# 第 3 章

# LabVIEW 基本数据类型

本章将介绍 LabVIEW 中基本的数据类型，包括数值、布尔和字符串。在介绍每种数据类型的过程中，分别介绍数据类型的概念、基本使用方法以及在 LabVIEW 中使用这种数据类型进行操作的函数节点和函数。

在 LabVIEW 中对数据类型的运算有一些与其他文本编程语言不同的地方，如数值的自动类型转换、布尔控件的机械动作，在每种数据类型的内容中都将进行讲解。

针对布尔数据类型，通过密码锁的实例讲解布尔数据类型的运算。

针对字符串数据类型，通过分析串口设备通信的实例，讲解和实践串口数据解析，实现串口数据的运算。


微课视频

## 3.1 数值

### 3.1.1 数值数据类型的概念

数值数据类型是 LabVIEW 中使用最广泛的数据类型，也是许多复杂数据类型组成的基本元素。数值数据类型最大的特点就是可以用来直接计算，大部分算法都是针对数值数据类型的。

一般来说，在 LabVIEW 中可以进行数学运算的单个元素，都归为数值数据类型。例如，整数、小数、复数等，这些都属于数值数据类型。

**1. 在前面板中放置数值控件**

在前面板空白处右击，打开“控件”选板，选择“新式”选板→“数值”选板，如图 3.1 所示，可以看到“数值”选板中根据不同的应用和外观罗列了多种数值类型的控件。

在“数值”选板中包含了输入控件和显示控件。在“数值”选板中单击选中相应的控件后，在前面板需要放置控件的位置再次单击，就可以放置控件。

**2. 在程序框图中放置数值常量和控件接线端**

1）在程序框图中放置数值常量

在程序框图空白处右击，打开“函数”选板，选择“编程”选板→“数值”选板，打开“数值”

选板后，其中的子面板提供了不同数值类型的常量，如图 3.2 所示。

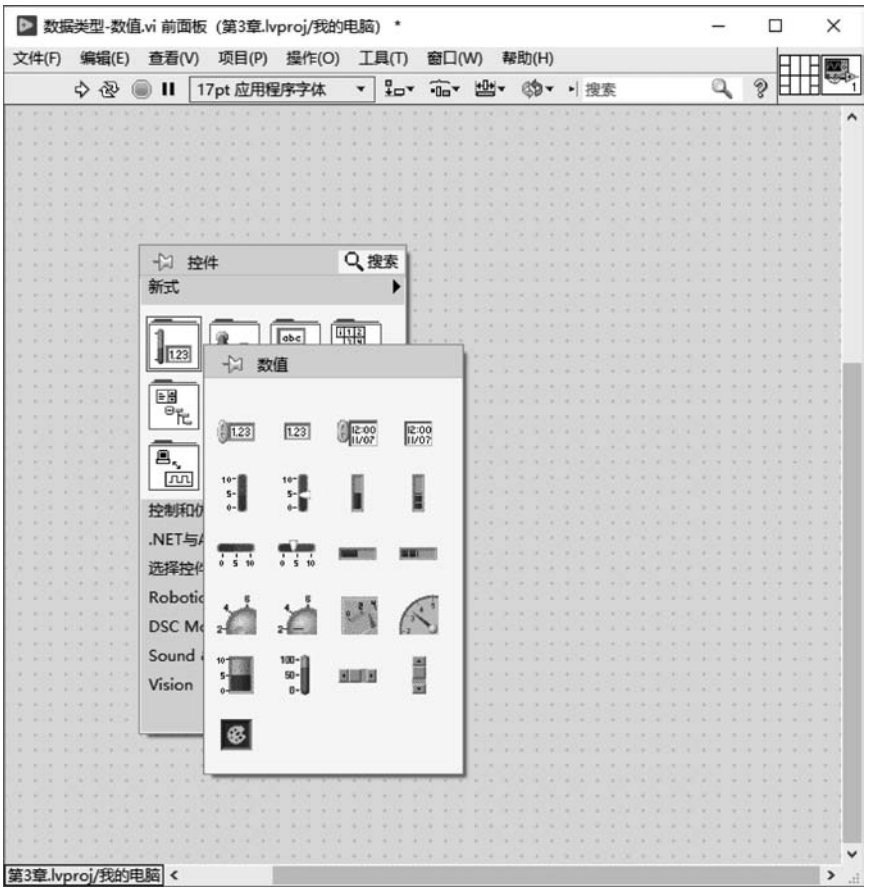


图 3.1　数值控件

一般来说，从前面板添加的数值类型输入控件和显示控件用来与用户进行数据交换。在打开 LabVIEW 程序文件时，前面板的数值输入控件都会被初始化为默认值，如果在程序执行的过程中数值不需要进行改变，一般将这些数值类型的元素作为常量添加在程序框图中，用于程序内部的算法计算。

2）在程序框图中放置数值接线端

在程序框图中，将鼠标指针放置在节点或函数节点的接线端上，通过“创建”→“输入控件”或“创建”→“显示控件”命令添加输入或输出的接线端。

这个过程等同于在前面板添加输入或显示控件，不同的是在程序框图中通过右击添加的接线端可以自动匹配数据类型。如果节点的接线端是复杂数据类型，通过这种方式可以避免数据类型不匹配带来的编译错误。

**3．添加数值控件实例**

在 LabVIEW 中添加数值控件的具体操作步骤如下。

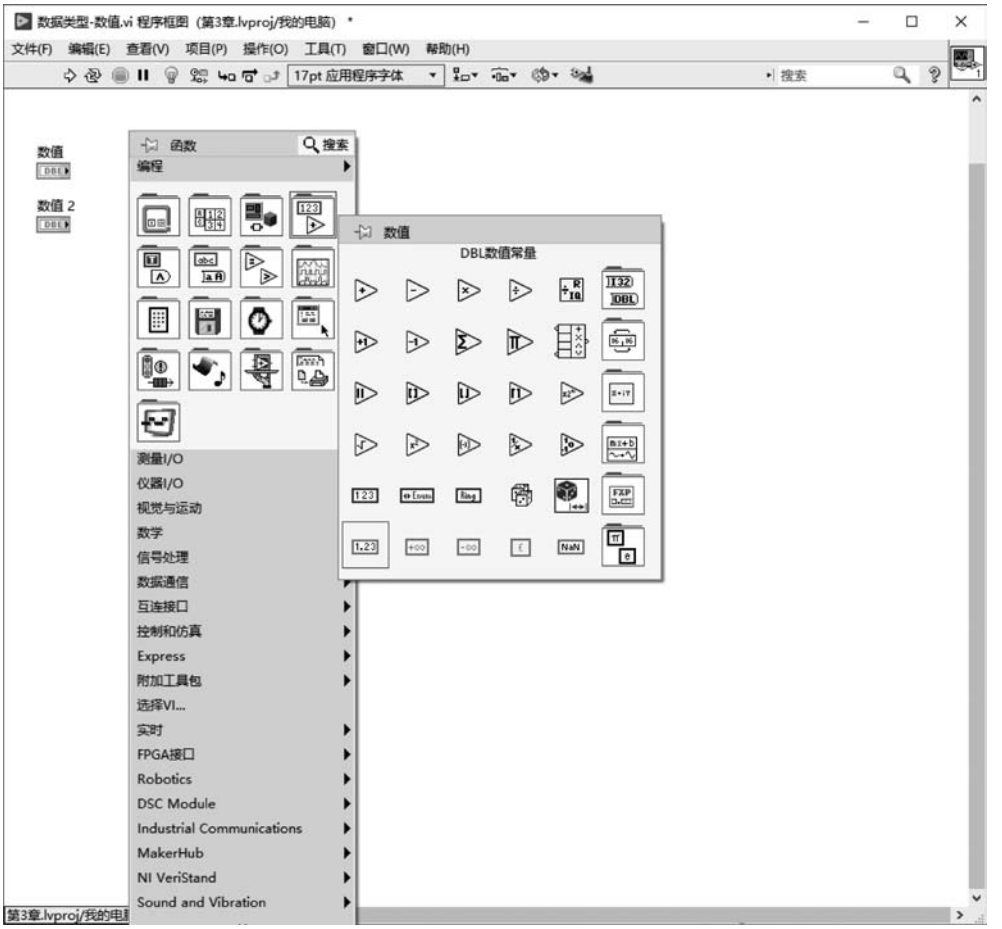


图 3.2　不同数值类型的常量

(1) 在 LabVIEW 菜单栏中选择“文件”→“新建 VI”命令,在打开的空白 VI 文件窗口的菜单栏中选择“文件”→“保存”命令,设置文件名为“数据类型-数值”。

(2) 在“数据类型-数值”VI 前面板空白处右击,打开“控件”选板,选择“新式”选板→“数值”选板,选择“数值输入控件”,然后在前面板中再次单击,可以看到前面板中放置了一个标签为“数值”的输入控件。

(3) 再次重复步骤(2),在前面板中放置一个新的数值输入控件,并且将新的控件放置在前一个控件的正下方。可以看到前面板中新增加了一个标签为“数值 2”的输入控件。

在向前面板添加控件时,LabVIEW 会自动为添加的控件添加标签,标签的内容和添加控件的数据类型有关。当多次添加同一种类型的控件时,LabVIEW 会自动增加标签中的序号,如添加的第二个数值类型的输入控件的标签名为“数值 2”。

(4) 在前面板空白处右击,打开“控件”选板,选择“新式”选板→“数值”选板,选择“数值

显示控件”，然后在前面板中再次单击，可以看到前面板中放置了一个标签为“数值 3”的显示控件。

如图 3.3 所示，可以看到数值输入控件和数值显示控件是被认为同一种类型的控件标记序号的，所以标签为“数值 3”。


图 3.3 在前面板中添加数值输入控件和显示控件

一般来说，输入控件和显示控件的颜色是不同的。输入控件是白色的，对应同样数据类型的显示控件是灰色的。

输入控件和显示控件可以相互转换。例如，在前面板中右击数值输入控件，在弹出的菜单中选择“转换为显示控件”，标签为“数值”的输入控件就变成了显示控件。在这个实例中，保持“数值”为输入控件。

(5) 在程序框图中可以看到在前面板中添加的“数值”输入控件、“数值 2”输入控件、“数值 3”显示控件对应的接线端，在空白处右击，打开“函数”选板，选择“编程”选板→“数值”选板，选择“DBL 数量常值”，然后在程序框图中的“数值 2”接线端下方再次单击，将 DBL 数量常值放置在程序框图中，如图 3.4 所示。

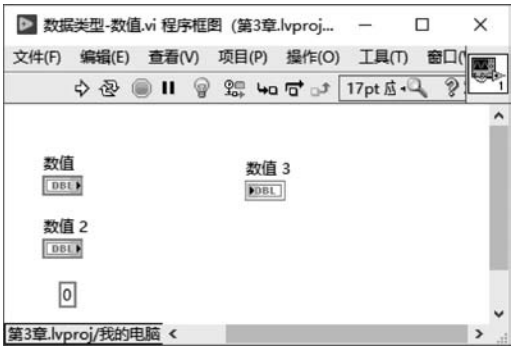


图 3.4 在程序框图中添加数值类型的常量

在程序框图中添加常量时，会自动赋值为这个数据类型的默认值。例如，LabVIEW 中数值类型的默认值为 0。

## 3.1.2　数值类型的表示法

数值类型的表示法是指数值在内存中的存储方式，不同的数值类型在内存中存储的方式和长度是不同的。LabVIEW 和其他的编程语言类似，数值类型分类主要有以下几种。

（1）无符号整型（U8，U16，U32，U64），如 1，2，3。

（2）有符号整型（I8，I16，I32，I64），如－2，－1，1，2。

（3）浮点型（SGL，DBL），如－1.234，1.234。

在使用数值进行运算的时候，当涉及数值的不同类型时，可以在数值控件上右击，在弹出的菜单中选择“表示法”，选择具体的数值类型。

**1. 修改控件的表示法实例**

打开“数据类型-数值”VI 文件，在前面板中右击标签为“数值”的输入控件上，在弹出的菜单中选择“表示法”→I16，如图 3.5 所示。

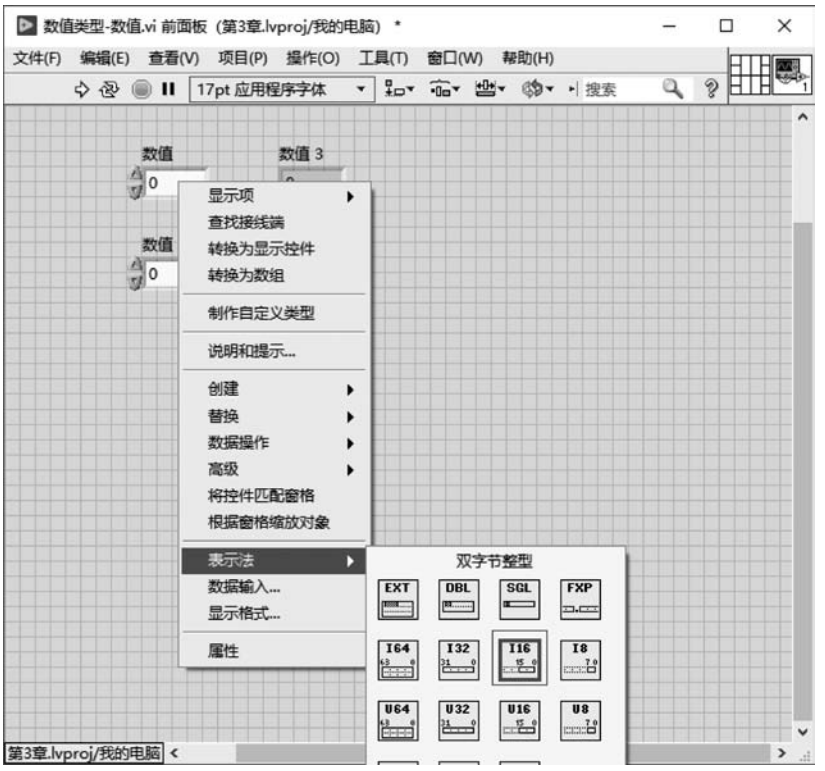


图 3.5　修改控件的表示法

保存“数据类型-数值”VI 文件。

**2. 不同编程平台的数据类型映射**

不同编程平台中,对于数据类型的定义基本相似,存在一定的映射关系。在不同编程平台进行数据通信的时候,需要尤其注意数据类型是否匹配。

这里以 LabVIEW 与 C 语言为例,列举了常用的数据类型比较,如表 3.1 所示。

**表 3.1 LabVIEW 与 C 语言的常用数据类型比较**

| LabVIEW | C 语言 |
|---|---|
| DBL | double |
| SGL | float |
| I32 | int |
| 字符串 | char |

如果想了解更加详细的信息,可以打开 NI 范例查找器中的“执行外部代码(DLL)”范例,查看 LabVIEW 中定义的数据类型与 C 语言中定义的数据类型的对应关系,如图 3.6 所示。


图 3.6 LabVIEW 和 C 语言中不同数据类型对应关系的范例

## 3.1.3 数值数据的运算

数值数据类型相关的函数和函数节点都在程序框图中的“数值”选板中。在程序框图空白处右击,打开“函数”选板,选择“编程”选板→“数值”选板,可以看到 LabVIEW 中提供的用于数值类型运算的函数节点,如图 3.7 所示。

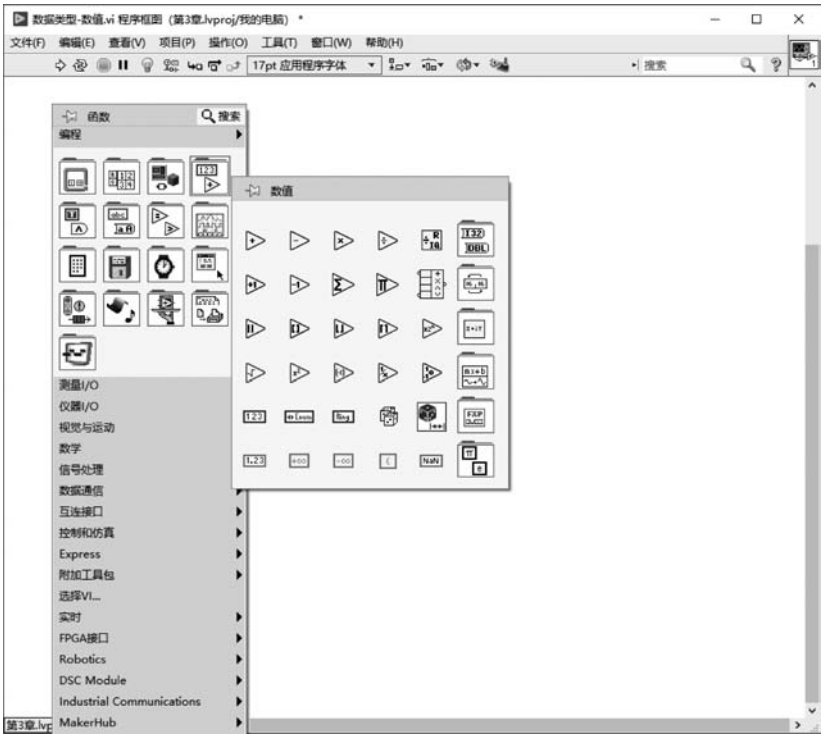


图 3.7　用于数值类型运算的函数节点

接下来通过实现摄氏度到华氏度转换的实例讲解数值运算的使用方法，具体操作步骤如下。

（1）打开“数据类型-数值”VI 文件，在程序框图中可以看到已经放置的接线端和常量，其中表示法分别是：数值类型为 I16，标签为“数值”的接线端；数值类型为 DBL，标签为“数值 2”的接线端；数值类型为 DBL，标签为“数值 3”的接线端；数值类型为 DBL 的常量，如图 3.8 所示。

（2）在程序框图中通过拖动鼠标，将输入控件和常量的接线端排布在左侧，将输出控件的接线端排布在右侧。

（3）在程序框图空白处右击，打开“函数”选板，选择“编程”选板→“数值”选板，选择“乘”和“加”函数节点，添加到程序框图中，如图 3.9 所示。

添加完成后，注意这时程序框图工具栏中的“运行”按钮是断线的状态，表示当前编译错误。错误原因是此时运算节点缺少输入数据，因为还没有将输入控件的接线端和常量与函数节点的接线端连接起来。

（4）按照摄氏度转换到华氏度的计算公式，实现数值的计算过程，如图 3.10 所示。

$$F = C \times 1.8 + 32 \tag{3.1}$$

其中，$F$ 为华氏度(℉)；$C$ 为摄氏度(℃)。


图 3.8 “数据类型-数值”VI 程序框图中的接线端和常量


图 3.9 添加“乘”和“加”函数节点

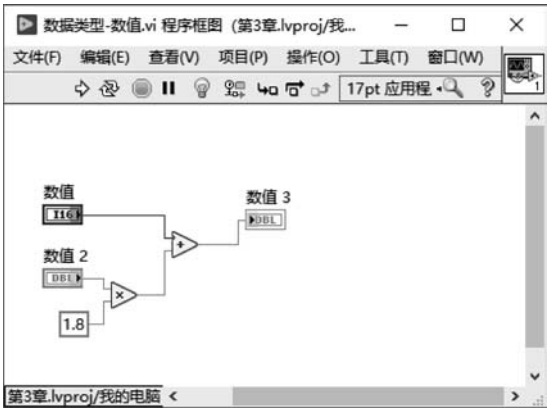


图 3.10 实现摄氏度到华氏度的转换

(5) 在前面板中单击“数值”输入控件,输入 32。因为式中的值 32 在计算过程中是不会改变的,所以也可以将“数值”输入控件改变为常量的数据类型。

(6) 在前面板中单击“数值 2”输入控件,输入 36。单击“运行”按钮,可以看到程序计算 36℃对应的华氏度为 96.8℉,如图 3.11 所示。


图 3.11　运行结果

## 3.1.4　数值数据的显示和可视化

LabVIEW 提供了非常丰富的方式以显示和观察数据。例如,进行数值显示的时候,可以有数值显示窗口、仪表码盘、进度条、波形图表等多种形式,对于人机交互的界面十分友好和方便。同时,LabVIEW 也提供了丰富的数据可视化工具,便于程序调试。

**1. 标签和标题**

1) 标签

标签是程序框图和前面板中控件的标识符号,通过标签索引和操作控件,所以标签是唯一的,不允许重复。

在前面板中添加控件的时候,默认显示的是控件的标签。在实例“数据类型-数值”VI 中,输入控件“数值”“数值 2”和显示控件“数值 3”分别是输入控件的标签和输出控件的标签,如图 3.12 所示。在前面板和程序框图中,每个控件的标签都是一致的。

一般添加控件的标签名称都是“数据类型+序号”的形式。例如,数值类型标签的默认名称形式是“数值+序号”,序号是根据当前数值类型控件数量依次递增的。在程序设计中,这样的标签命名方式是不友好的,一般需要将标签修改为具有明确含义的名称。

2) 修改控件标签实例

“数据类型-数值”VI 实现了摄氏度到华氏度的转换。在前面板中,控件都使用了默认的名称,用户通过标签名称无法知道每个控件的真实物理含义。

修改标签名称的方法是在前面板中双击需要修改的控件的标签,标签会黑色高亮显示,输入新的标签名称即可。


图 3.12 “数据类型-数值”VI 前面板中的控件标签

接下来按照下面的规则对前面板的控件进行标签的重命名：将“数值”重命名为“计算常量”，将“数值 2”重命名为“摄氏温度值”，将“数值 3”重命名为“华氏温度值”。修改后的前面板如图 3.13 所示。


图 3.13 修改控件标签

修改完前面板控件的标签名称，程序框图中的对应控件接线端的标签名称也对应更新，如图 3.14 所示。

3）标题

标签是控件唯一的表示符号，具有唯一性和不可重复性。标题是控件的别名，可以重复命名。例如，多个控件可以有同样的标题。

在前面板中右击控件，在弹出的菜单中选择“显示项”，勾选“标题”，这时在前面板中的控件就会显示标题，如图 3.15 所示。与修改标签的方法相同，双击标题，当标题的背景黑色高亮显示时，输入新的名称就可以修改标题。

如果将前面板中“计算常量”输入控件的标题改为“摄氏温度值”，在前面板中会出现两个控件具有同样的标题“摄氏温度值”，但是在程序框图中，“计算常量”接线端的标签

并没有改变，这是因为在程序框图中，每个接线端都只使用标签作为唯一标识，如图 3.16 所示。


图 3.14　“数据类型-数值”VI 程序框图中标签名称对应更新


图 3.15　显示控件标题


图 3.16　前面板中控件的标题修改后程序框图中对应接线端的标签没有变化

4）标签和标题的使用实例

在 LabVIEW 程序设计中，绝大多数情况都是使用控件的标签，而非标题。进行控件命名的时候，一般只命名控件的标签。标签是控件在程序中的唯一标识，程序编写中需要索引控件的时候使用的是标签。控件的标题只在前面板中使用，是为了更好地标识控件的物理意义，更改标题对于程序的设计没有影响。

在一些情况下，前面板中会出现使用多个具有同样物理含义的控件。例如，项目中同时监测多个测量温度的节点，如“标签和标题”VI 中，需要在前面板中进行 10 个节点的摄氏度和华氏度的转换，使用标签就不得不在具体含义的后面不断添加序号，在前面板中就会出现“摄氏度 2”“摄氏度 3”“摄氏度 4”“摄氏度 5”这种标签名称，如图 3.17 所示。


图 3.17　包含多个同样物理含义控件的前面板

实际上，在界面中需要表示的是当前节点的实际物理含义，后面的注释序号（如“摄氏度 5”中的“5”）是没有意义的。在这种情况下，可以将标签改为标题，并且将标题统一为物理含义的名称，修改后的前面板如图 3.18 所示。

前面板中标题的这种重复命名并不会影响程序框图中程序设计对控件的引用，因为程序框图中的每个控件都是按照标签来标识和使用的。

**2. 控件外观**

LabVIEW 前面板中的“控件”选板提供了丰富的控件显示形式。例如，需要显示数值类型数据，前面板的“控件”选板→“新式”选板→“数值”选板中，提供了垂直填充滑动杆、垂直进度条、旋钮、仪表、液罐、温度计等多种控件形式，如图 3.19 所示。


图 3.18　使用标题命名的前面板

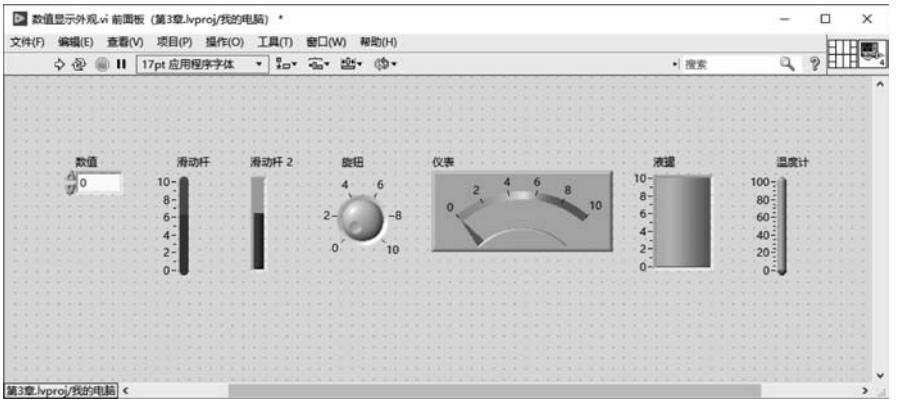


图 3.19　前面板中数值类型数据的多种控件形式

1）修改显示控件外观实例

友好的人机界面使用户可以快速了解程序的各种功能。在前面板中使用与物理含义匹配的控件可以使人机交互更加友好，因为图形符号会比文字更加直观和容易理解。接下来将“数据类型-数值”VI 中显示温度的“华氏温度值”控件形式修改为温度计，具体步骤如下。

（1）打开“数据类型-数值”VI 文件，在前面板中右击“华氏温度值”显示控件，在弹出的菜单中选择“替换”，如图 3.20 所示。


图 3.20　替换显示控件

（2）弹出“控件”选板，选择“新式”选板→“数值”选板→“温度计”显示控件，如图 3.21 所示。

（3）完成控件的替换后，程序框图中控件的接线端也会自动更新，并且保持替换前接线端的连线关系。更新完成后，不需要重新进行控件在程序框图中接线端的连线，在前面板工具栏中的“运行”按钮显示为白色箭头，代表当前程序编译没有错误，可以直接运行，如图 3.22 所示。

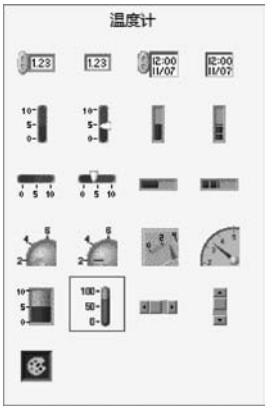


图 3.21　“温度计”显示控件


图 3.22　更新后的控件外观

2）使用“替换”选项修改控件外观的优点

在程序的设计中经常需要修改控件的外观。对于一个复杂的程序，如果在前面板中添加了控件，还需要在程序框图中对新增的控件进行连线操作。

使用“替换”选项就会容易一些。如果使用“替换”选项，更新控件的数据类型与原控件是一致的，如都是数值类型，那么更新的控件就不需要调整程序框图。更新后查看程序框图，会发现并没有变化，如图 3.23 所示。


图 3.23　修改控件外观后的程序框图

## 3.1.5　数值运算的类型转换

### 1. 出错的程序

如果仔细观察“数据类型-数值”VI 的程序框图会发现，在之前的程序设计中，在“加”函数节点处将不同类型的数据混合在一起进行了运算，“计算常量”接线端的数值类型是整型，“摄氏温度值”为双精度浮点型，“摄氏温度值”与 1.8 相乘的结果是双精度浮点型，再与“计算常量”相加时程序并没有报错，并且得到了一个双精度浮点型的“华氏温度值”输出值。

在一般的文本编程环境中，将不同数据类型放在一起计算的时候，编译器都会报错。但在 LabVIEW 中的数值类型的计算是允许这种情况的，并且可以计算出结果。虽然程序是一种“出错”的状态，但是编译器并不会报错，并且程序可以正常运行。

### 2. 自动数据类型转换

大多数的文本编程语言中不允许混合不同数据类型的数值进行计算，而在 LabVIEW 中遇到这种情况的时候，所有数值类的数据类型遵循一个向上转换的原则，也就是两个不同的数值类型会都转换成两者中精度更高的数据类型再进行计算。

例如，在“数据类型-数值”VI 中，在程序框图中“加”函数节点处不同类型的数值数据混合在一起计算，编译器会将整型和双精度浮点型的数值都自动转换成浮点数，然后计算结果输出。

如图 3.24 所示，“数据类型-数值”VI 程序框图中的“加”函数节点处，输入的是整型数值和双精度浮点型数值，输出的是双精度浮点型数值。LabVIEW 编译器在不同类型数值进行计算的时候做了自动数据类型转换，并且在“加”函数节点处标记了一个红点。程序框图中的红点代表编译器进行了自动数据类型转换。


图 3.24　编译器进行自动数据类型转换

**3. 自动数据类型转换的特点**

事实上，LabVIEW 编译器自动数据类型转换的好处是可以快速地进行算法的设计和得到系统的原型，而不必过多地关注底层数据的严格定义，这样大大提高了程序原型的验证效率，并缩短了实现时间。

同时也需要注意，这种自动数据类型转换会导致程序的运行并没有完全在编程设计之中，因为很多数据类型是自动进行了转换，那么程序就容易出现一些设计中未预料到的情况，这为程序稳定性带来极大的隐患。

**4. 显性数据类型转换**

从稳定性来看，更好的方式是通过编程的方式进行显性数据类型转换。在程序框图空白处右击，打开“函数”选板，选择“编程”选板→“数值”选板→“转换”选板，在该选板中选择相应的数据类型转换工具，如图 3.25 所示。

**5. 将整型显性转换为浮点数的实例**

接下来通过具体的实例讲解如何进行数据类型的转换，具体操作步骤如下。

(1) 在“数据类型-数值”VI 的程序框图中，右击“计算常量”接线端与“加”函数节点的连线，在弹出的菜单中选择“插入”→“数值”选板→“转换”选板，选择“转换为双精度浮点数”函数节点。

(2) 如图 3.26 所示，在插入“转换为双精度浮点数”函数节点后，红点消失了，证明 LabVIEW 编译器没有进行自动数据类型转换，因为此时输入“加”函数节点的都是双精度浮点型的数值。

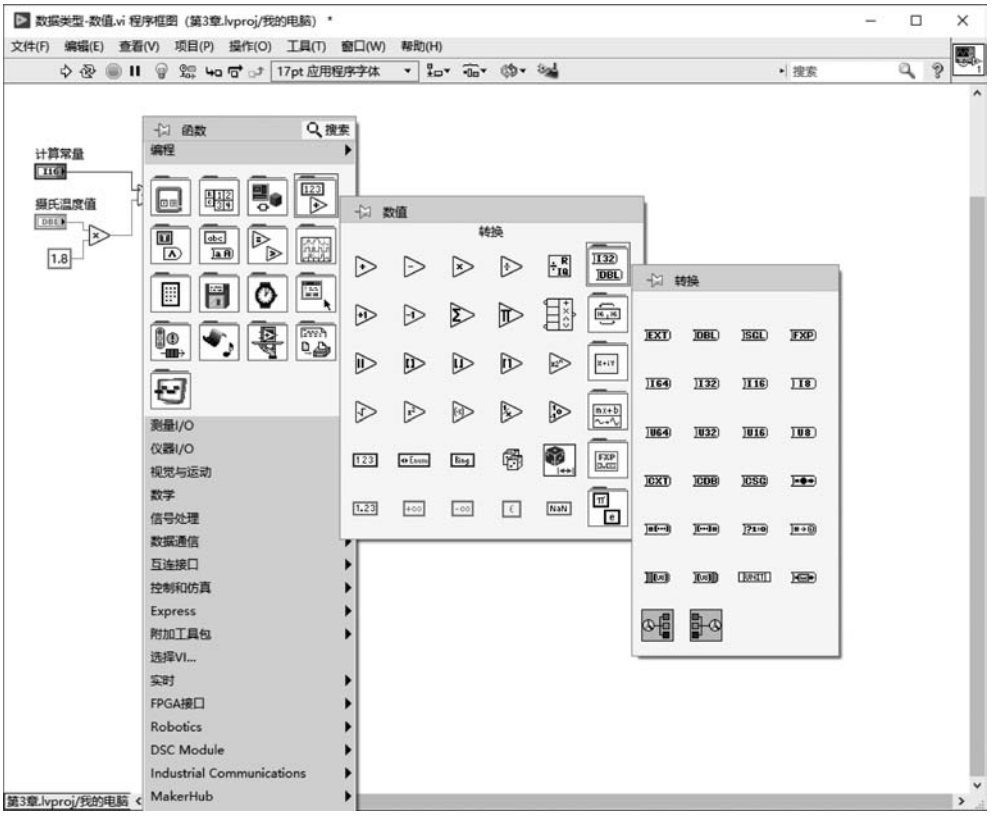


图 3.25　显性数据类型转换工具

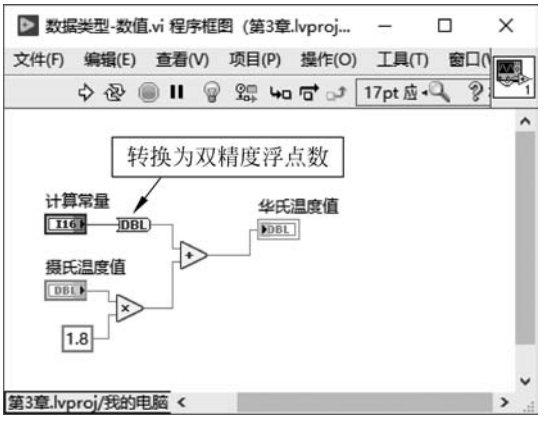


图 3.26　将整型显性转换为双精度浮点数

## 3.2 布尔


微课视频

### 3.2.1 布尔数据类型的概念

布尔是只有真、假两种值的数据类型。在编程中一般会使用布尔值表示只有两种状态的数据，如运算是否成功、程序是否完成等。

### 3.2.2 在前面板放置布尔控件

在前面板空白处右击，在弹出的菜单中选择“控件”选板→“新式”选板→“布尔”选板，可以添加布尔类型的输入和显示控件，如图 3.27 所示。

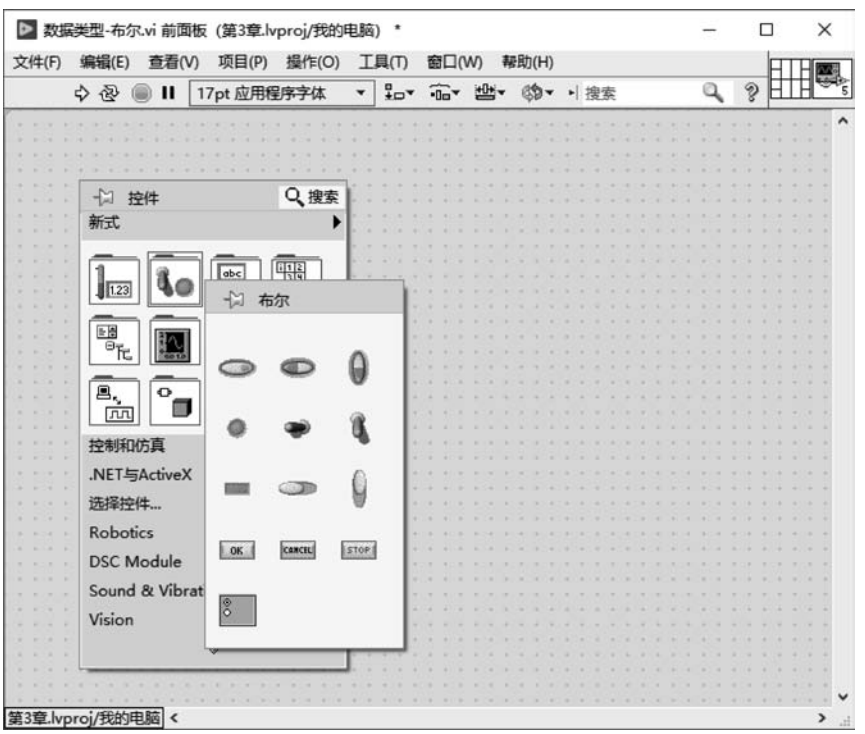


图 3.27 前面板“布尔”选板

LabVIEW 中的布尔数据类型控件有特别的属性，称为机械动作。这个概念来源于实际应用场景。实际应用中布尔数据类型的机械动作会有不同的区别，这里以开关和鼠标按键的机械动作为例进行说明。

首先来看表示开关的布尔状态，如下所示。

(1) 当按下开关到开的状态后，会保持在当前开的状态。

(2) 直到再次按下，开关会变成关的状态并保持。

如果用布尔数值类型描述这个开关的过程，当按下开关后，开关的状态从假值变为真值，并且保持在真值状态；如果再次按下开关，开关的状态从真值变为假值，并且保持在假值状态。

在 LabVIEW 中，用来表示这种开关的布尔数据类型的机械动作称为“单击时转换”，如图 3.28 所示。

接下来再看表示鼠标按键的布尔状态：当按下鼠标按键时，鼠标会先处于被按下的状态，但是并不会保持被按下的状态，而是快速恢复到没有按下的状态。

如果用布尔数值类型描述鼠标按键的过程，当按下鼠标按键后，鼠标按键的状态从假值变为真值，在没有再次按键之前，会迅速从真值变为假值。

在 LabVIEW 中，用来表示这种鼠标按键的布尔数据类型的机械动作称为“保持转换直至释放”，如图 3.29 所示。

图 3.28　单击时转换的机械动作

图 3.29　保持转换直至释放的机械动作

在 LabVIEW 中，通过布尔控件的机械动作描述这些实际布尔状态。

在前面板中右击布尔控件，在弹出的菜单中选择“机械动作”，在“机械动作”选板中可以选择机械动作类型。LabVIEW 的布尔数据类型有 6 种不同的机械动作，分别为单击时转换、释放时转换、保持转换直到释放、单击时触发、释放时触发、保持触发直到释放。

在实际应用中，根据程序设计的需要选择布尔控件的机械动作。例如，在上面的实例中，鼠标按键的机械动作是保持转换直至释放，而实际上在设计程序中，可以捕捉鼠标键的按下、保持、释放等多种状态进行人机界面的程序设计，根据实际的需要将鼠标键的布尔动作定义为不同的机械动作。

### 3.2.3　布尔数据的运算

可以通过“布尔”选板中的函数节点进行布尔数据运算。在程序框图空白处右击，打开“函数”选板，选择“编程”选板→“布尔”选板，该选板提供了布尔数据运算的函数节点，如图 3.30 所示。

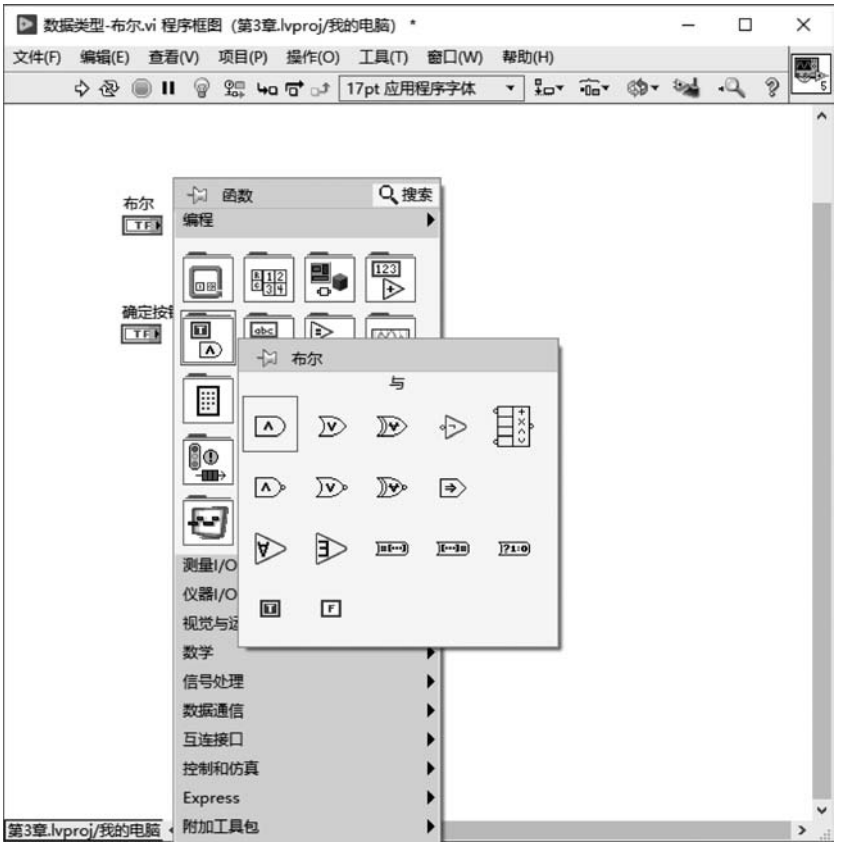


图 3.30　程序框图“布尔”选板


微课视频

### 3.2.4　密码锁实现实例

接下来使用布尔值设计实现密码锁功能的程序。程序的需求是密码锁有 3 个按钮(A,B,C),密码锁内置了一套密码规则,分别对应按钮 A,B,C 的状态,当输入的 A,B,C 按钮状态与内置的规则一致,密码锁就被打开,并且显示灯会点亮。在实例中设定按钮 A 为假值,按钮 B 和 C 同时为真值时,密码锁会打开并且显示灯亮。

密码锁实现的步骤如下。

(1) 在菜单栏中执行“文件”→“新建 VI”命令,创建新的 VI 文件。

(2) 在前面板菜单栏中执行“文件”→“保存”命令,将 VI 命名为“数据类型-布尔”。

(3) 在“数据类型-布尔”VI 前面板空白处右击,选择“控件”选板→“新式”选板→“布尔”选板,选择添加“开关按钮”控件,右击“开关按钮”控件,在弹出的菜单中选择“机械动作”,并选择“单击时触发”。

(4) 双击“开关按钮”标签,输入 A 作为这个按钮的标签名。

(5) 用同样的方法创建标签为 B 和 C 的按钮。

（6）在“数据类型-布尔”VI的前面板空白处右击，选择“控件”选板→“新式”选板→“布尔”选板，选择添加“圆形指示灯”控件，双击“圆形指示灯”标签，输入OK作为标签名，如图3.31所示。

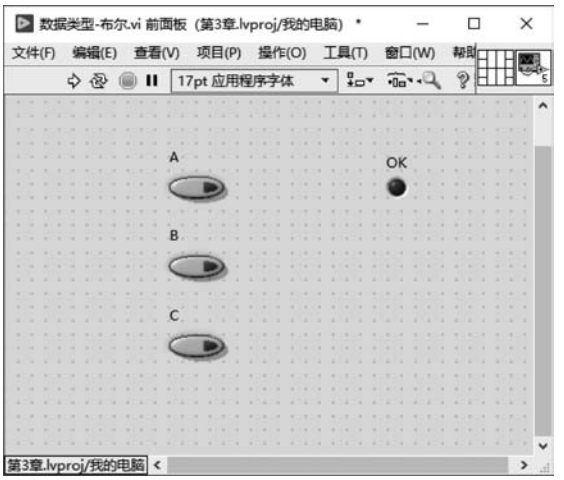


图3.31　在前面板中添加布尔输入和显示控件

（7）接下来在程序框图中编程实现一套内置的密码规则，实现当接线端A,B,C分别为假、真、真值时输出真值。在程序框图中接线端A,B,C后分别放置假、真、真值的布尔常量，如图3.32所示。

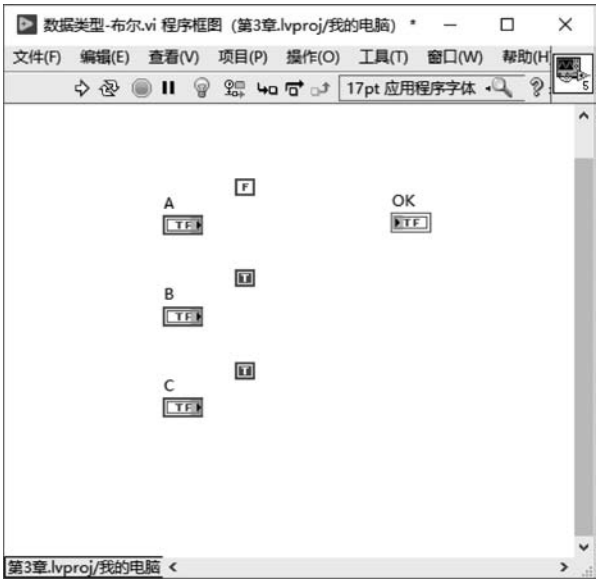


图3.32　在程序框图中放置布尔常量

(8) 接下来通过布尔运算实现密码锁开锁的判断。在程序框图中将接线端 A,B,C 的值与布尔常量进行比较运算,当接线端 A,B,C 与内置的布尔常量都相等的时候,在 OK 接线端输出真值。

在程序框图空白处右击,打开“函数”选板,选择“编程”选板→“比较”选板,选择“等于?”函数节点放置在接线端 A 和布尔常量 F(假值) 之后,将接线端 A 和布尔常量 F 连接到“等于?”函数节点。

用同样的方法将接线端 B 与布尔常量 T(真值)连接到函数节点“等于?”;将接线端 C 与布尔常量 T(真值)连接到函数节点“等于?”,如图 3.33 所示。

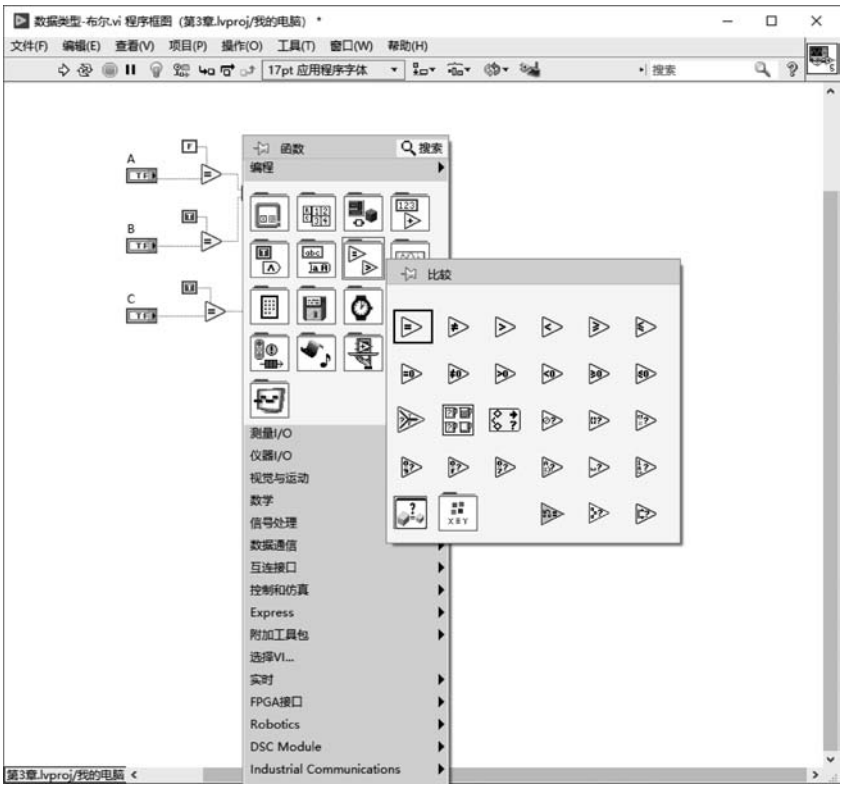


图 3.33 添加“等于?”函数节点

打开密码锁的条件是接线端 A,B,C 输入的值都正确的时候,才输出真值,所以将接线端 A,B,C 与内置的布尔常量值的对比得到的 3 个结果再进行“与”运算。

在程序框图空白处右击,打开“函数”选板,选择“编程”选板→“布尔”选板,再选择“与”函数节点。通过两次“与”运算得到的结果输出到 OK 接线端,如图 3.34 所示。

(9) 下面验证密码锁的程序,首先尝试输入正确密码的情况。在“数据类型-布尔”VI 的前面板中,通过单击将输入控件 A,B,C 选择为假、真、真,单击“运行”按钮。程序运行结束后,可以看到前面板中的 OK 控件高亮显示,当前 OK 显示控件的值为真。

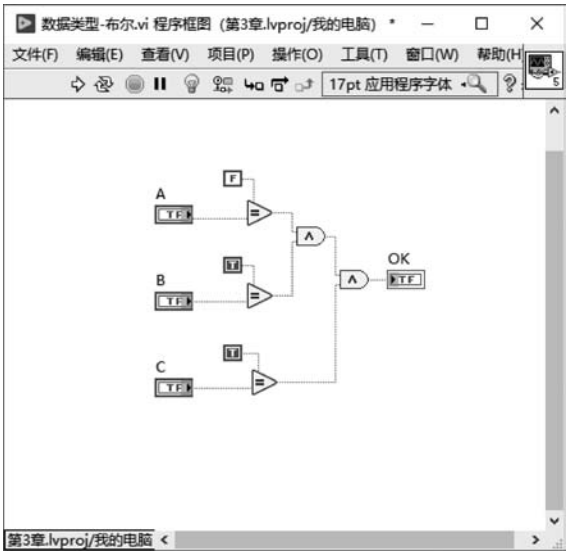


图 3.34　添加“与”函数节点

因为输入布尔输入控件的值与内置的布尔常量的值相等，所以得到的结果为真，代表密码锁被打开，如图 3.35 所示。

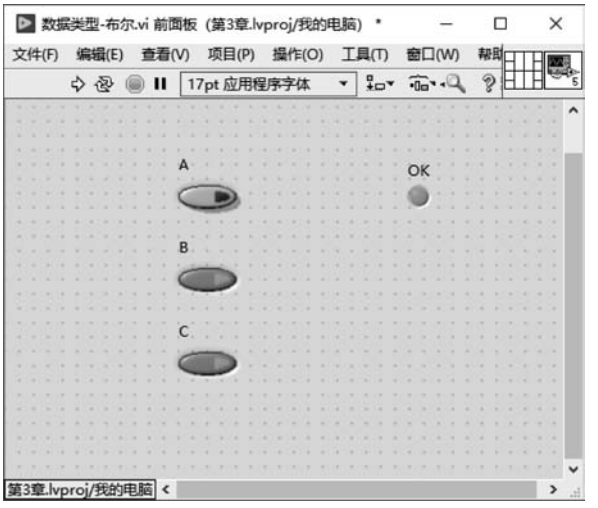


图 3.35　输入正确密码时的程序运行结果

(10) 下面继续尝试错误密码输入的情况。在“数据类型-布尔”VI 的前面板中，通过单击将输入控件 A，B，C 选择为假、假、真，单击“运行”按钮。程序运行结束后，可以看到前面板中的 OK 控件没有高亮显示，当前 OK 显示控件的值为假。

因为输入布尔控件的值与内置的布尔常量的值不一致，所以密码没有匹配，密码锁没有

被打开，如图 3.36 所示。

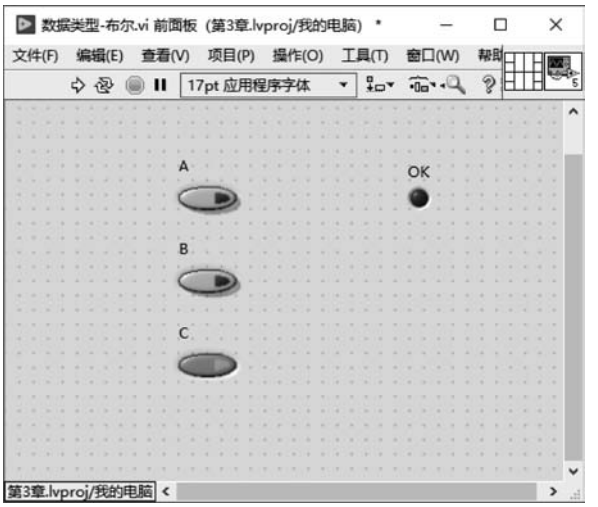


图 3.36　输入错误密码时的程序运行结果

## 3.3　字符串


微课视频

### 3.3.1　字符串的概念

字符串一般用于程序设计中对文本字符的显示和处理，在计算机中所有的文本都是字符串的形式。文本字符串在计算机中是以二进制数值保存的，在显示的时候遵循美国信息交换标准代码(American Standard Code for Information Interchange，ASCII)格式。

ASCII 码是基于拉丁字母的一套计算机编码系统，主要用于显示现代英语和其他西欧语言。它是最通用的信息交换标准，并等同于国际标准 ISO/IEC 646。ASCII 码第一次以规范标准的类型发表是在 1967 年，最后一次更新则是在 1986 年，到目前为止共定义了 128 个字符。一些典型的 ASCII 码与字符串对应如表 3.2 所示。

**表 3.2　字符串的 ASCII 码对应表**

| Bin(二进制) | Oct(八进制) | Dec(十进制) | Hex(十六进制) | 缩写/字符 | 解释 |
|---|---|---|---|---|---|
| 0100 0001 | 101 | 65 | 0x41 | A | 大写字母 A |
| 0100 0010 | 102 | 66 | 0x42 | B | 大写字母 B |
| 0100 0011 | 103 | 67 | 0x43 | C | 大写字母 C |
| 0100 0100 | 104 | 68 | 0x44 | D | 大写字母 D |
| 0100 0101 | 105 | 69 | 0x45 | E | 大写字母 E |
| 0100 0110 | 106 | 70 | 0x46 | F | 大写字母 F |

## 3.3.2　字符串在硬件通信中的应用

在 LabVIEW 编程中，字符串一般的使用场合是与第三方软件平台的数据交互，如第三方硬件和软件的数据交互。

在程序设计中与第三方硬件是通过各种总线进行通信的，总线的形式包含串口、I2C、串行外设接口（Serial Peripheral Interface，SPI）、以太网口等，这些总线上的数据通信都是将数值和命令等信息转化为一定格式的字符串进行传输。例如，在程序设计中与一台示波器进行控制和读取通信的时候，如果是通过串口总线进行控制，第一步会发送内容为 * IDN 的字符串命令查询当前仪器的名称，示波器也会在串口总线上通过字符串的方式返回设备仪器的名称。

**1. 使用字符串与设备进行通信的实例**

在 NI 范例查找器中查找针对 Agilent 34401 万用表仪器的 Agilent 34401 Read Math Measurement 范例程序。通过这个程序了解在设备通信中字符串的具体使用情况。

（1）执行“帮助”→“查找范例”命令，打开 NI 范例查找器。

（2）在打开的“NI 范例查找器”窗口中，在左侧的菜单栏中选择“搜索”标签页，在“输入关键词”文本框内输入 Agilent，可以看到在中间的窗口中罗列了与关键词 Agilent 相关的范例程序，如图 3.37 所示。

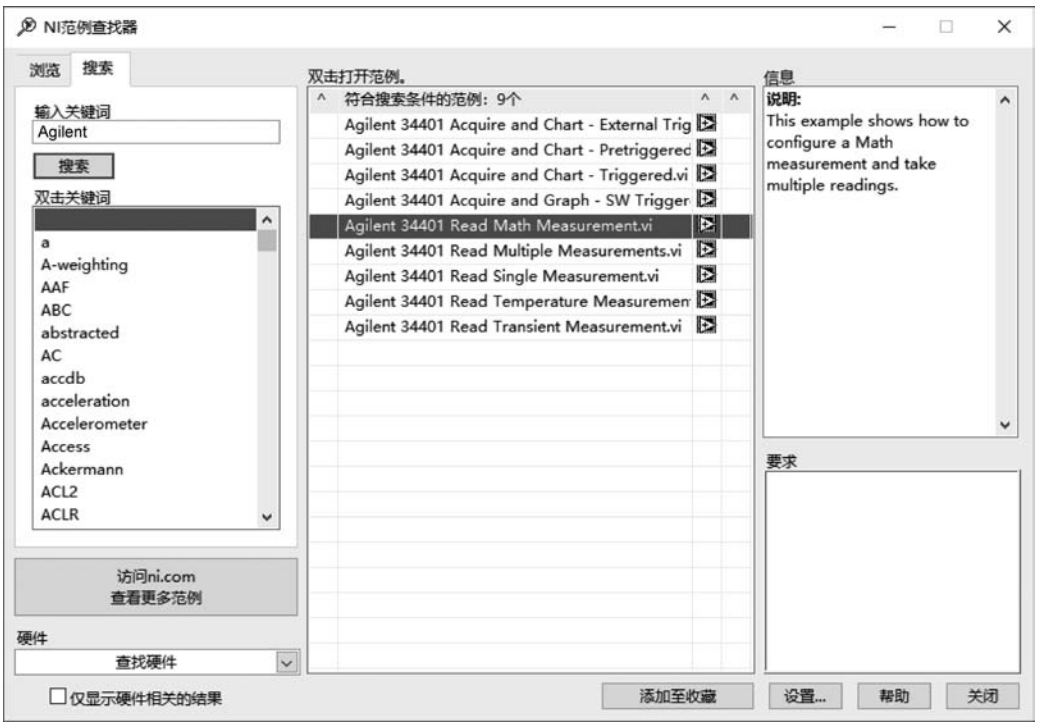


图 3.37　搜索关键词 Agilent

(3) 双击 Agilent 34401 Read Math Measurement 范例,打开这个 VI 文件。在这个程序中使用了串口通信的方式控制设备 Agilent 34401,接下来通过函数节点定位到具体的通信代码位置。

在程序框图中双击 Agilent 34401.lvlib:Read(Multiple Points)节点,如图 3.38 所示,打开它的前面板和程序框图。

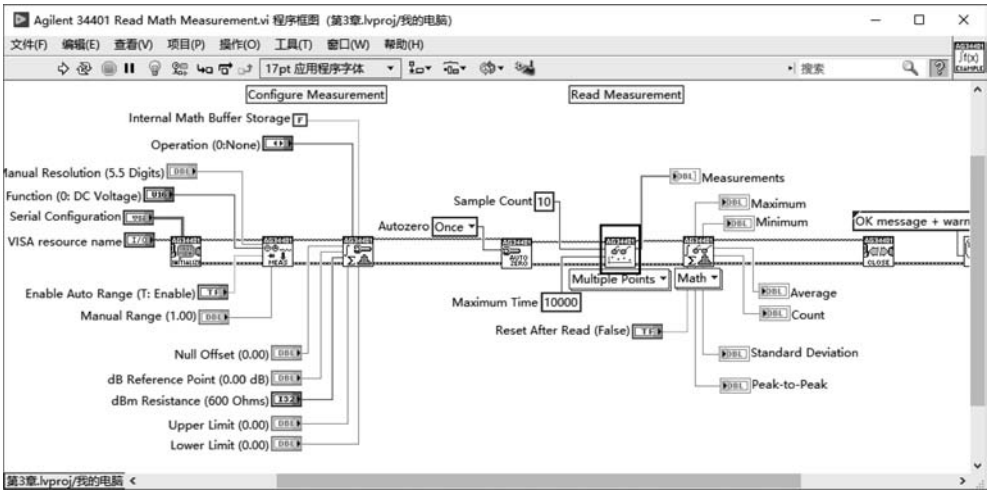


图 3.38 Agilent 34401 Read Math Measurement 范例程序

(4) 在 Agilent 34401.lvlib:Read(Multiple Points)的程序框图中,双击 Agilent 34401.lvlib:Fetch Measurement 节点,如图 3.39 所示,打开它的前面板和程序框图。

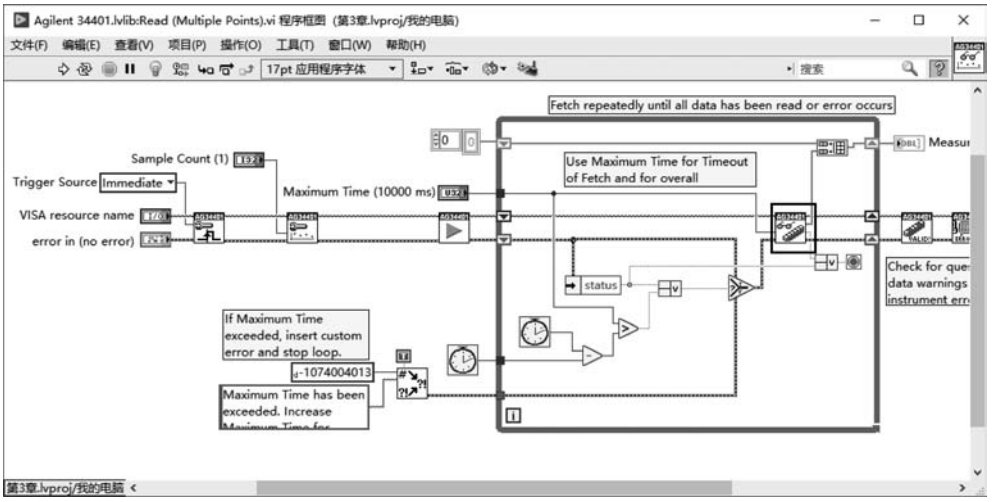


图 3.39 Read(Multiple Points)节点的程序框图

（5）在 Agilent 34401. lvlib：Fetch Measurement 节点的程序框图中，双击 Agilent 34401. lvlib：Fetch Measurement（Fetch）节点，如图 3. 40 所示，打开它的前面板和程序框图。

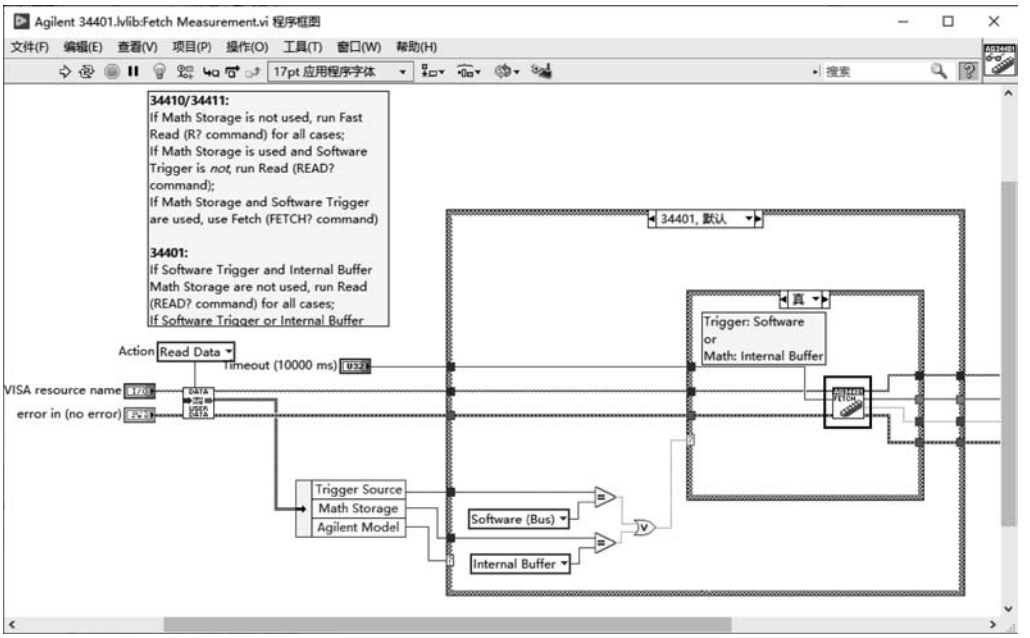


图 3. 40　Fetch Measurement 节点的程序框图

（6）在当前的程序框图中，可以看到具体通过字符串通信的代码。Agilent 34401. lvlib：Fetch Measurement（Fetch）程序框图中的第一个函数节点是“VISA 写入”，通过该函数节点向仪器设备发送了字符串“：FETC?”，这段命令的作用是让仪器返回当前读取的数据。

VISA 系列函数是 LabVIEW 提供的用于总线通信的驱动。

程序框图中的第二个函数节点是“VISA 读取”，这个函数从万用表仪器中读取了返回的数据，数据是字符串的格式。

接下来范例程序通过一系列的函数节点将字符串转化为浮点型数据，如图 3. 41 所示。

**2. 使用字符串与第三方程序通信的实例**

在 LabVIEW 编程中，与第三方程序进行通信一般是通过 TCP/IP，这也是基于字符串进行通信。一般需要将数据打包成字符串通过 TCP/IP 发送，同时将接收到的字符串再解析成数据。接下来，通过一个 TCP/IP 通信的实例讲解具体字符串的应用，操作步骤如下。

（1）打开 NI 范例查找器，在左侧“搜索”标签页中的“输入关键词”文本框中输入 TCP，双击中间窗口中的 Simple TCP. lvproj 范例程序，如图 3. 42 所示。

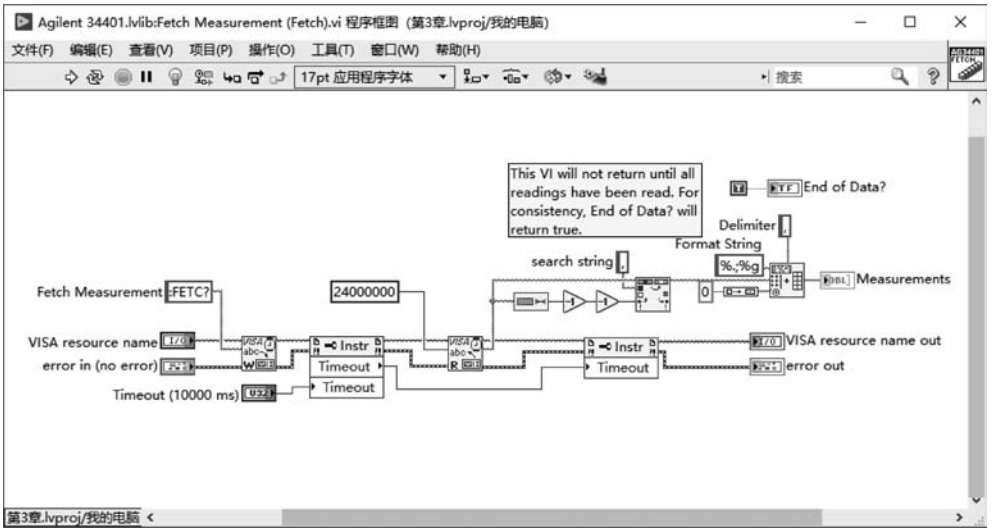


图 3.41　Fetch Measurement(Fetch)程序框图

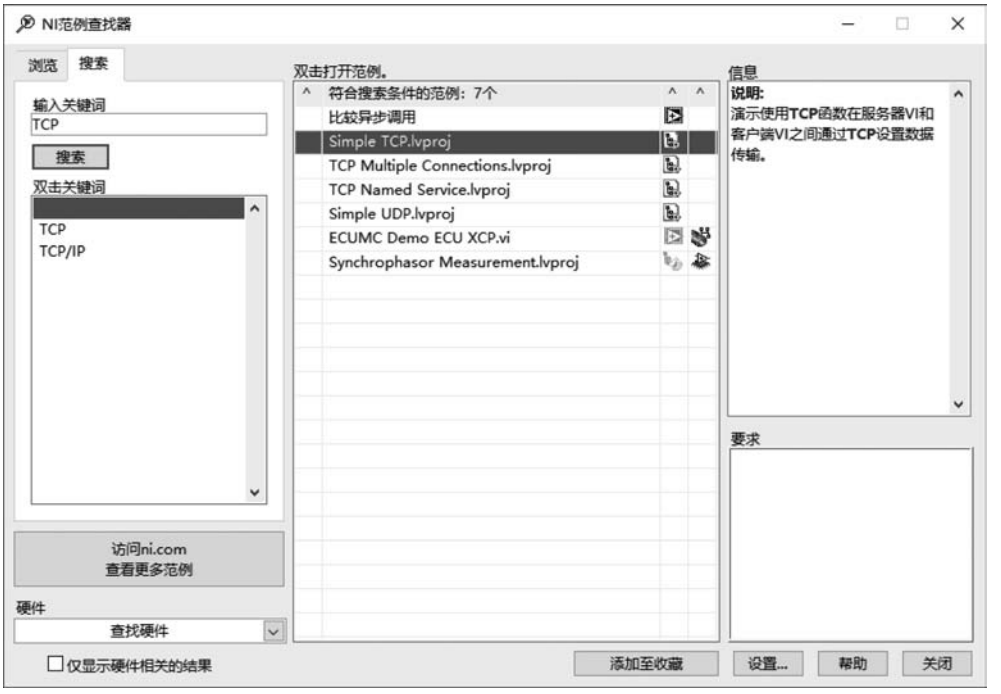


图 3.42　打开 SimpleTCP.lvproj 范例程序

（2）在 SimpleTCP.lvproj 范例程序中打开 Simple TCP-Client VI 文件，在程序框图中可以看到在“读取 TCP 数据”节点处输出的是通过 TCP/IP 接收到的字符串，之后又通过其他节点将字符串转换为数值，如图 3.43 所示。

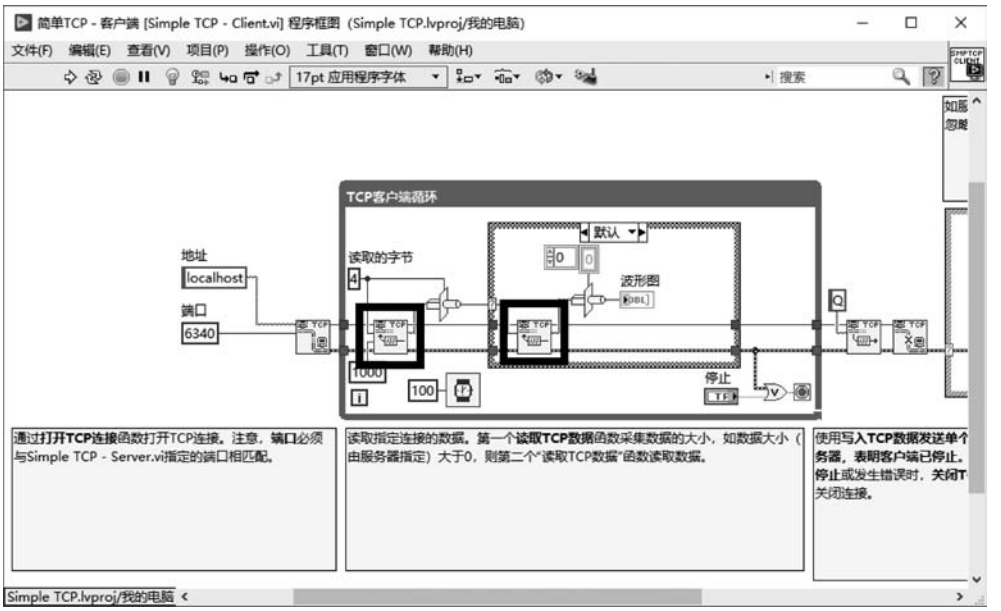


图 3.43　SimpleTCP-Client 程序框图

### 3. 单纯字符串的处理

LabVIEW 程序框图中的“字符串”选板提供了处理字符串的函数节点。

实际上，在 LabVIEW 中处理单纯字符串的情况比较少，主要有以下两个原因。

（1）LabVIEW 中提供了便捷的数据显示，取代了需要通过字符串将数值输出到前端的过程。

例如，程序运行过程中需要对一些变量的值进行检测。在文本编程语言中需要将这些过程变量转换为字符串输出到前面板，而 LabVIEW 可以直接提供数值的显示控件用来显示过程中的数据和最终的结果数据。所以在很多文本编程语言的 Debug 过程中通过字符串的数值显示就并不需要。

这也是我们在开始并没有进行 Hello World 字符串输出实验的原因，因为这个实验在单机版的 LabVIEW 的程序中并没有实际的意义。

（2）在 LabVIEW 中很少处理单纯基于文字的算法应用，如处理包含大量人名、属性信息的数据库。尽管 LabVIEW 提供了针对 Office、SQL 数据库的工具包，但是 LabVIEW 超过 80% 的应用是基于数值数据的，所以应该使用 LabVIEW 处理这样类型的项目应用。


微课视频

### 3.3.3 字符串运算操作实例

程序框图中的“字符串”选板提供了字符串操作的函数节点。在程序框图空白处右击，打开“函数”选板，选择“编程”选板→“字符串”选板，可以看到字符串运算的节点和子选板，如图 3.44 所示。


图 3.44 “字符串”选板

字符串的运算操作中，很多情况下是从字符串文本中判断和截取某个特定的字符串，下面通过一个实例讲解字符串的运算操作。

本实例实现的功能是模拟程序与第三方硬件设备通信。在通信中使用串口总线的通信协议，串口总线通信过程中使用的数据是字符串形式，需要通过字符串运算将具体的数值数据解析出来。

解析的第一个步骤是截取出表示数据的字符串，具体实现如下。

(1) 在 LabVIEW 菜单栏中执行“文件”→“新建 VI”命令，创建一个空白 VI 文件。

(2) 在 LabVIEW 菜单栏中执行“文件”→“保存”命令，将文件命名为“数据类型-字符串”。

(3) 在“数据类型-字符串”VI 的前面板空白处右击，在弹出的菜单中选择“控件”选板→“新式”选板→“字符串与路径”选板，选择“字符串”输入控件放置在前面板中。

(4) 在“字符串”输入控件中输入文本内容,模拟从硬件读取回来的数据。单击“字符串”输入控件的文本框,会进入输入模式,输入以下文本。

Initialize……

Transfering data:

Voltage: 32

Voltage: 34

Voltage: 36

(5) “字符串”输入控件中的字符串数据中包含了硬件的状态和返回的数据,数据是字符串类型,无法直接进行数值的计算,接下来通过字符串的处理解析出数值类型的数据。首先将通过字符串的操作定位到字符串中数据的位置。

(6) 在程序框图空白处右击,打开“函数”选板,选择“编程”选板→“字符串”选板,如图 3.45 所示,选择“截取字符串”函数节点,放置在程序框图中。

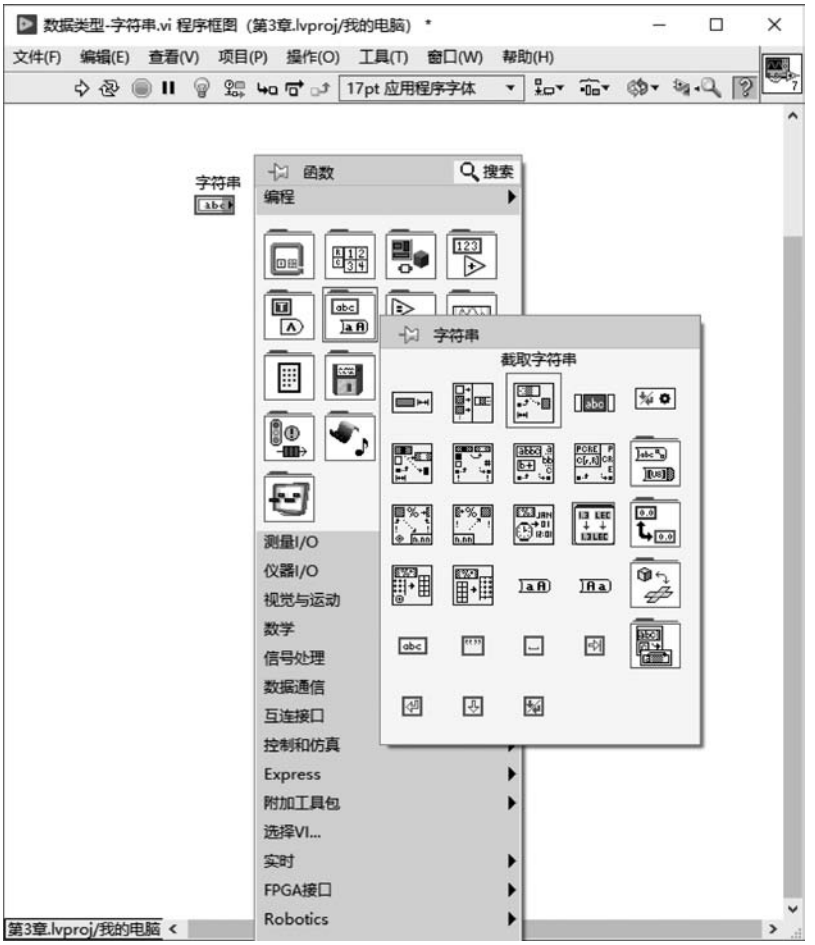


图 3.45　在程序框图中添加“截取字符串”函数节点

(7) 在程序框图中,右击“截取字符串”函数节点左侧的“字符串”接线端,在弹出的菜单中选择“创建”→“输入控件”。可以看到,“截取字符串”函数节点创建了一个字符串数据类型的接线端。

同样,依次为“截取字符串”函数节点创建“偏移量(0)”“长度(剩余)”接线端。在“截取字符串”函数节点右侧创建“子字符串”接线端,如图 3.46 所示。

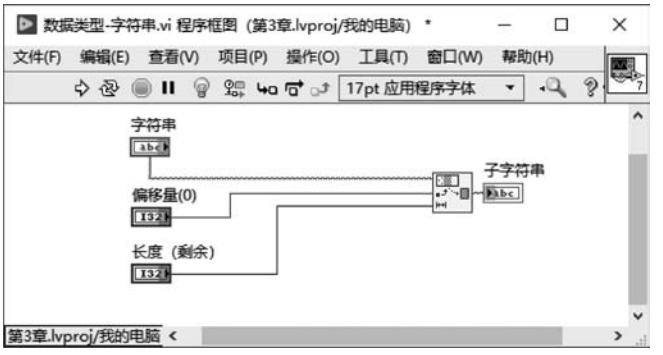


图 3.46 在程序框图中为“截取字符串”函数节点添加接线端

(8) 为“截取字符串”函数节点添加接线端后,在前面板中会自动添加与接线端对应的输入和显示控件。分别在“偏移量(0)”和“长度(剩余)”输入控件的文本框中输入数值 42 和 2。这样,会从“字符串”输入控件的字符串文本中从起始位置偏移 42 个字符后,读取两个字符,也就是“32”这两位字符串。

在前面板菜单栏中单击“运行”按钮,可以看到在“子字符串”显示控件的文本框中显示字符串“32”,如图 3.47 所示。

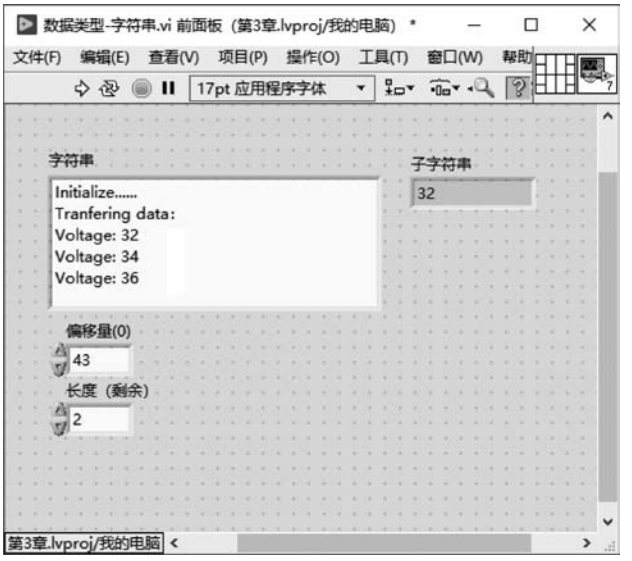


图 3.47 设置截取字符串的参数

这样，“数据类型-字符串”VI 就完成了一个基本字符串提取的功能。

### 3.3.4　字符串的转换

LabVIEW 对字符串的处理中，很多情况下需要将字符串转换为不同的数据格式。在“数据类型-字符串”这个实例中，通过截取字符串已经得到数值数据的字符串，接下来还需要进行字符串至数值的转换，这样才可以进行数值的计算。

“字符串”选板中的“数值/字符串转换”选板提供了用于字符串与数值相互转换的函数节点，如图 3.48 所示。

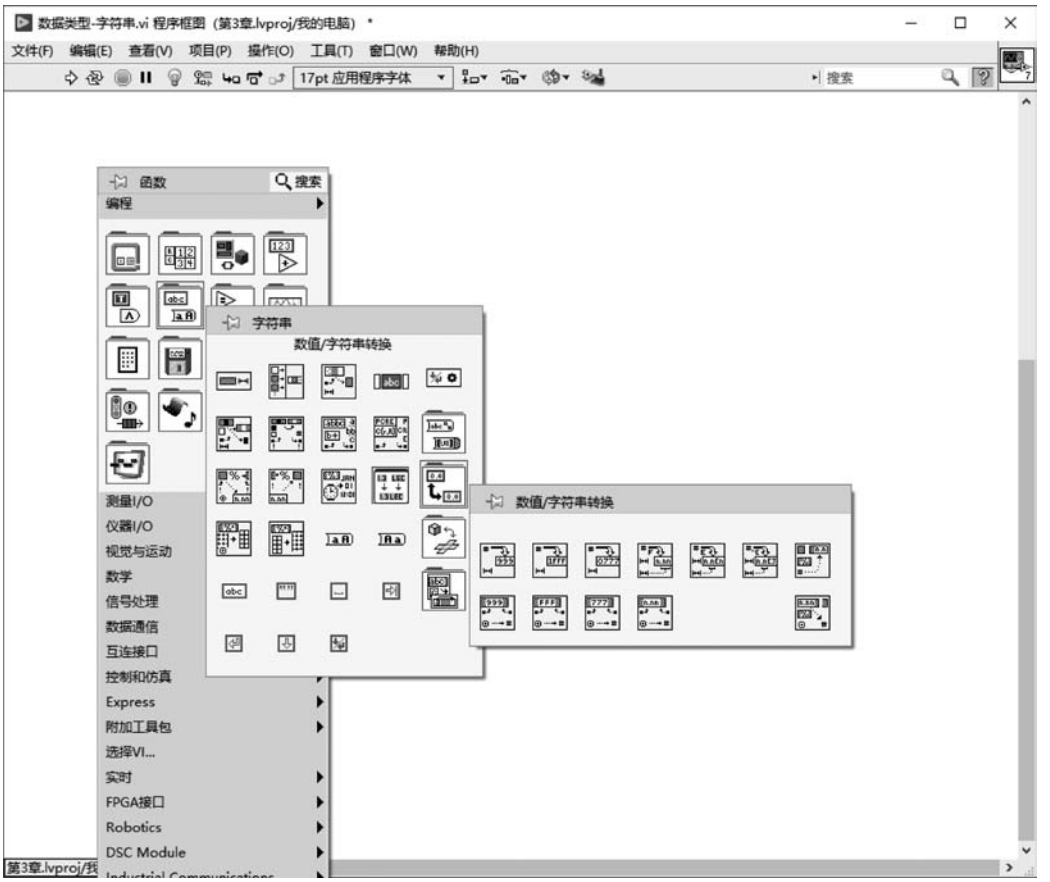


图 3.48　“数值/字符串转换”选板

接下来在“数据类型-字符串”实例程序中，将截取出的字符串数据转换为数值数据，具体操作步骤如下。

（1）在“数据类型-字符串”VI 的程序框图空白处右击，打开“函数”选板，选择“编程”选板→“字符串”选板→“数值/字符串转换”选板，选择“十进制数值字符串至数值转换”函数节点，放置在“截取字符串”函数节点的后面，如图 3.49 所示。

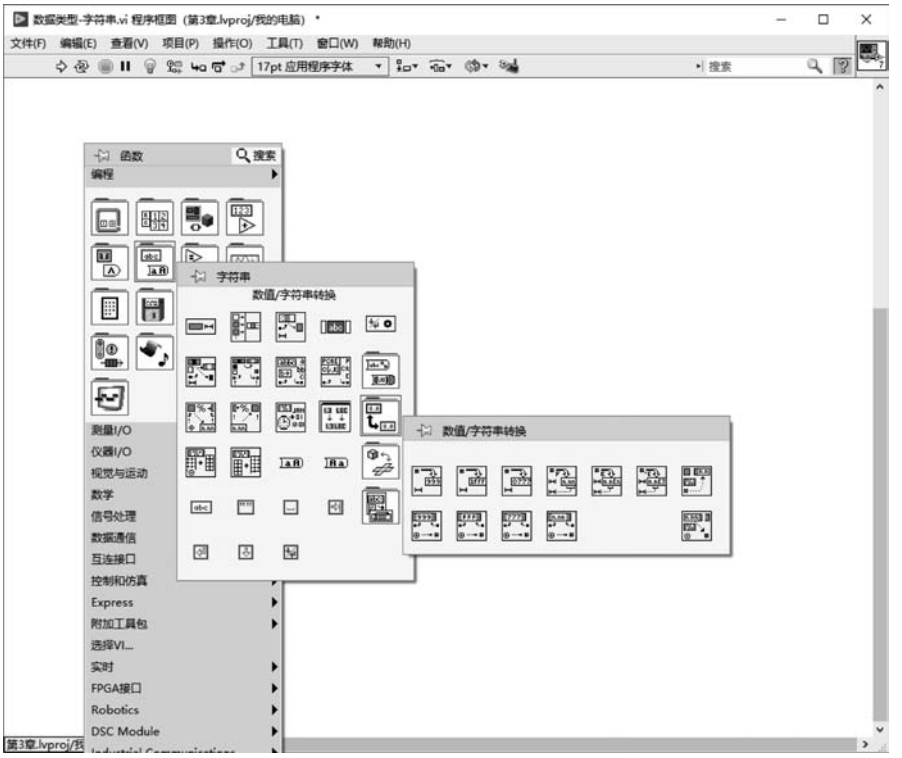


图 3.49　添加“数值/字符串转换”函数节点

（2）在程序框图中，将“截取字符串”函数节点的“子字符串”接线端输出连接至“十进制数值字符串至数值转换”的输入端，右击“十进制数值字符串至数值转换”函数节点右侧的“数字”接线端，在弹出的菜单中选择“创建”→“显示控件”，在“十进制数值字符串至数值转换”函数节点右侧会出现一个标签为“数字”的接线端，如图 3.50 所示。

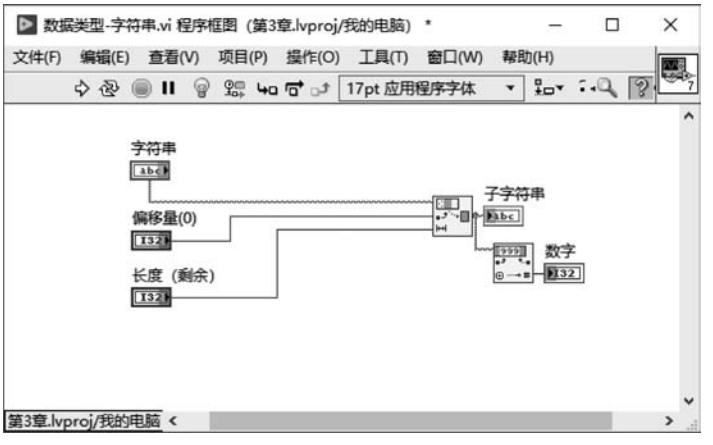


图 3.50　将截取的字符串转换为数值

（3）在前面板工具栏中单击“运行”按钮，从“数字”显示控件中可以看到，“子字符串”显示控件中的字符串数据类型的“32”已经被转换为数值数据类型的“32”，如图 3.51 所示。这样就完成了从串口总线得到字符串数据，并且解析出第一个数值数据。

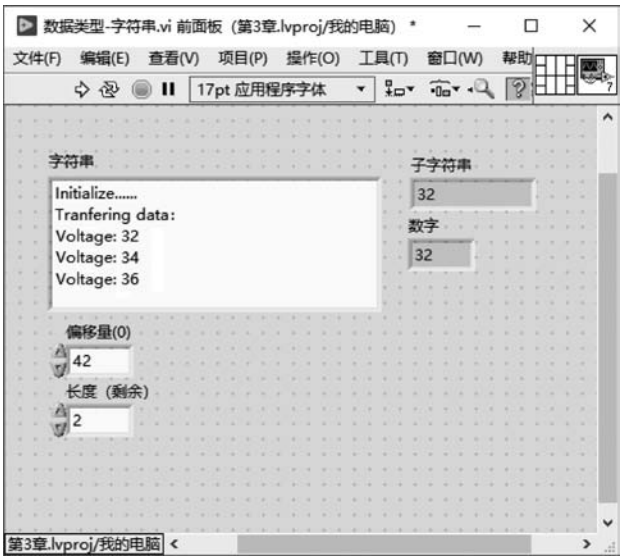


图 3.51　从字符串解析出第一个数值数据