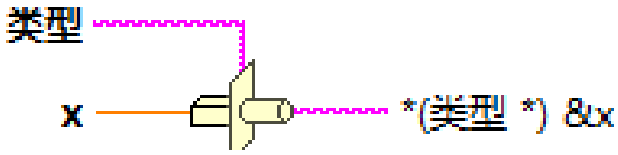
第3章-b LabVIEW的数据类型转换

几种常用的数据类型转换节点函数

✱ 3.1.1 强制类型转换函数

对存储在连续内存中的数据进行重新构造。



X和类型大小不一致的结果

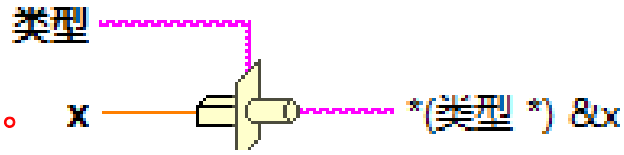
如**x**和类型的大小不一致，函数将生成非预期数据。 如**x**需要的存储位数大于**类型**可提供的位数，该函数将使用**x**的高位字节，丢弃剩余的低位字节。 如**x**是小于**类型**的数据类型，该函数将移动**x**中的数据至**类型**的高位字节，并在剩余的低位字节中填充0。 例如，8位不带符号整数1转换为16位不带符号整数时，结果是256。

数组的类型转换

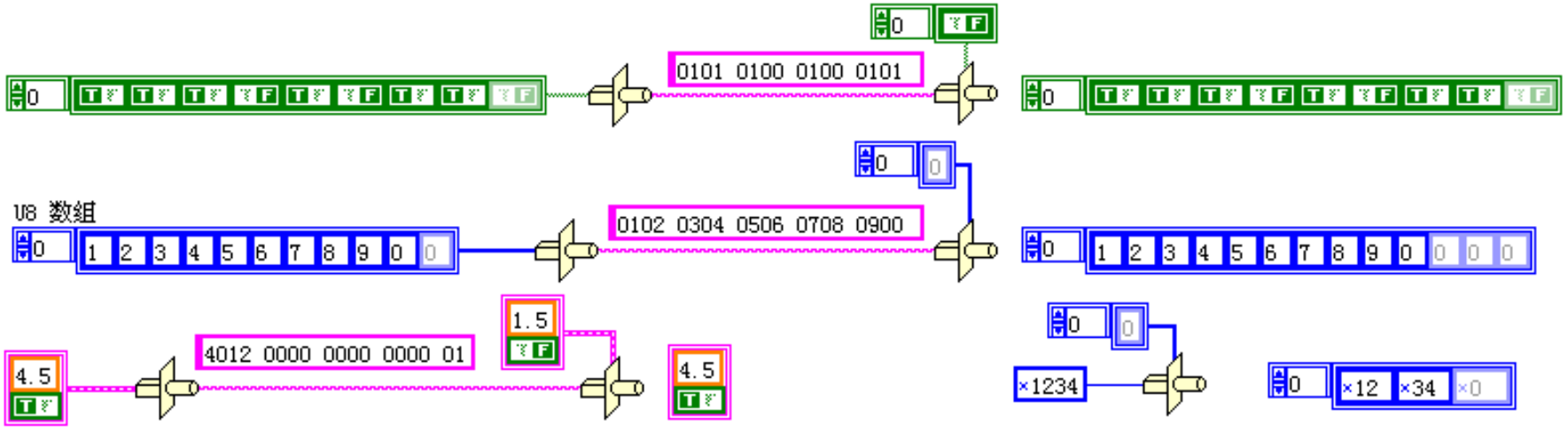
该函数可用于标量数组或标量簇数组。 例如，如将4个16位整数组成的数组转换为32位整数数组，输出数组包含两个元素，每个元素由输入数组的两个元素的字节连接后组成。 如输入数组部分元素包含的字节数不足以形成输出数组的一个元素，LabVIEW将忽略输入数组的最后几个元素。

几种常用的数据类型转换节点函数

✱ 3.1.1 强制类型转换函数



必须保证两个要转换的数据所占内存大小的一致性。



几种常用的数据类型转换节点函数

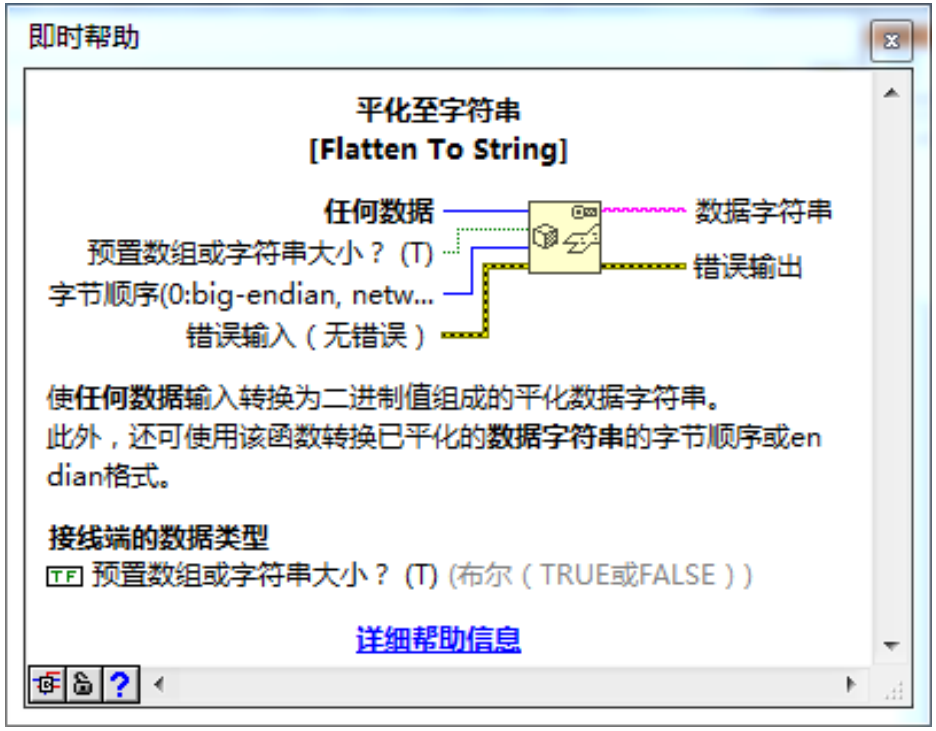
✱ 3.1.2 平化数据至字符串及字符串还原平化数据函数

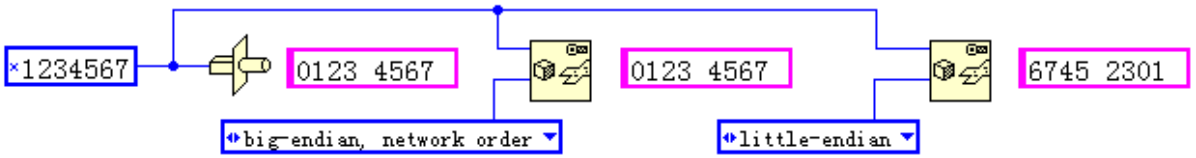
将数据从其内存格式，转换为一种更适合于进行文件读写的格式。

预置数组或字符串大小？（LabVIEW中数组本身可以保存数组的长度值）指定当**任何数据**为数组或字符串时，LabVIEW在**数据字符串**的开始是否包括数据大小信息。 如**预置数组或字符串大小？**的值为FALSE，LabVIEW将不包含大小信息。 默认值为TRUE。

字节顺序用于设置返回的平化字符串中数据的endian形式。 字节顺序，或endian形式，表明在内存中整数是否按照从最高有效字节到最低有效字节的形式表示，或者相反。

0	**big-endian, network order**（默认）—最高有效字节占据最低的内存地址。
1	**native, host order**—使用主机的字节顺序格式。
2	**little-endian**—最低有效字节占据最低的内存地址。

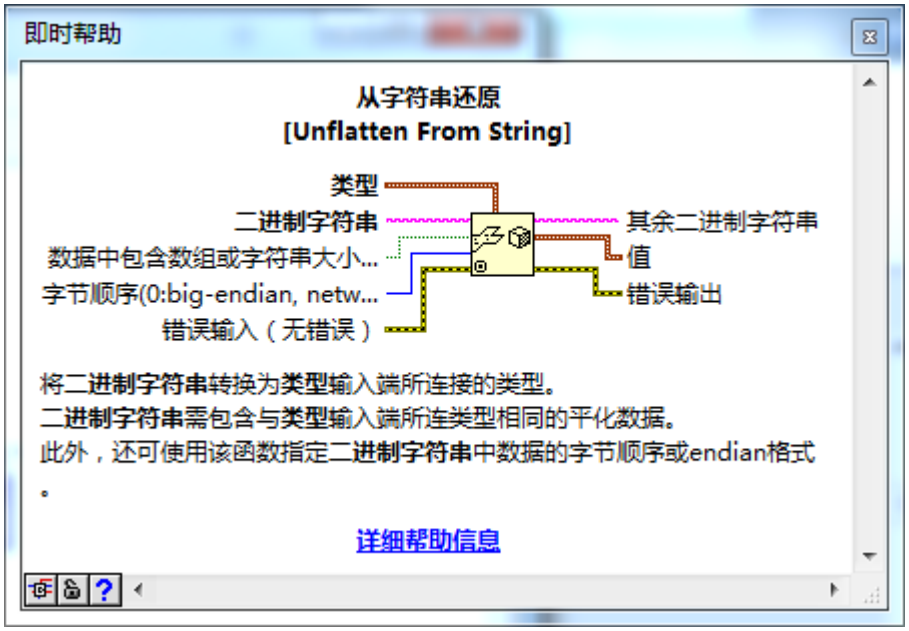


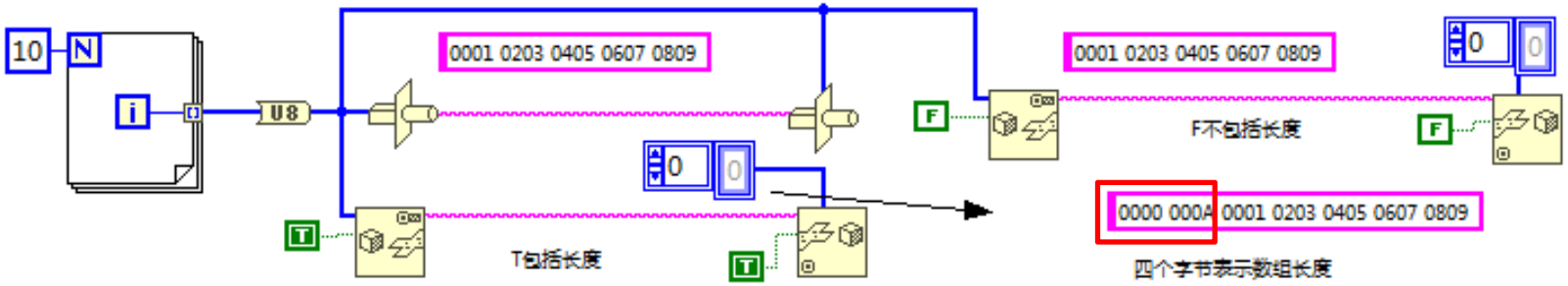


几种常用的数据类型转换节点函数

✱ 3.1.2 平化数据至字符串及字符串还原平化数据函数

把平化字符串转换成相应的数据类型。



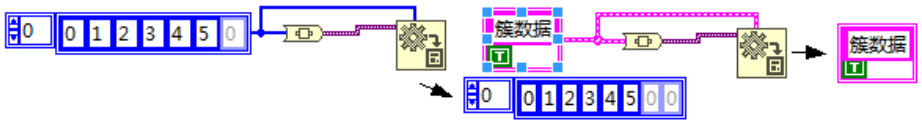


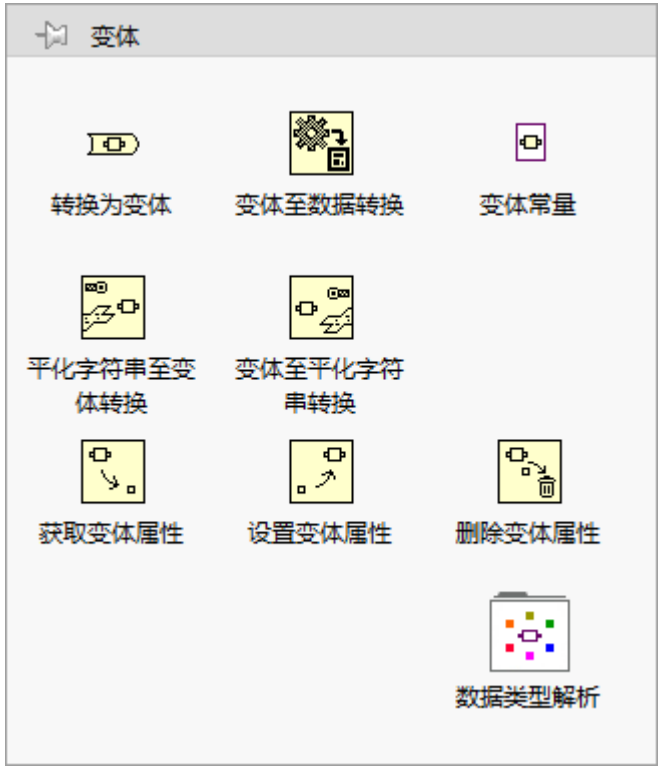
几种常用的数据类型转换节点函数

✱ 3.1.3 变体类型数据

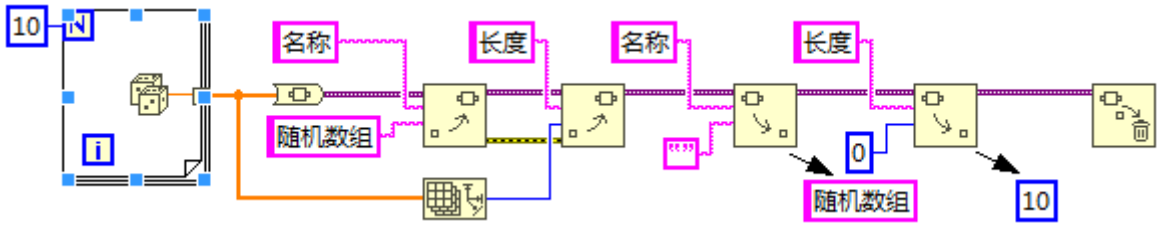
常规语言也支持变体数据，变体数据在自动化服务器、ActiveX编程和网络通信方面应用广泛。

使用变体数据最大的好处是，它兼容各种数据类型。





变体数据具备属性。

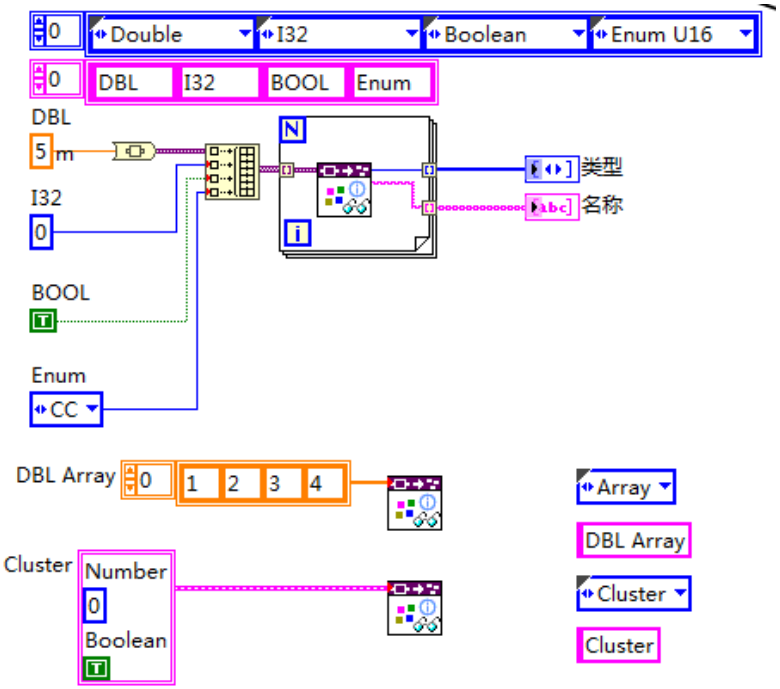


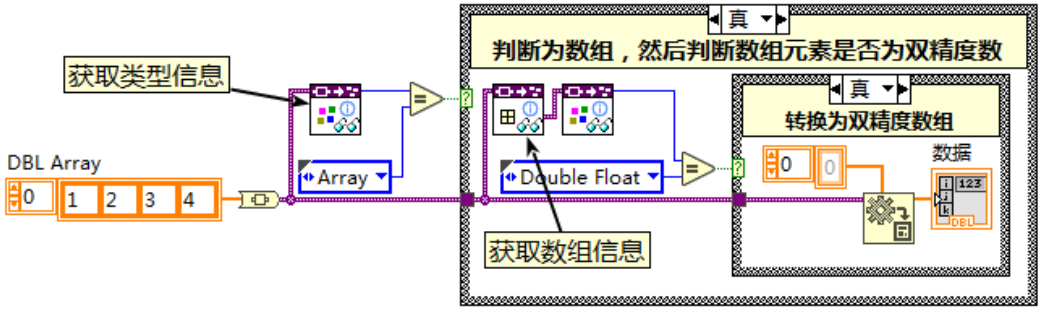
几种常用的数据类型转换节点函数

✱ 3.1.3 变体类型数据

变体数据可以存储任意数据类型，根据指定的数据类型，还可以还原成原始数据。

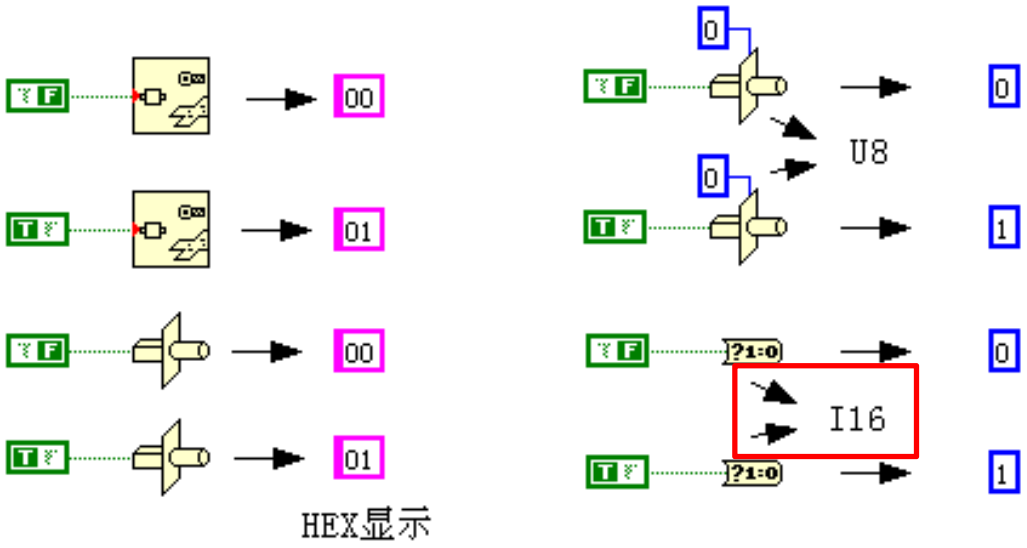
例如在队列状态机中，每次传输的变体数据其实际数据类型并不相同，这样在还原数据时，就无法指定相应的数据类型。





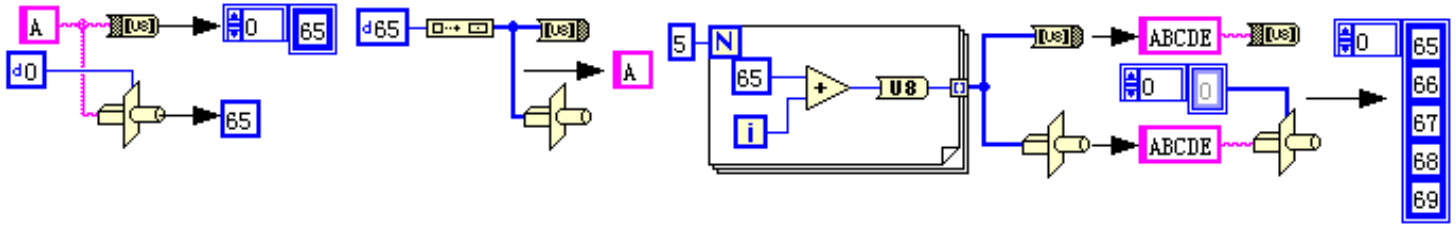
整数的类型转换及内存映射

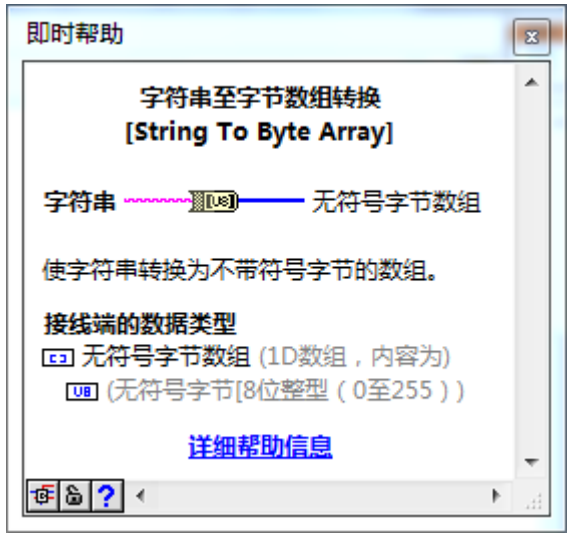
- **3.2.1** 布尔类型与字符串和数值的相互转换

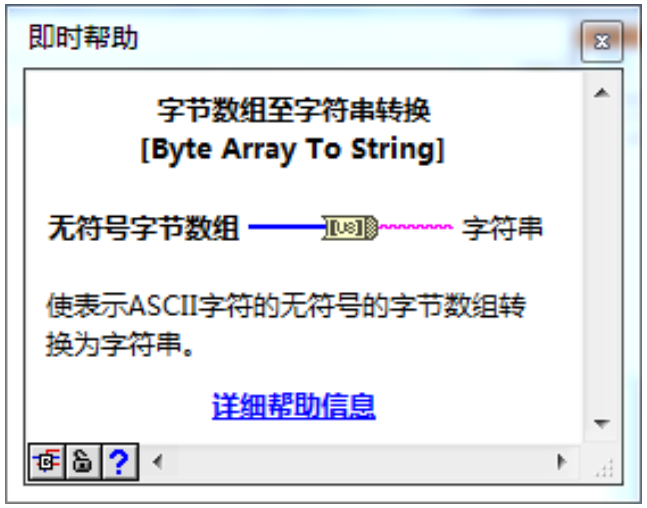


整数的类型转换及内存映射

- **3.2.2 U8类型与字符串**

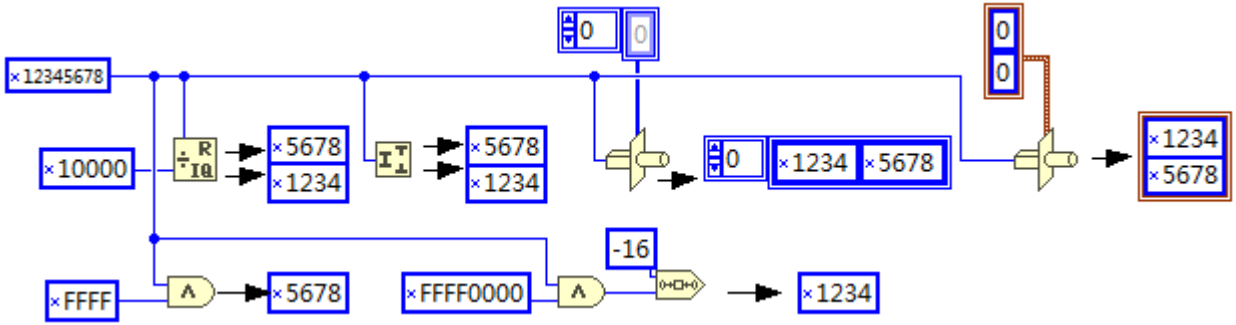


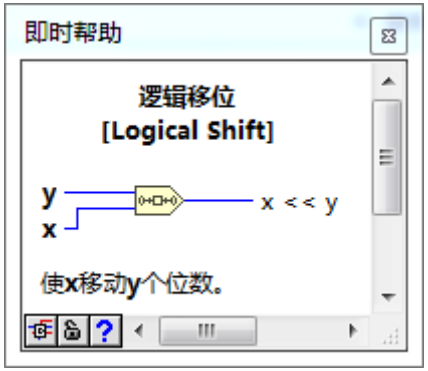


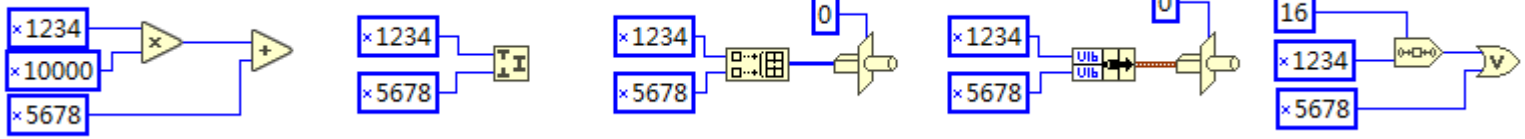


整数的类型转换及内存映射

- **3.2.3** 其它整数的相互转换



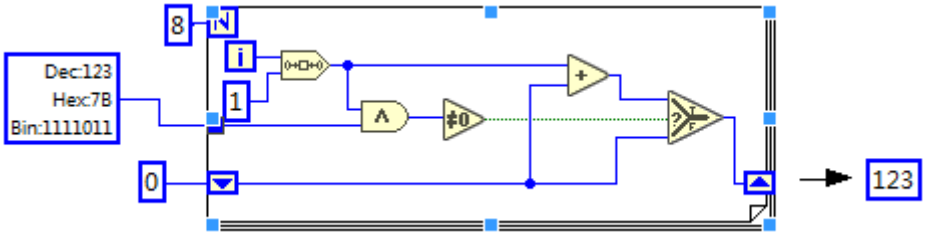




其它标量数据类型的类型转换及内存映射

✱ 3.3.1定点数和浮点数的类型转换和内存映射

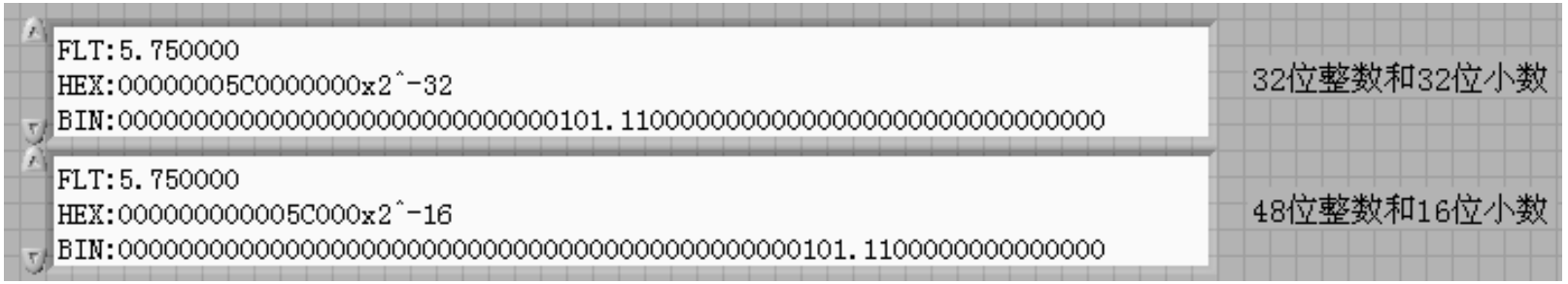
整数的二进制表示



其它标量数据类型的类型转换及内存映射

✱ 3.3.1定点数和浮点数的类型转换和内存映射

定点数的二进制表示



小数点后相当于2的负多少次方。

整数和定点数可以选择多种数制显示方式，但双精度和单精度型不可以。

LabVIEW采用了人为设置小数位数的方法，因此使用定点数双方必须约定小数点的位置，否则无法正确解析这个数。

其它标量数据类型的类型转换及内存映射

✱ 3.3.1定点数和浮点数的类型转换和内存映射

浮点数的二进制表示

IEEE 单精度浮点数

符号 Sign	指数 Exponent	尾数 Mantissa
1 bit	8 bits	23 bits

IEEE 双精度浮点数

符号 Sign	指数 Exponent	尾数 Mantissa
1 bit	11 bits	52 bits

其它标量数据类型的类型转换及内存映射

✱ 3.3.1定点数和浮点数的类型转换和内存映射

已知：double类型38414.4，求：其对应的二进制表示。

由最高到最低位分别是第63、62、61、……、0位：
最高位63位是符号位，1表示该数为负，0表示该数为正；62-52位，一共11位是指数位；51-0位，一共52位是尾数位。
步骤：按照IEEE浮点数表示法，下面先把38414.4转换为十六进制数。
把整数部和小数部分开处理:整数部直接化十六进制：960E，
小数的处理:0.4=0.5*0+0.25*1+0.125*1+0.0625*0+……，直到加上前面的整数部分算够53位就行了。

38414.4(d)=1001011000001110.0110011001100110011001100110011001100(b)

科学记数法为：1.0010110000011100110011001100110011001100110011001100，右移了15位，所以指数为15。

即：**1.0010110000011100110011001100110011001100110011001100×2^15**

按IEEE标准一共11位，可以表示范围是**-1024 ~ 1023**。因为指数可以为负，为了便于计算，规定都先加上1023(2^10-1)，在这里，阶码：15+1023=1038。二进制表示为：100 00001110；

符号位：因为38414.4为正对应为0；
合在一起（注：尾数二进制最高位的1不要，隐藏位技术：最高位的1不写入内存）：
01000000 11100010 11000001 110 01100 11001100 11001100 11001100 11001100

其它标量数据类型的类型转换及内存映射

✱ 3.3.1定点数和浮点数的类型转换和内存映射

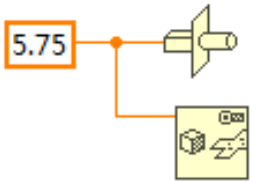
浮点数的二进制表示

IEEE 单精度浮点数

符号 Sign	指数 Exponent	尾数 Mantissa
1 bit	8 bits	23 bits

IEEE 双精度浮点数

符号 Sign	指数 Exponent	尾数 Mantissa
1 bit	11 bits	52 bits

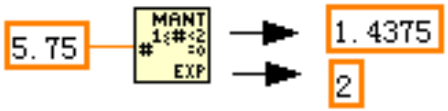


4017 0000 0000 0000

0100 0000 0001 0111 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000

0 10000000001 0111 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000

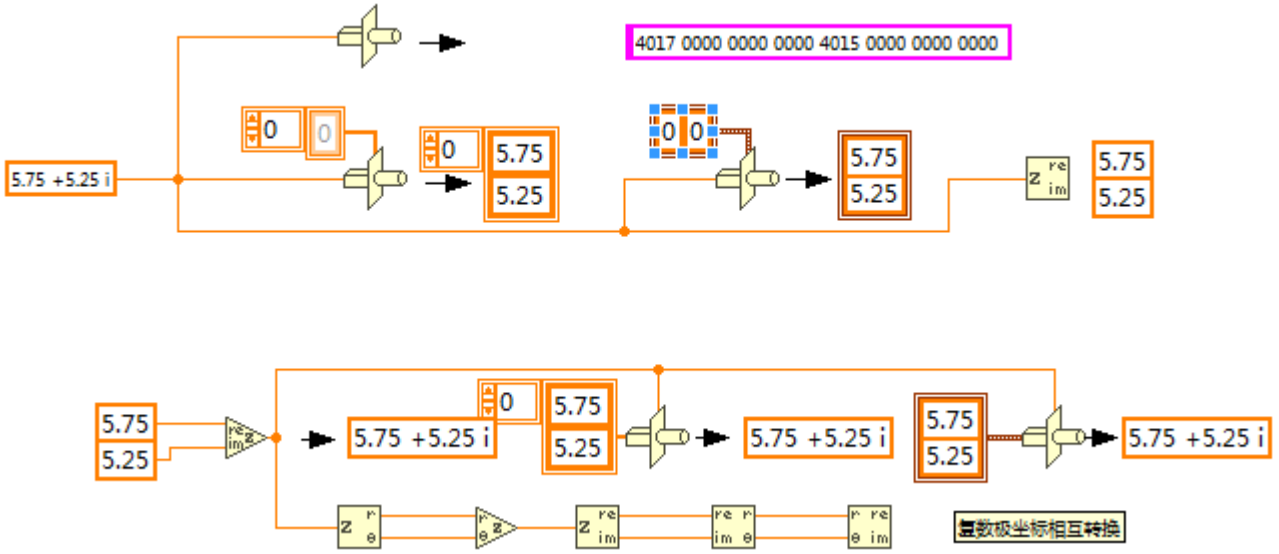
\+ 1025-1023=2 1.0111



其它标量数据类型的类型转换及内存映射

✱ 3.3.2复数的类型转换及内存映射

双精度复数的实部和虚部是连续存储的。



其它标量数据类型的类型转换及内存映射

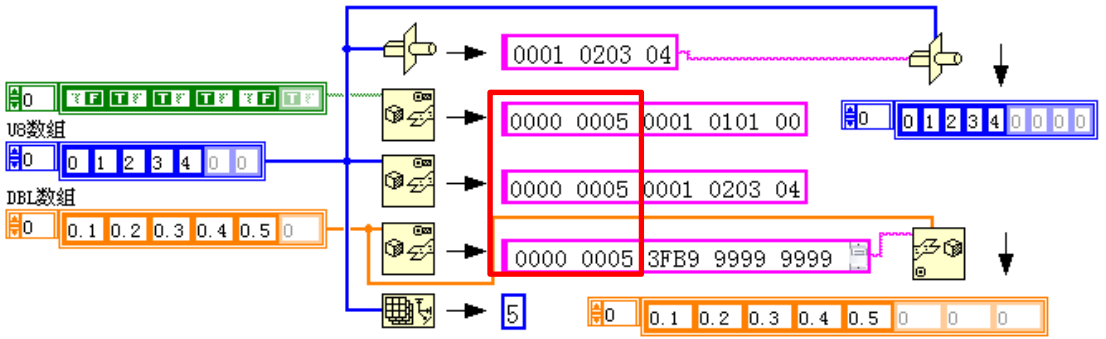
✱ 3.3.3时间标识符内存映射

4个**I32**组成的簇。

复合数据类型

- **3.4.1** 标量数组及其内存映射

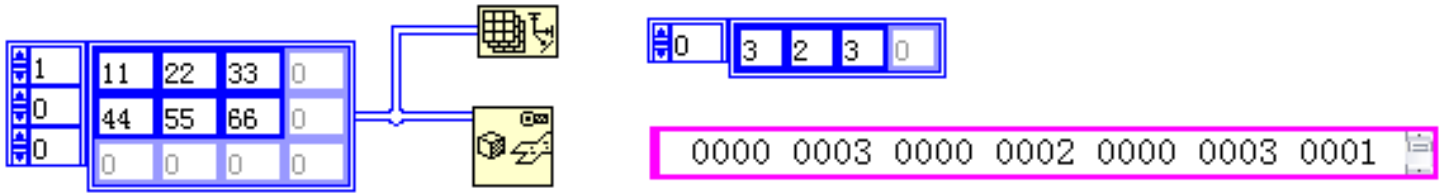
一维标量数组在内存中占据一段连续的存储空间，前**4**字节代表一个**I32**数据，用来表示数组的长度，之后的内存空间用来存储元素。



复合数据类型

- **3.4.1** 标量数组及其内存映射

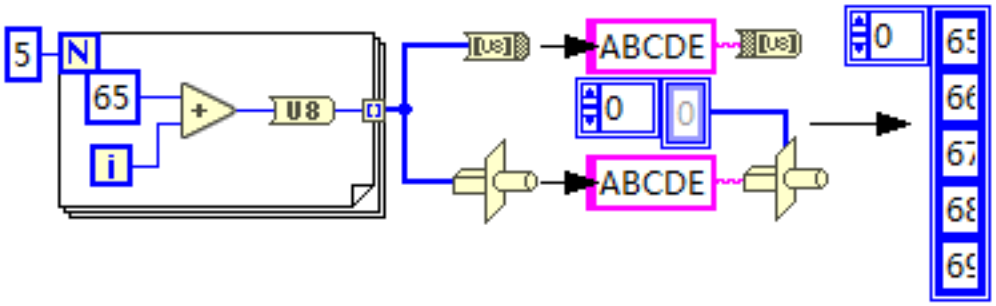
多维数组中每一维长度都需要**4**字节来表示，所以额外需要**DIM*4**字节的空间。

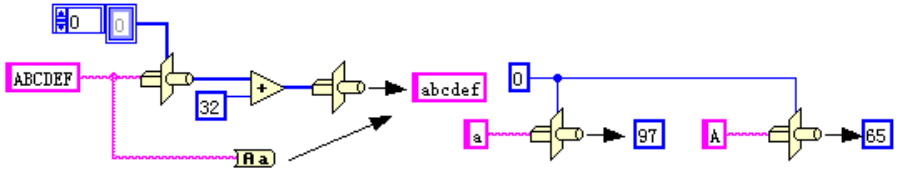


复合数据类型

- **3.4.2字符串、路径和字符串数组的内存映射**

字符串与**U8**数组之间，可以直接进行强制转换。



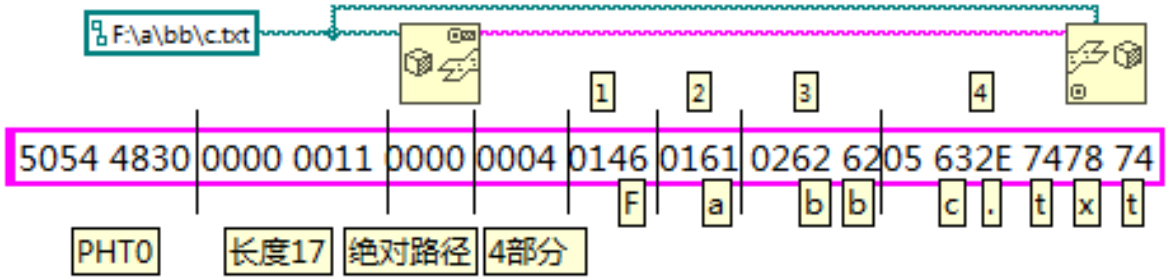


大写转换成小写

复合数据类型

- **3.4.2字符串、路径和字符串数组的内存映射**

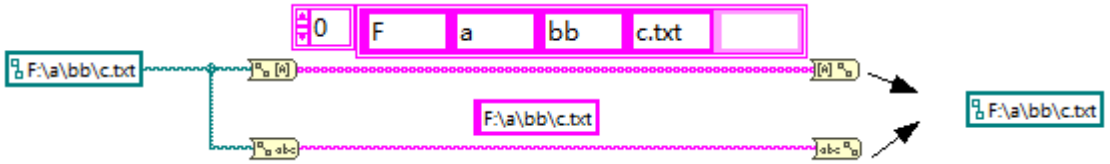
路径无法使用强制类型转换函数，因为它的长度以在内存中占用的实际大小为准。（字符串至少要用**4**字节，以表示字符串的长度。）



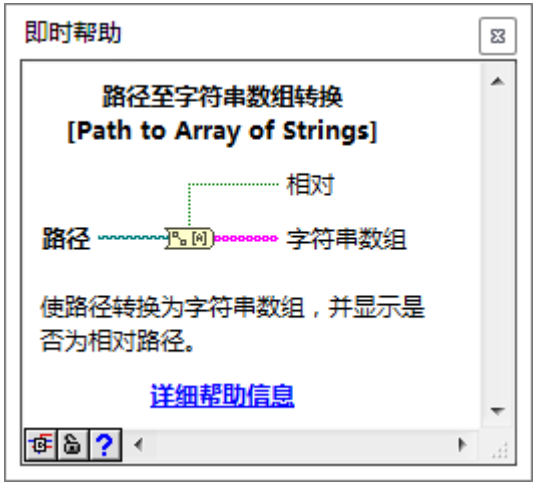
平化后，自动在前面添加**8**字节，前**4**表示**PHT0**，后**4**表示路径长度。**0000**绝对路径，**0001**相对路径。**F: 0146**中 **01**表示字符串长度。

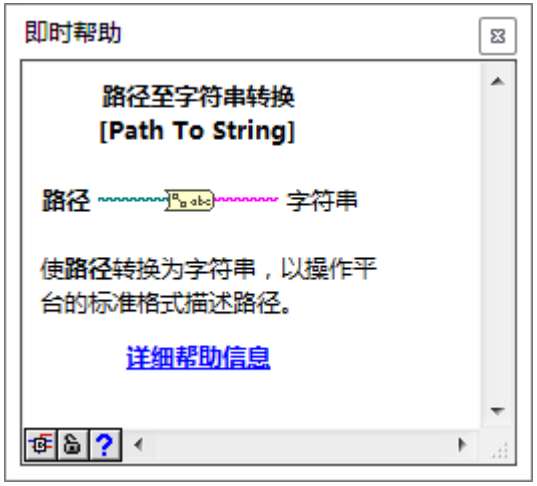
复合数据类型

- **3.4.2字符串、路径和字符串数组的内存映射**



字符串与路径相互转换





复合数据类型

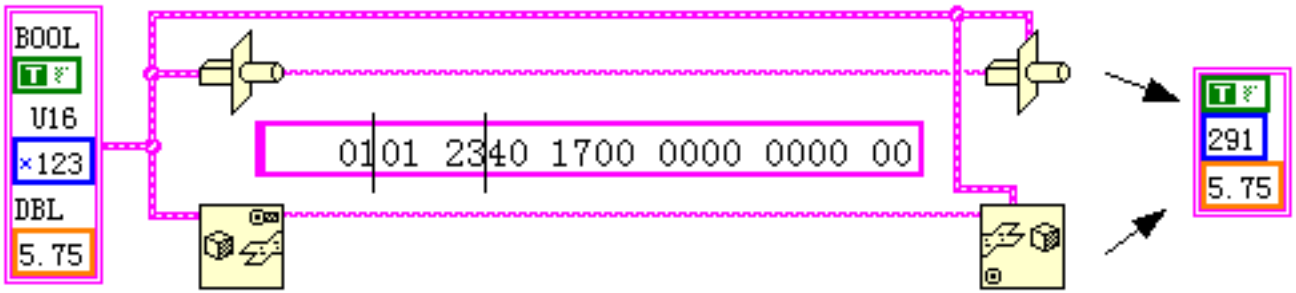
- **3.4.2**字符串、路径和字符串数组的内存映射

字符串数组不是连续存储的，每个字符串都占据独立的内存空间。字符串数组实际保存的是每个字符串的句柄（指向字符串的指针）

平化后，前**4**个字节表示一维字符串数组的长度。接下来是字符串数组中，每个元素平化后的字符串。

簇的内存映射

- **3.5.1** 由标量组成的簇

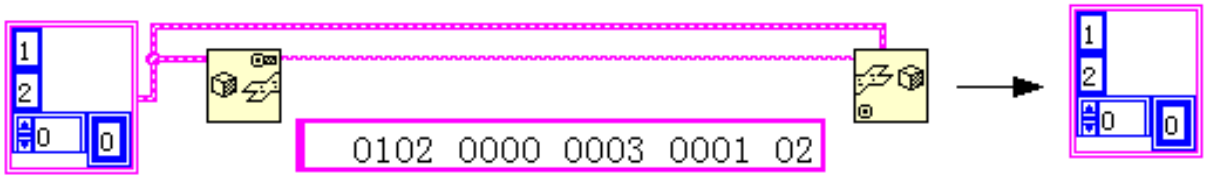


标量组成的簇，连续存储。

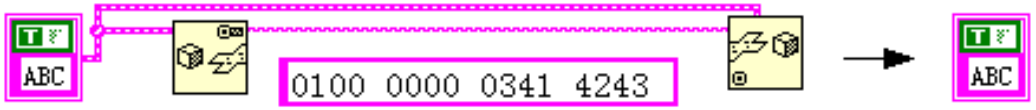
簇的内存映射

- **3.5.2** 包含数组和字符串的簇

由于包含句柄，无法使用强制类型转换函数。



4字节表示**U8**数组的长度



4字节表示字符串的长度

类型描述符

- **3.6.1** 类型描述符的基本构成要素
- **3.6.2** 常用类型描述符列表
- **3.6.3** 常见数据类型的类型描述符结构