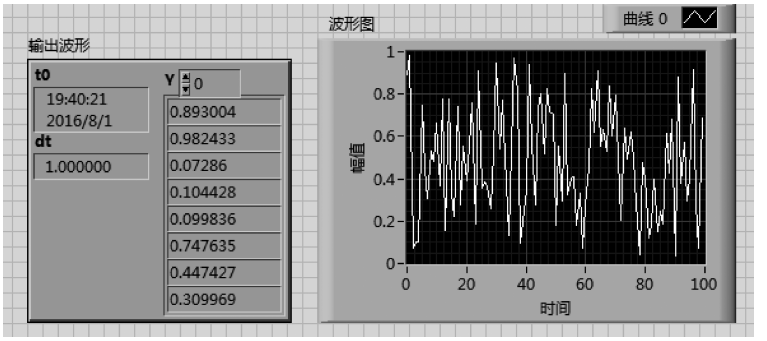


图 5.27　例 5.13 VI 的前面板

【例 5.14】　生成一段正弦波形，要求其频率为 50Hz，幅值为 2，初相位为 60°。

这个例子的 VI 的程序框图如图 5.28 所示，其前面板见图 5.29。对该 VI 需要说明的有：①它调用了“正弦”函数，此函数经“函数”选板→“数学”→“初等与特殊函数”，在“三角函数”子选板上可以找到；②幅值输入控件中的数值是单个值，将其乘以 For 循环生成的数组，即幅值输入控件中的数值将依次与数组中的每个元素相乘；③正弦波形的周期为其频率的倒数，波形中任意两个相邻数据点之间的时间间隔 dt 等于周期除以“点数/周期”。很容易理解，如果将 For 循环中的“正弦”函数换成其他函数，那该 VI 就可以产生相应函数随时间变化的波形。

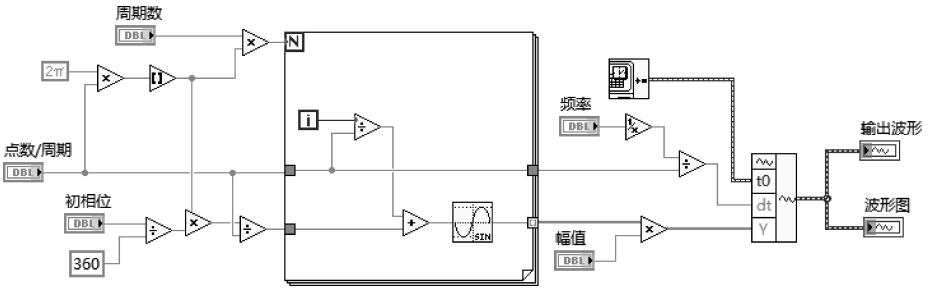


图 5.28　例 5.14 VI 的程序框图

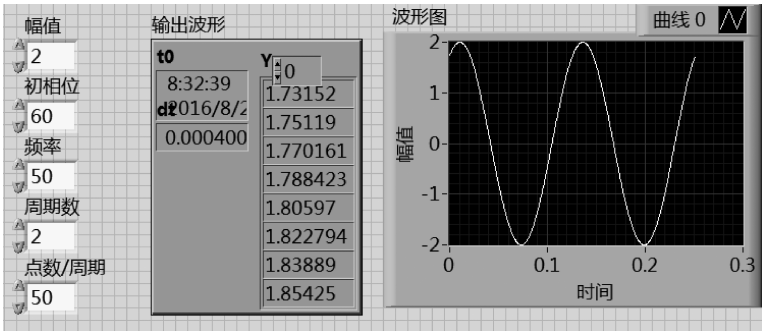


图 5.29　例 5.14 VI 的前面板

在例 5.14 中，通过调用一个 For 循环和一个“正弦”函数，再通过一些运算，就得到了一段正弦波形。由于在构建测试系统时经常需要生成仿真信号，所以，LabVIEW 中提供有一系列典型的函数，利用它们，可以直接生成相应函数的波形。这些函数经“函数”选板→“信号处理”→“波形生成”子选板可以找到，如图 5.30 所示。

图 5.30 “波形生成”子选板

【例 5.15】 生成一段正弦波形，并获得它的波形成分。

例 5.15 的 VI 的程序框图如图 5.31 所示。可见，其中先调用“正弦波形”函数，以产生一段正弦波，然后，再利用“获取波形成分”函数将该正弦波形的各个成分提取出来，波形成分分别是 dt 和“数组 Y”。

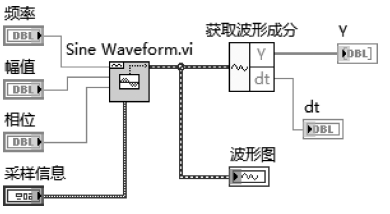


图 5.31 获取波形成分示例 VI 的程序框图

要产生正弦波形，需要设置以下几个参数：①频率；②幅值；③相位；④采样信息。其中，前三个参数很容易理解，下面重点介绍一下采样信息。

采样信息是一个簇类型的数据，它包含了两个元素，分别是采样率和样本数。在生成仿真信号时，采样率是指 1s 时间内生成多少个数据；而样本数，则是指一共生成多少个数据；这两个数据配合起来，就是生成数据的时间长度，即“样本数/采样率”秒。在图 5.32 所示的该 VI 的前面板上可以看到，设置的采样率是 1000，样本数是 1000，如此，就会生成 1s 的数据。请注意，

由于现在只是在计算机中生成了一段仿真信号，所以，虽然波形图的横轴显示的是时间，但却没有实际意义。运行此 VI 后会发现，1s 的数据瞬间就生成了。只有当将这段仿真信号输出到计算机外，例如，用示波器去观察这段信号波形时，才会感受到信号波形的时间长度，这时，时间长短就有意义了。本教材将在第 8 章学习如何将仿真信号输出到计算机外，变成真实世界中的信号。

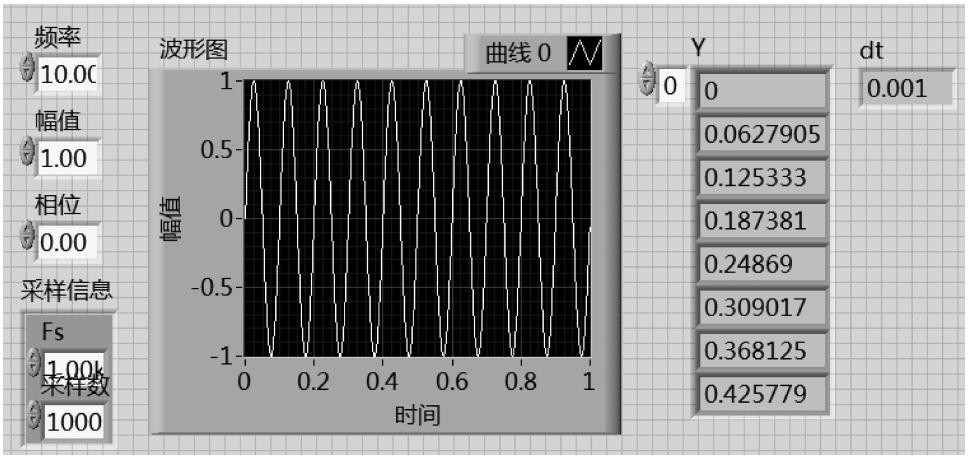


图 5.32　获取波形成分示例 VI 的前面板

除上述外，在“波形”子选板上，还提供有很多波形函数，且还有不少用于实现波形测量和波形发生的子 VI，学习者可以在需要使用时自己选择。其中，一些波形函数较为简单，可在框图上双击其函数图标，打开它的对应 VI 窗口，查看了解其内部的实现细节和原理。

另外，在实际的数据采集中，常常要从多个数据通道的每个通道中各采集一个波形。对此，数据采集函数输出的数据类型就是一个波形数组，即由波形数据作为元素组成的数组。对于波形数组，可以先使用数组函数，从该波形数组中提取出相关波形，然后再利用“获取波形成分”函数对各个波形分别进行处理。下面通过例 5.16 学习波形数组的生成和处理方法。

【例 5.16】　生成两路波形，一路是正弦波，另一路是方波，并提取出各自的波形成分。

例 5.16 的 VI 的前面板和程序框图分别如图 5.33 和图 5.34 所示。可见，该 VI 分别调用了“正弦波形”和“方波波形”函数，产生了两路波形。该 VI 还调用了“创建数组”函数，这样，生成的数组的元素就是波形，即实现了波形数组的创建。波形数组中存放的波形数据由波形图控件显示出来——从如图 5.33 所示的前面板上可以看出，生成了两路波形，一路是正弦波（白色曲线），另一路是方波（红色曲线）。

那么，如何获取这两路波形各自的波形成分呢？首先，应调用“索引数组”函数，输入索引号 0，则将波形数组中的第 0 个元素正弦波形提取出来，接下来，就可以再调用“获取波形成分”函数，以提取出正弦波形的数组 Y 和 dt。

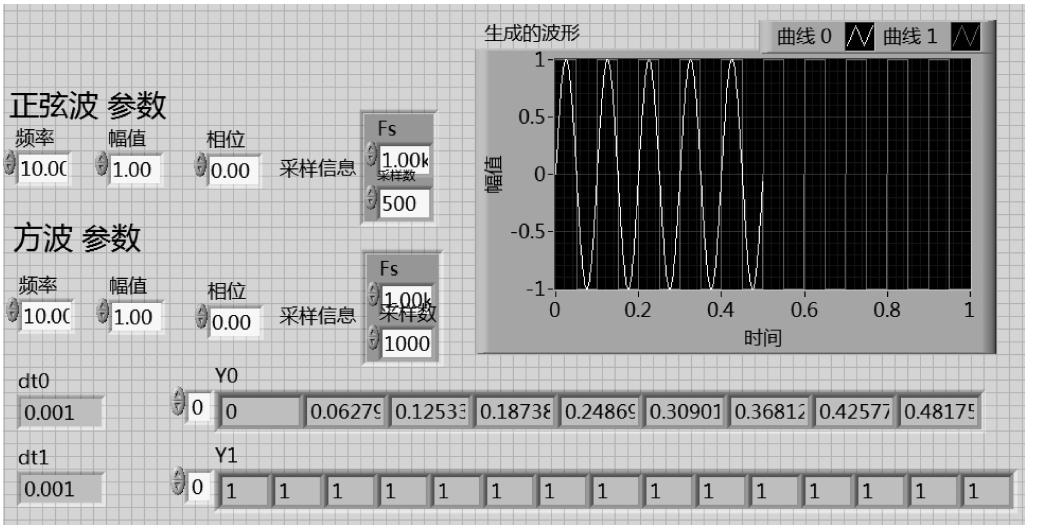


图 5.33　例 5.16 VI 的前面板

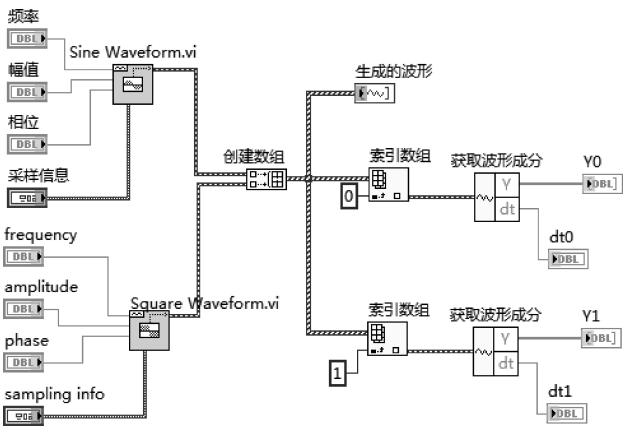


图 5.34　例 5.16 VI 的程序框图

5.4　DDT

DDT 即所谓动态数据类型。DDT 是专门针对 ExpressVI 设计的。DDT 的函数位于“函数”选板→“Express”→“信号操作”子选板上，常用的 DDT 函数有“合并信号”“拆分信号”“从动态数据转换”和“转换至动态数据”等。通常，对于动态数据类型 DDT，可以利用“从动态数据转换”函数，将 DDT 数据转换成波形或数组等，然后，再利用波形和数组函数对相应的数据进行分析处理。

常见问题 24：什么是 Express VI?

Express VI 是 LabVIEW 中按某个目标将一些基本函数或函数模块整合在一起的功能函数单元，利用它，可为用户提供更方便、更简捷的编程途径，因此这类 VI 得名“快速 VI”即“Express VI”。从函数图标的外观看，与前述的基本函数不一样，这类 VI 即函数的图标有专门的蓝色底框的标识，如图 5.35 所示就是一个“仿真信号正弦”Express VI 的图标。另外，在函数选板上，还有专门的“Express”子选板。

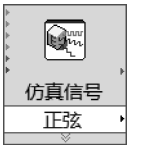


图 5.35　Express VI“仿真信号正弦”的图标

把某个 Express VI 刚放到程序框图面板上，配置该 Express VI 的对话框就会自动打开，用户可以按照自己的需求，以交互方式配置该 Express VI 的属性。图 5.36 所示的是“仿真信号正弦”Express VI 弹出的对话框，在此界面上可以进行参数设置，而且配置结果可从对话框的“结果预览”框中查看。单击“确定”按钮，就完成了参数的设置。之后，若用户希望再修改或调整该 Express VI 的参数配置，可双击已放置在程序框图面板上的该 Express VI 的图标；或右键单击它，在弹出的快捷菜单中选择“属性”，即再次打开其“属性配置”对话框，对相关参数重新进行设置。

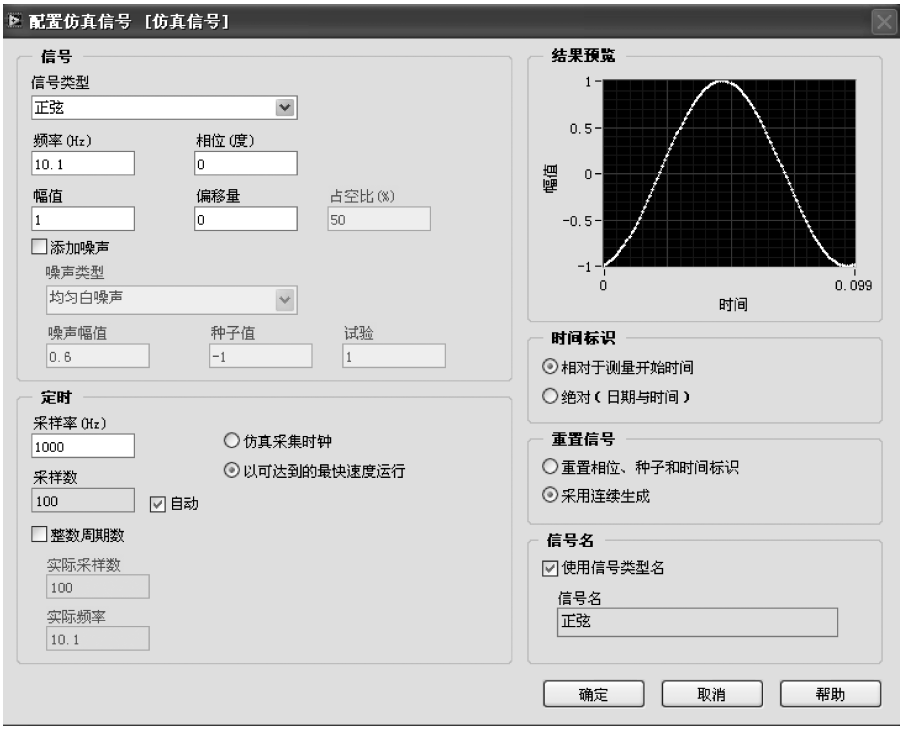


图 5.36　Express VI“仿真信号正弦”的参数设置界面

很明显，在编写 VI 中使用 Express VI，可减少连线、简化框图、凸显所编写 VI 的主脉络。但是，Express VI 具有简便、易用等优点的同时，也丧失了一些功能和灵活性。因此，若想得到一个功能更为强大、更体现使用者个性化特点的应用程序即 VI，还是应该更多地选用 LabVIEW 提供的基本函数。

【例 5.17】 将动态数据类型 DDT 转换成波形或数组。

在此 VI 中，首先调用 Express VI“仿真信号正弦”，生成一段正弦波形，在前面板，用一个波形图控件显示出来，此 VI 的程序框图如图 5.37 所示，其前面板如图 5.38 所示（有关波形图控件的使用，将会在本教材第 7 章做具体介绍）。

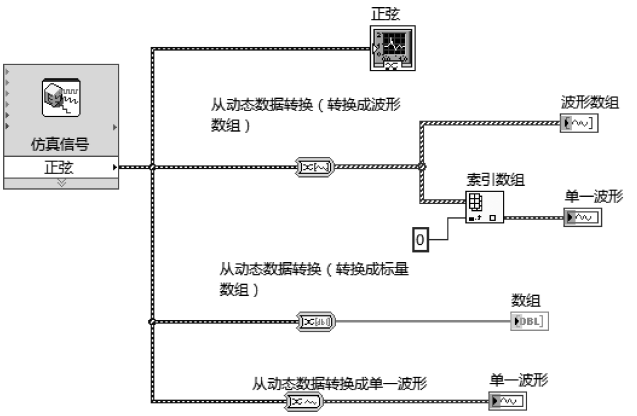


图 5.37 例 5.17 VI 的程序框图

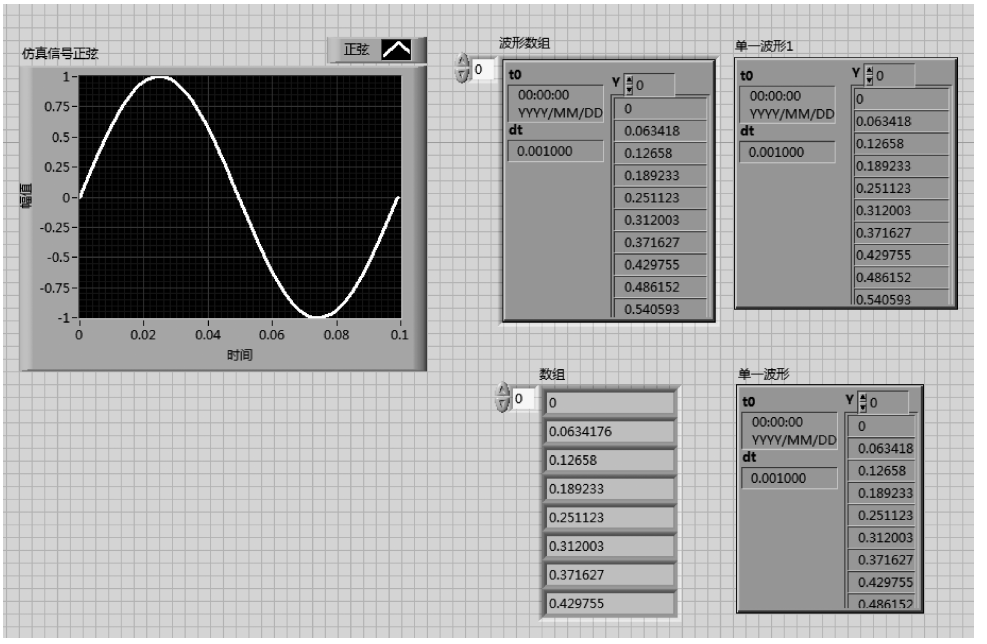


图 5.38 例 5.17 VI 的前面板

可以看出，Express VI“仿真信号正弦”生成的 DDT 数据在连线上使用粗的蓝色绞线表示。要想对 DDT 数据进行处理，一般要通过“从动态数据转换”函数，将 DDT 数据转换成数组或者波形。在图 5.37 所示的该 VI 的程序框图上，调用的第 1 个“从动态数据转换”函数的作用，是将 DDT 数据转换成一维的波形数组，转换设置界面如图 5.39 所示。转换成一维数组后，要先调用索引数组函数，将波形元素从数组中提取出来，然后再利用有关波形方面的功能函数，对此单一波形做进一步处理。调用的第 2 个“从动态数据转换”函数，是将 DDT 数据转换成一维的标量数组，可以看出，得到的就是波形的数组 Y 成分。由于此例只产生了一路信号，因此，调用的第 3 个“从动态数据转换”函数，就是将 DDT 数据转换成单一波形，可以看出，得到的就是波形。在使用“从动态数据转换”函数时，要根据实际需求选择将 DDT 数据转换成所需类型的数据。

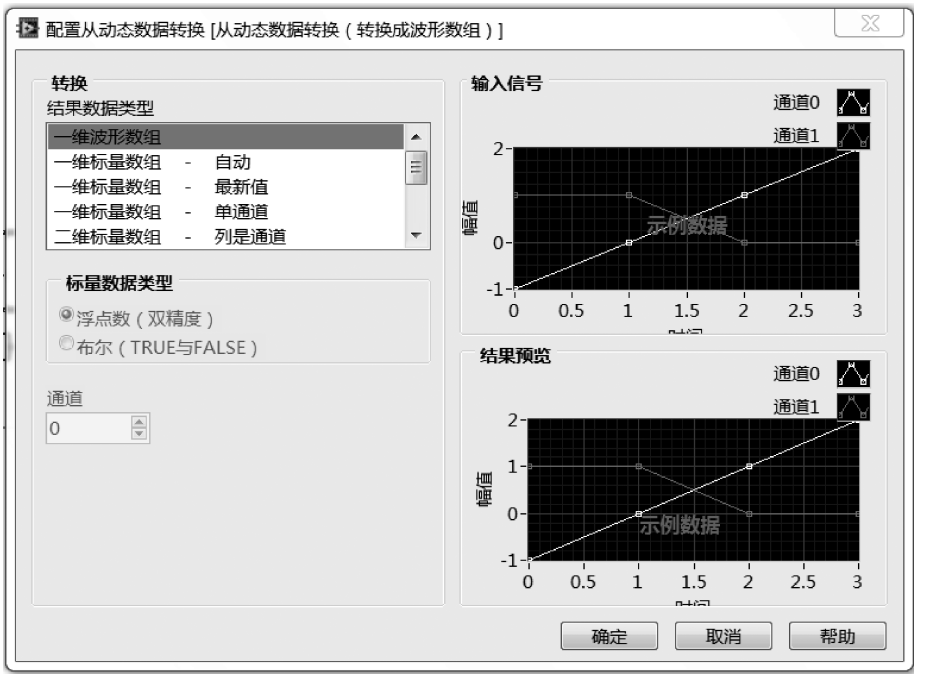


图 5.39　“从动态数据转换”函数的对话框

【例 5.18】　生成两路正弦信号并显示出来。

例 5.18 给出了“合并信号”函数的使用示例，编写的 VI 的程序框图和前面板如图 5.40 所示。可见，该 VI 先两次调用了 Express VI“仿真信号正弦”，然后，将生成的数据连至“合并信号”函数的输入端上，并将“合并信号”函数的输出数据连至波形图控件上。运行此 VI 可以看出，在波形图控件上显示出了两路正弦信号，幅值分别是 1 和 2。

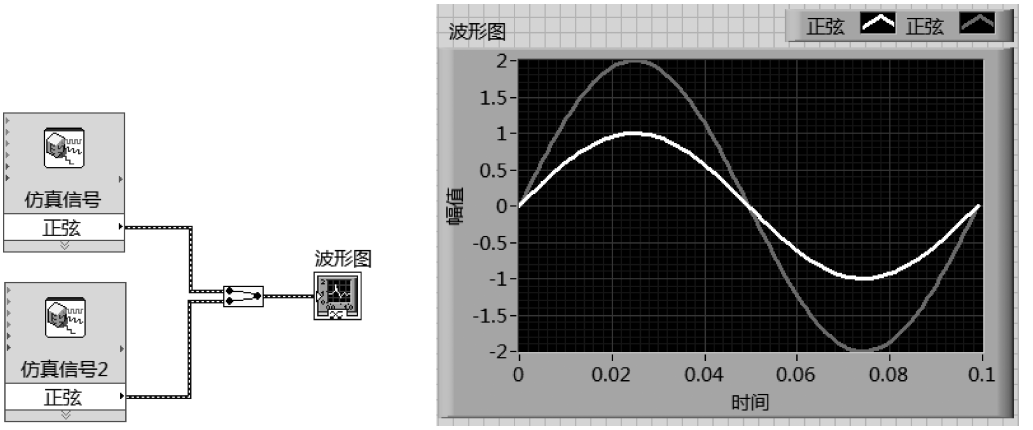


图 5.40　利用快速 VI 生成两路正弦信号并显示出其波形

5.5　变体

变体数据类型，是 LabVIEW 中多种数据类型的容器。将其他数据转换为变体时，变体将会存储数据和数据的原始类型，以保证日后还能将变体数据反向转换为原始数据类型的数据。例如，如果将字符串数据转换为变体，变体将存储字符串的文本，以及说明该数据是从字符串（而不是路径、字节数组或其他 LabVIEW 数据类型）转换而来的信息。

变体数据类型的一大优势是：利用变体数据，VI 可以采用通用的方式处理不同类型的数据。例如，在使用队列消息处理器设计模式时，其消息数据可能是多种数据类型的，可能传递的是数组，也可能是波形或者图像，等等。如果为每种数据类型各写一个 VI，这样就有多个副本的 VI，如果将来程序有变动，如此较难维护。对此，采用变体数据就是一种好的解决方案。

变体数据类型的另一个优点是：可存储数据的属性。属性数据可以是任意数据类型。例如，在 5.3.1 节中介绍到的波形，它的一个成分“属性”就是变体类型，可用于携带任意的属性信息。

变体输入控件位于“控件”选板→“新式”→“变体与类”子选板上。在前面板上拖曳出一个变体输入控件，然后，在程序框图上选中此变体输入控件，右击鼠标，在弹出的快捷菜单中选择“转换为常量”，会在程序框图上生成一个变体常量；而如果选择“转换为显示控件”，则会在程序框图上生成一个变体显示控件。

LabVIEW 中有关变体的函数，都位于“函数”选板→“编程”→“簇、类与变体”→“变体”子选板上，如图 5.41 所示。常用的变体函数请参见表 5.4。

图 5.41　变体子选板

表 5.4　常用的变体函数

序号	名　　称	图　　标	功 能 说 明
1	转换为变体	任何数据 变体	转换任意 LabVIEW 数据为变体数据。也可用于使 ActiveX 数据转换为变体数据
2	变体至数据转换	类型 变体 错误输入 数据 错误输出	转换变体数据为 LabVIEW 可显示或可处理的数据类型。也可用于使变体数据转换为 ActiveX 数据
3	获取变体属性	变体 名称 默认值（空变体） 错误输入 变体副本 找到 值 错误输出	获取所有属性的名称和值，如果连接了名称参数，则返回该属性的值
4	设置变体属性	变体 名称 值 错误输入 变体输出 替换 错误输出	用于创建或改变变体数据的属性或值
5	删除变体属性	变体 名称（所有属性） 错误输入 变体输出 找到 错误输出	删除变体数据中的属性和值

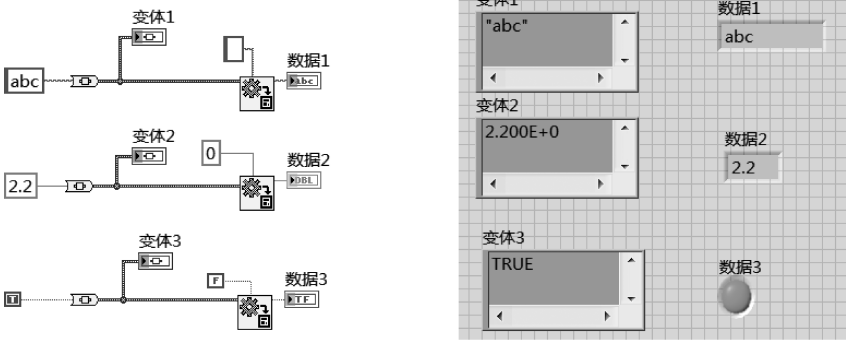
下面将通过例 5.19～例 5.22 学习在 LabVIEW 中如何使用变体。

【例 5.19】　变体与基本数据类型的转换。

例 5.19 VI 的程序框图和前面板，如图 5.42 所示。本例将把三种基本数据类型（字符串、数值和布尔量）转换为变体，然后，再将变体转换为基本数据类型。

在本例中，最上面是将字符串转换成变体，通过调用“转换为变体”函数实现；然后再调用“变体至数据转换”函数，将该变体转换成字符串类型的数据。需要注意的是，在使用“变

体至数据转换”函数时，应根据实际情况，在该函数的“类型”输入端上接入正确的数据类型。例如本例中，对于原始数据是字符串类型的，应在“变体至数据转换”函数的“类型”输入端上接入字符串常量(可以是任意字符串，其值没有意义，只要保证数据类型正确即可)。很容易理解，对于原始数据是数值型的，需在“类型”输入端上接入数值常量；而对于原始数据是布尔型的，则需在“类型”输入端上接入布尔常量。

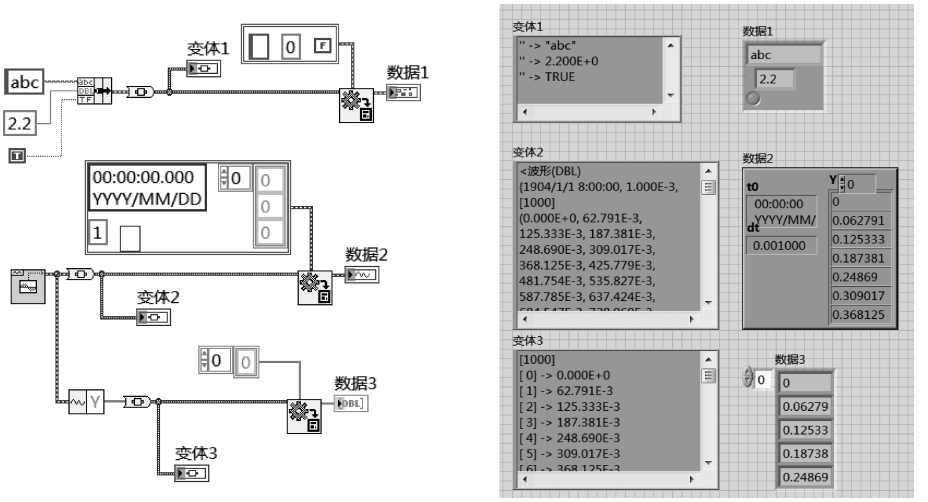


(a) 程序框图　　(b) 前面板

图 5.42　例 5.19 VI 的程序框图和前面板

【例 5.20】 变体与复合数据类型的转换。

例 5.20 的 VI 的程序框图和前面板，如图 5.43 所示。在本例中，分别将簇、波形和数组转换成变体(调用“转换为变体”函数实现)，然后，再将变体转换为原始数据(调用“变体至数据转换”函数实现)。



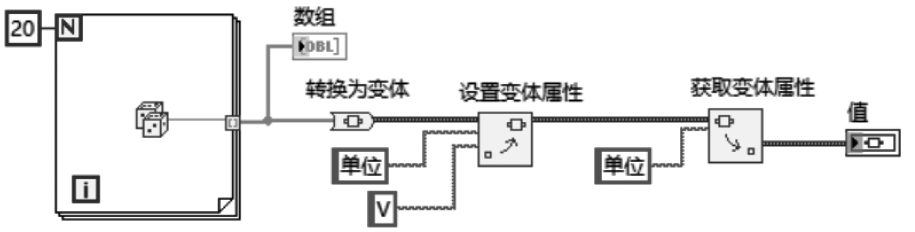
(a) 程序框图　　(b) 前面板

图 5.43　例 5.20 VI 的程序框图和前面板

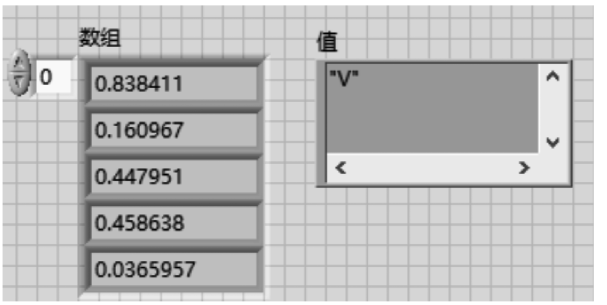
从例 5.19 和例 5.20 可以看出，要想正确使用“变体至数据转换”函数，需要知道变体数据存储的原始数据的类型。在实际编程时，要将原始数据的类型连接到“变体至数据类型转换”函数的“类型”输入端上。

【例 5.21】 生成一个一维的随机数组，该数组的单位是 V(电压的单位“伏特”)。

例 5.21 的 VI 的程序框图和前面板，如图 5.44 所示。在本例中，先调用 For 循环，生成了一个一维随机数组；然后，调用“转换为变体”函数，将该随机数组转换为变体；再调用“设置变体属性”函数，设置该数组的“单位”是“V”；最后，调用“获取变体属性”函数，将“单位”的值提取出来，并显示在前面板上。



(a) 程序框图



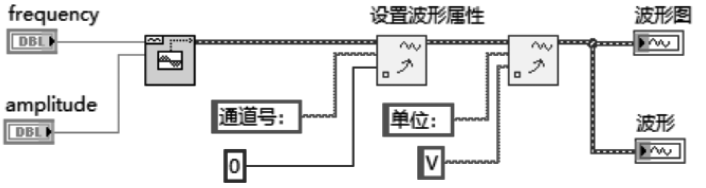
(b) 前面板

图 5.44　例 5.21 的 VI 的程序框图和前面板

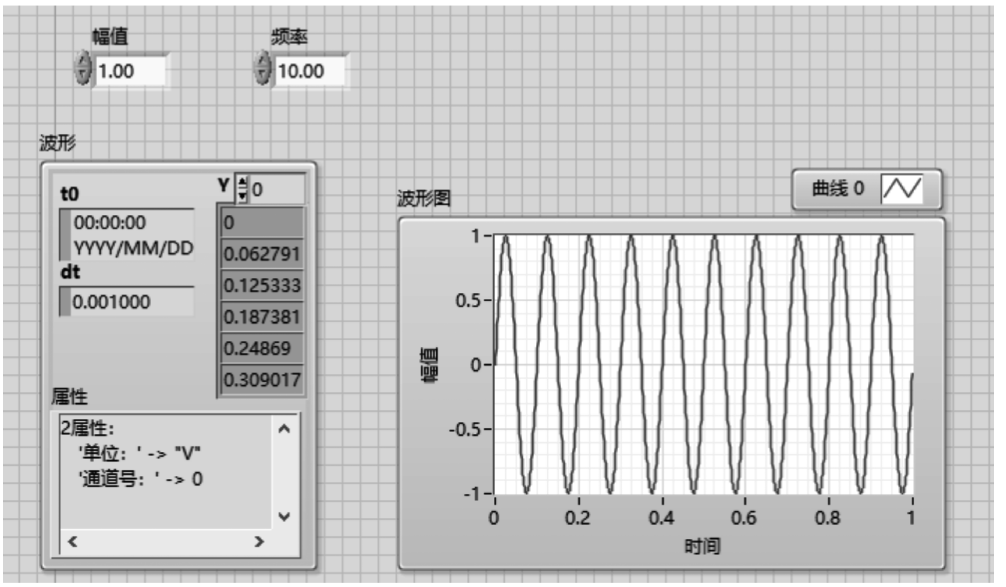
在第 5.3 节介绍波形时，曾说到波形的一个成分“属性”就是变体数据类型。下面，将调用“设置波形属性”函数对波形的“属性”进行设置，具体实现，请见例 5.22。

【例 5.22】 生成一段正弦波，其幅值为 1V，频率为 10Hz，该波形是从模入通道 0 采集进来的，单位是 V。

例 5.22 的 VI 的程序框图和前面板，如图 5.45 所示。本例中，先生成一段正弦波，然后调用“设置波形属性”函数，分别设置该波形的“通道号”为“0”，“单位”为“V”，最后用波形图控件和波形显示控件，将波形在前面板显示出来。



(a) 程序框图



(b) 前面板

图 5.45　例 5.22 VI 的程序框图和前面板

5.6　本章小结

本章学习了 5 种复合数据类型，即：数组、簇、波形、DDT 和变体。从学习相应知识过程中可以感受到，对这些数据类型的创建都有两种方法。以数组为例，一种方法是按部就班地先建框架，再添加元素，然后赋值；另一种方法是通过编程来实现，例如，通过利用循环的自动索引功能或者利用 LabVIEW 提供的某个函数来实现。对初学者而言，第一种方法可以帮助大家建立起 LabVIEW 中数组的基本概念；而第二种方法在实际编程中用得更多，因为通过这种方式，可以更高效地完成虚拟仪器 VI 的设计和编程。

本章习题

5.1　构建一个 VI，将有 10 个随机数的一个数组的元素排列顺序颠倒过来，之后，再将数组的后 5 个元素按顺序移到数组的前端，形成一个新的数组。

5.2　创建一个簇控件，其元素分别为字符串型控件“姓名”，数值型控件“学号”，布尔型控件“是否注册”；从该簇控件中提取出元素“是否注册”，将其显示在前面板上。

5.3　生成一个数组，要求该数组中的元素个数为10，每个元素均为偶数，元素值大于或等于0且小于或等于20，并在前面板上显示出所生成的数组。

5.4　生成一段三角波形，并将其在前面板上显示出来。要求其幅值、频率和采样信息等参数均可调节。改变采样信息中的采样率和样本数，观察其波形的相应变化。

5.5　生成三路信号波形，分别是正弦波、方波和锯齿波，将它们显示在同一个波形图中。要求正弦波的频率为10Hz，幅值为2，总时间长度为1s；方波的频率为20Hz，幅值为3，总时间长度为2s；锯齿波的频率为40Hz，幅值为5，总时间长度为2s。提示：注意采样信息中具体的采样率和样本数的设置。

5.6　生成一段正弦波形，要求其幅值、频率、相位等参数均可调。提取出该正弦波形中的Y数组和dt成分，并求出Y数组中元素的最大值以及个数，且还要将结果在前面板上显示出来。

参考文献

[1] Johnson G W，Jennings R. LabVIEW 图形编程[M]. 武嘉澍，陆劲昆，译. 北京：北京大学出版社，2002.

[2] 侯国屏，王珅，叶齐鑫. LabVIEW 7.1 编程与虚拟仪器设计[M]. 北京：清华大学出版社，2006.

[3] 黄松岭，吴静. 虚拟仪器设计基础教程[M]. 北京：清华大学出版社，2008.

[4] Bishop R H. LabVIEW 8 实用教程[M]. 乔瑞萍，林欣，译. 北京：电子工业出版社，2008.