Labview宝典

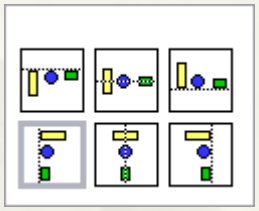
入门篇

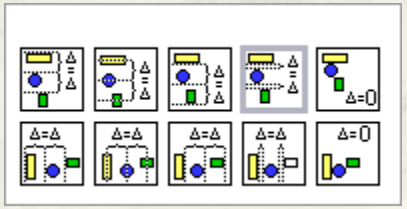
高级篇

应用篇

入门篇

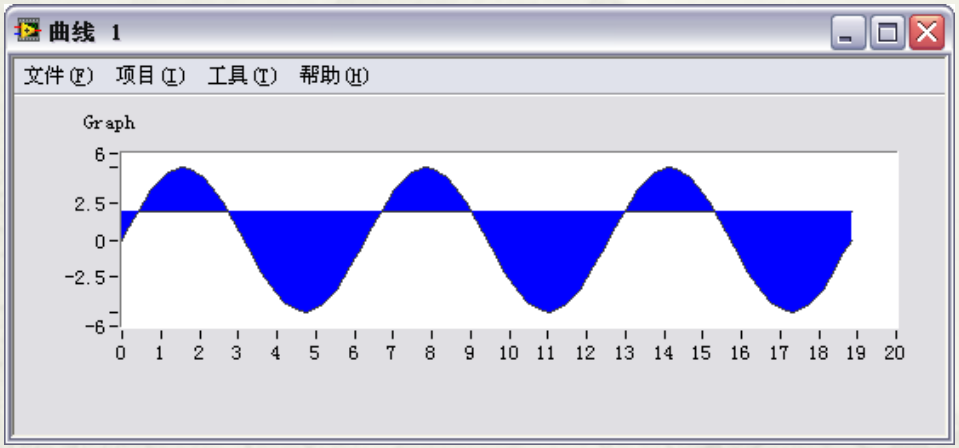
- 第一章：打开LabVIEW编程之门
- 第二章：LabVIEW基本函数
- 第三章：LabVIEW的程序运行结构
- 第四章：LabVIEW的数据结构及内存优化
- 第五章：字符串与文件存储

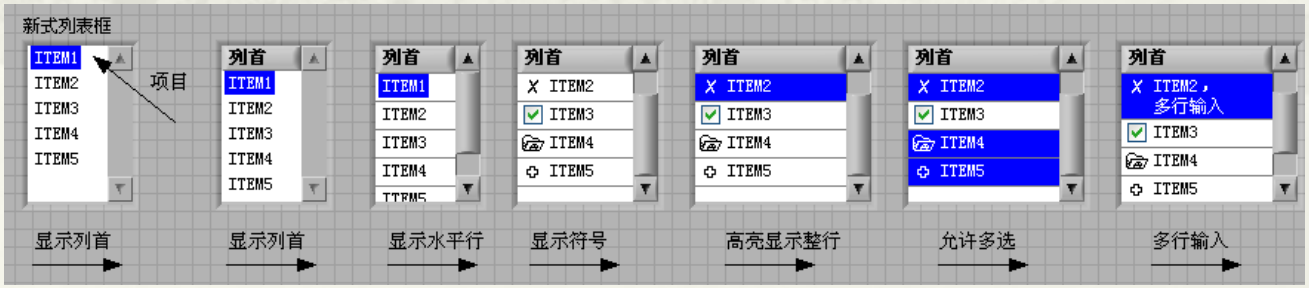




高级篇

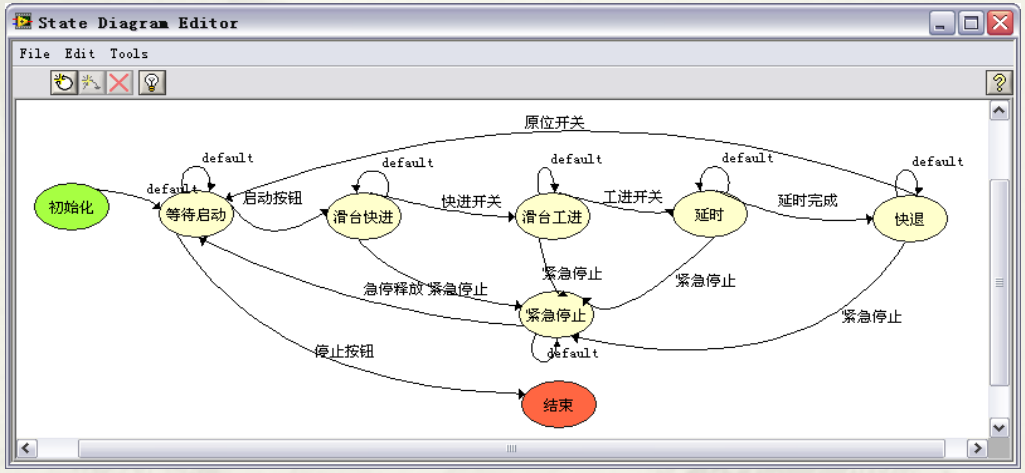
- 第六章：属性节点、方法节点及引用
- 第七章：高级控件的运用
- 第八章：文本编程与外部接口
- 第九章：MathScript
- 第十章：基于组件的程序结构
- 第十一章：人机交互与编程风格

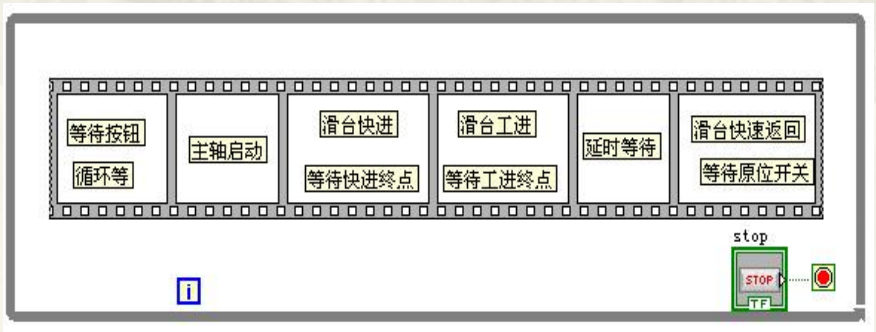


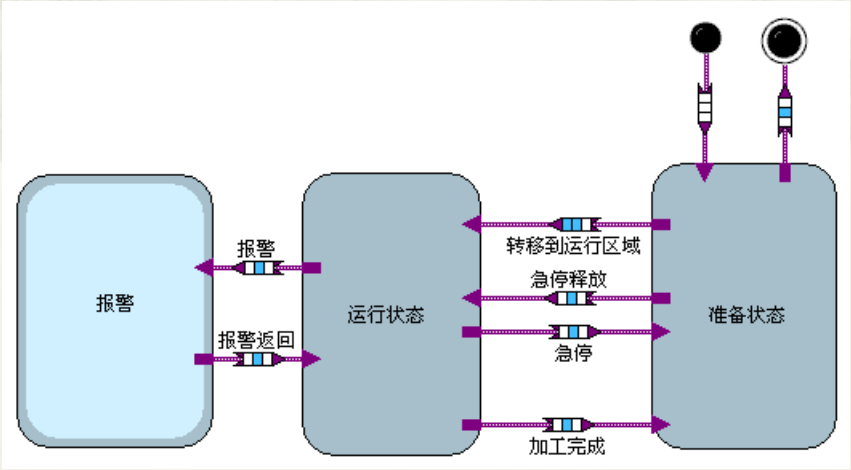


应用篇

- 第十二章：VI模板、设计模式、状态图
- 第十三章：串并口通讯、网络与DSC
- 第十四章：数据库、报表工具
- 第十五章：LabVIEW与RT系统
- 第十六章：LabVIEW与数据采集
- 第十七章：FPGA工具包

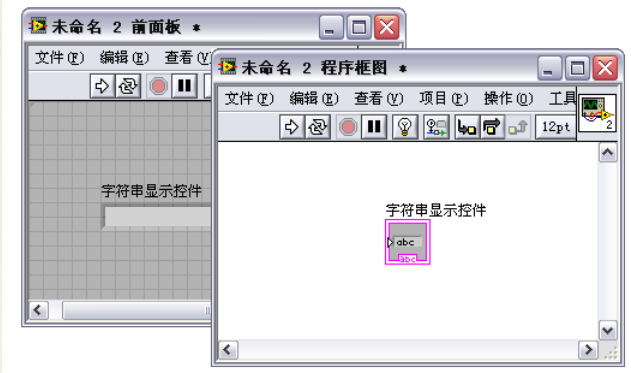


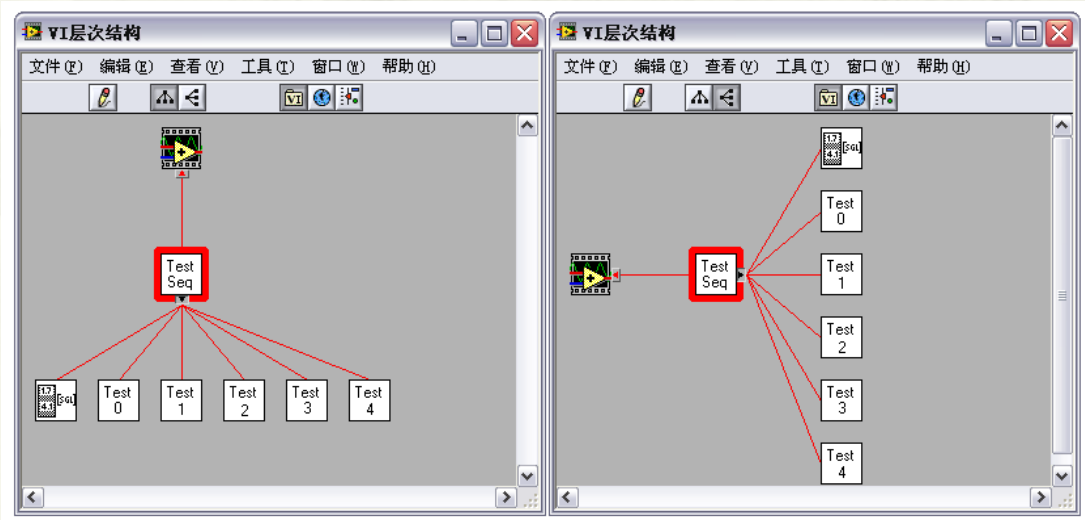




第1章：打开 LabVIEW编程之门

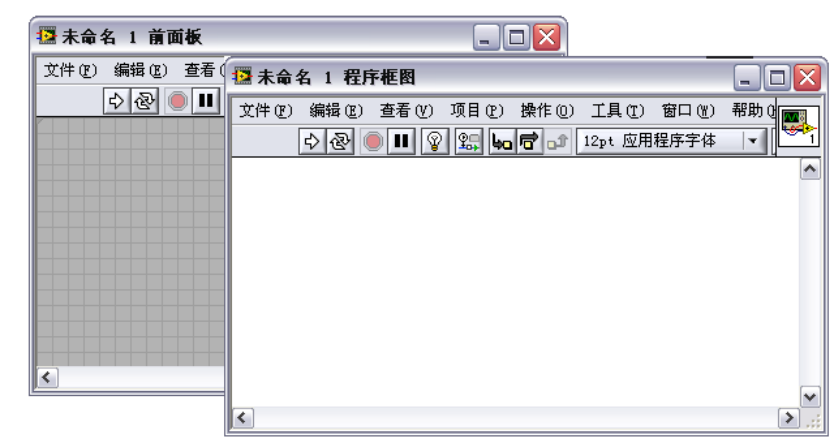
- 1.1 从VI开始
- 1.2 编辑前面板和程序框图
- 1.2 VI及其属性对话框
- 1.4 基本控件及其使用方法
- 1.5 小结

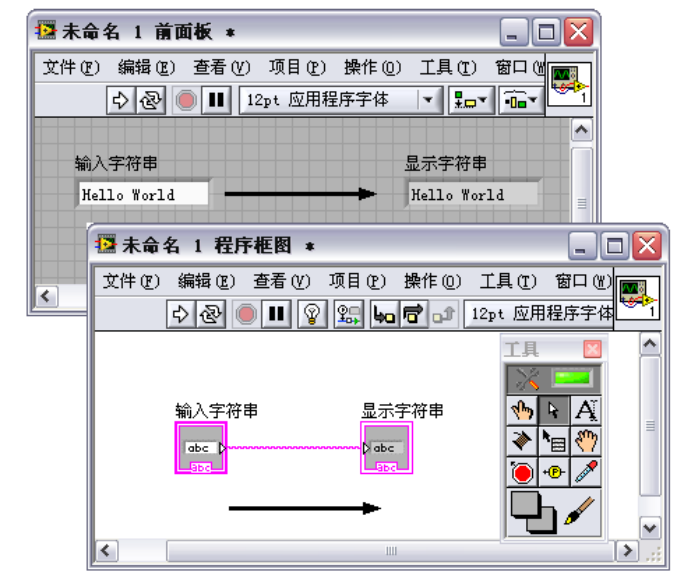




1.1 从VI开始

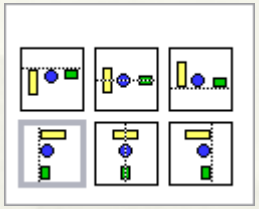
- 1.1.1 如何创建VI
- 1.1.2 控件属性设置与快捷菜单
- 1.1.3创建控件、常量、局部变量、属性节点的常用方法
- 1.1.4 创建自定义控件

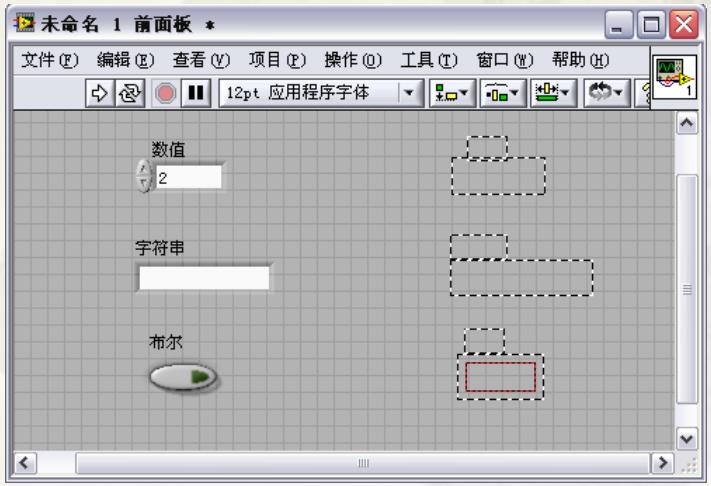


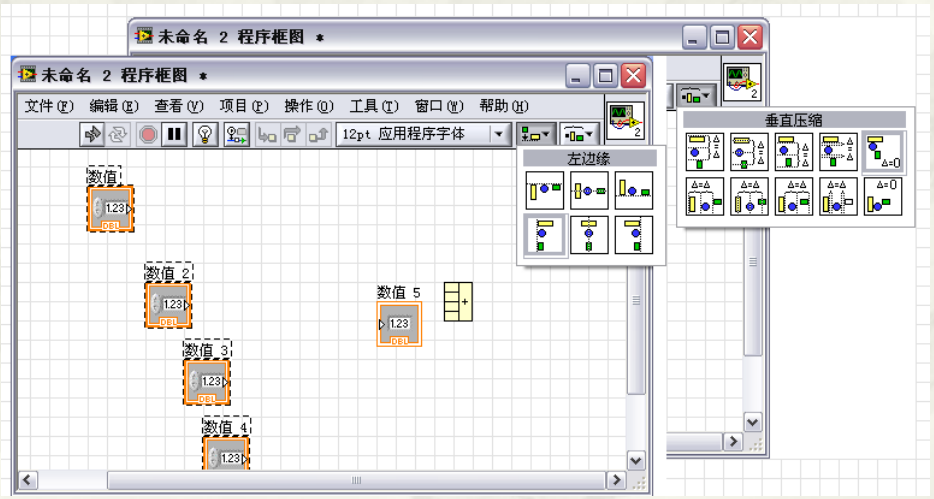


1.2 编辑前面板和程序框图

* 1.2.1 选取、移动和删除对象
* 1.2.2 使用布局工具

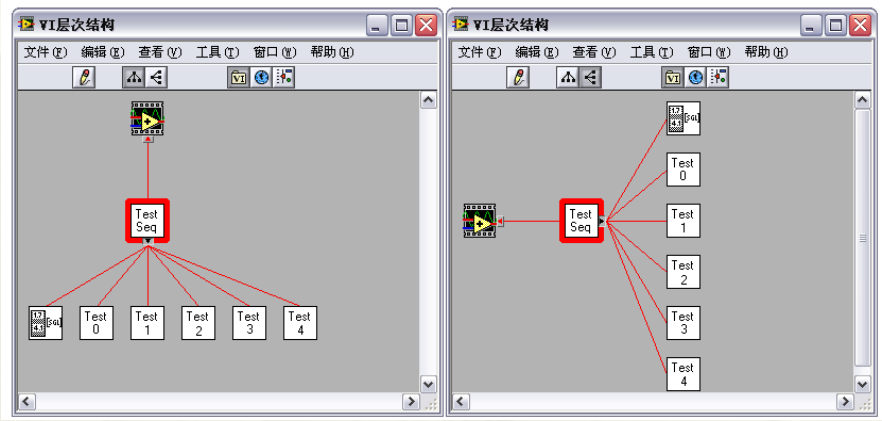


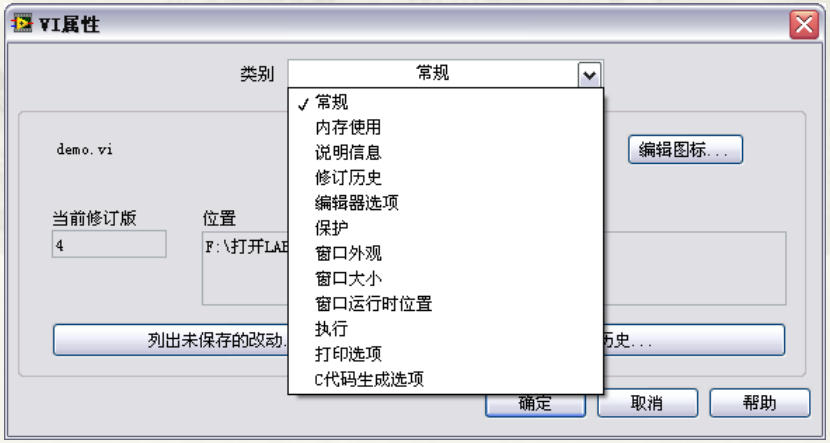


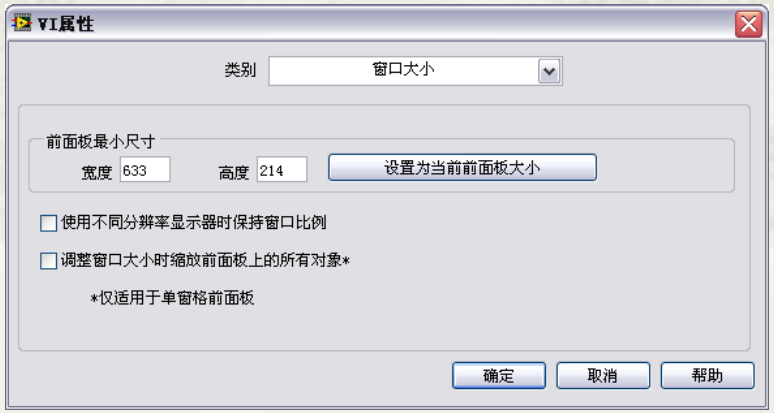


1.3 VI及其属性对话框

- 1.3.1 VI的层次结构
- 1.3.2 调用子VI
- 1.3.3 VI的属性设置

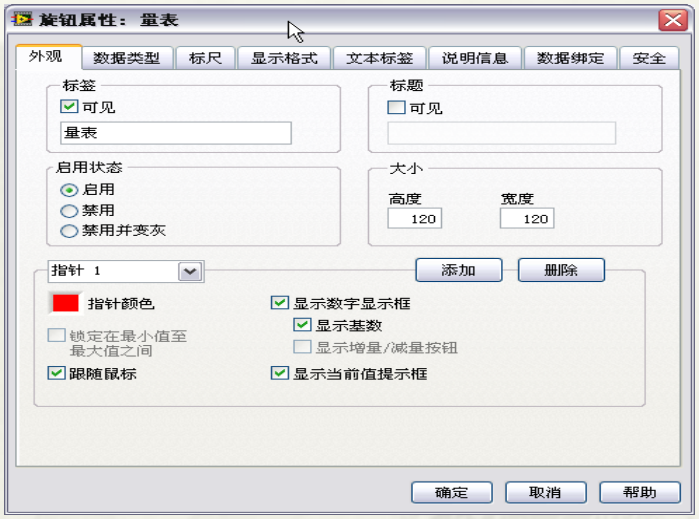


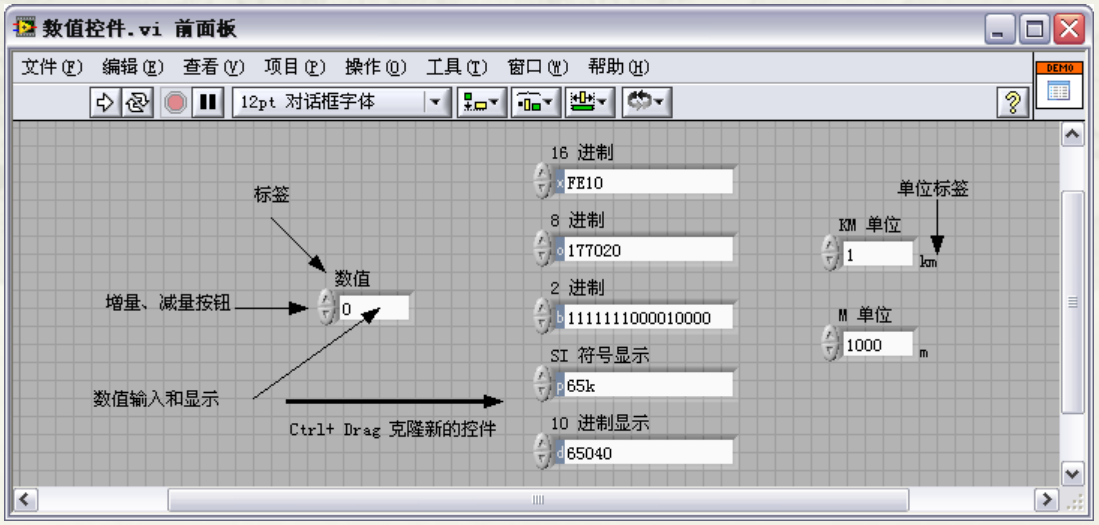




1.4 基本控件及其使用方法

- 1.4.1 基本数值控件及其属性设置
- 1.4.2 基本布尔控件及其属性设置
- 1.4.3 控件的通用编辑方法
- 1.4.4 字符串和路径控件
- 1.4.5下拉列表与枚举控件
- 1.4.6 数组控件及其属性设置
- 1.4.7 簇控件
- 1.4.8 时间标识控件与波形数据控件





第2章 LabVIEW基本函数

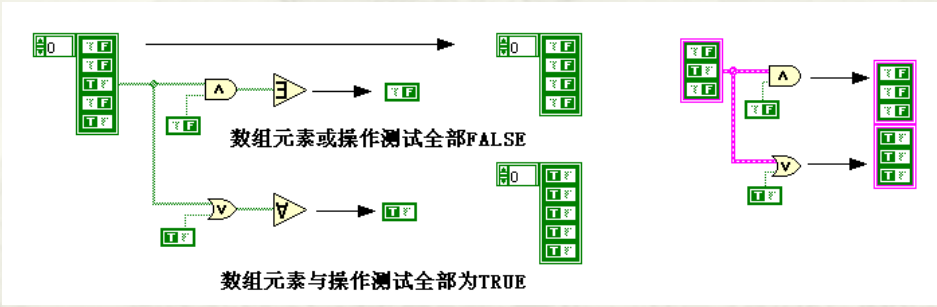
* 2.1 必须了解的一些基本算术运算节点函数
* 2.2 必须了解的位运算函数和逻辑运算函数
* 2.3 必须了解的关系运算函数和比较节点函数
* 2.4 小结

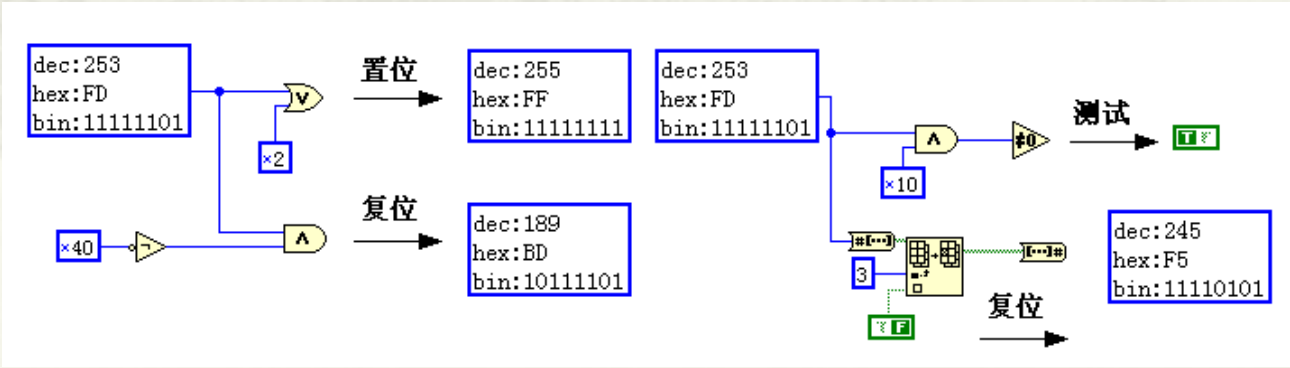
2.1 必须了解的一些基本算术运算节点函数

- 2.1.1 LabVIEW支持的基本数据类型
- 2.1.2 基本运算符函数节点
- 2.1.3 标量与标量的基本运算
- 2.1.4 标量与数组
- 2.1.5 数组与数组的运算
- 2.1.6 数组的函数运算
- 2.1.7 标量与簇的基本运算
- 2.1.8 簇与簇的运算
- 2.1.9簇的节点函数

2.2 必须了解的位运算函数和逻辑运算函数

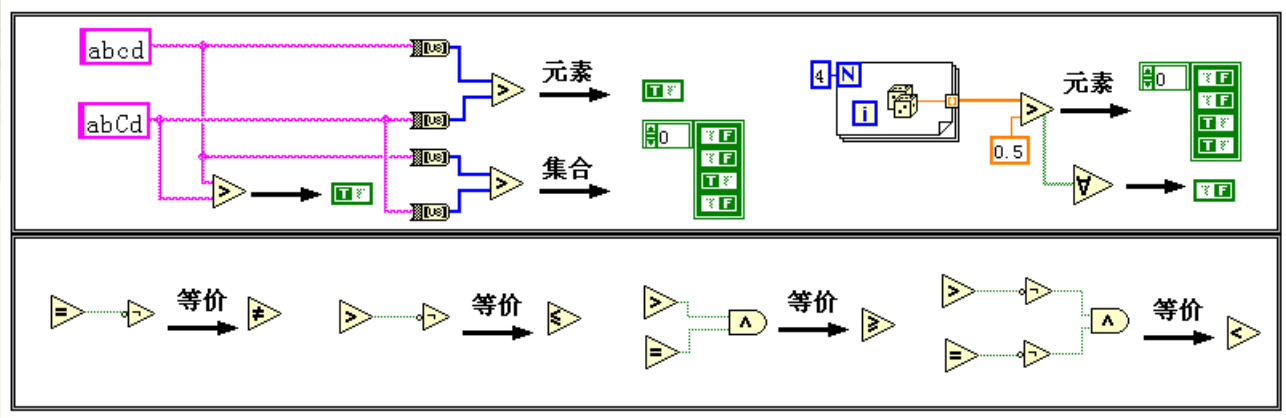
- 2.2.1 常用逻辑运算函数
- 2.2.2 位运算
- 2.2.3 深入理解复合运算节点函数





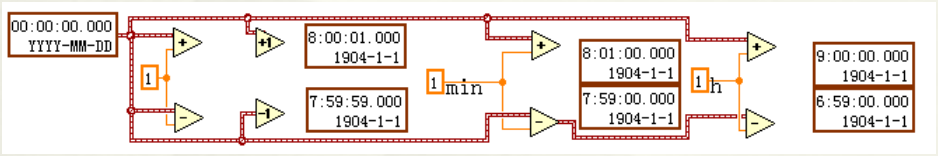
2.3 必须了解的关系运算函数和比较节点函数

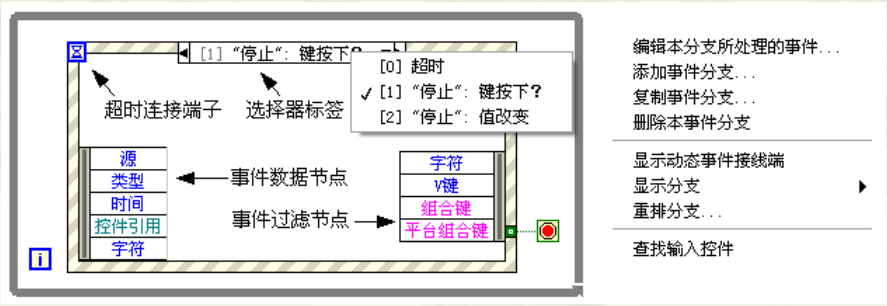
- 2.3.1 比较模式
- 2.3.2 通用关系运算函数
- 2.3.3 比较0关系运算节点函数
- 2.3.4 复杂关系运算节点函数
- 2.3.5 字符关系运算节点函数
- 2.3.6 表达式节点与公式快速VI

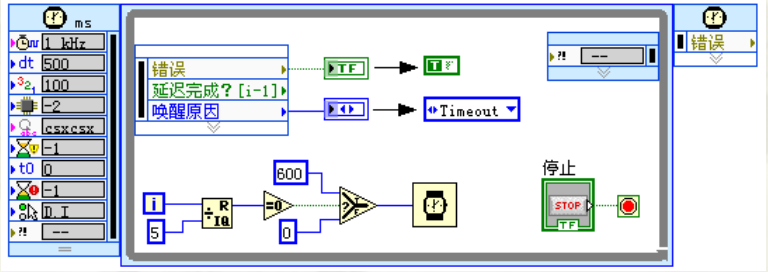


第3章 LabVIEW的程序运行结构

- 3.1 两种不同的循环结构
- 3.2 定时结构
- 3.3 独特的条件结构
- 3.4 不和谐的顺序结构
- 3.5 禁用部分程序框图结构
- 3.6 局部变量、内置全局变量和函数全局变量
- 3.7事件结构

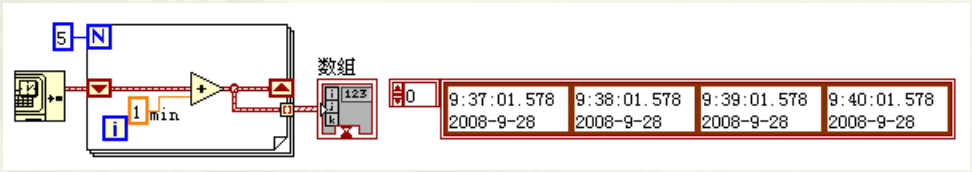


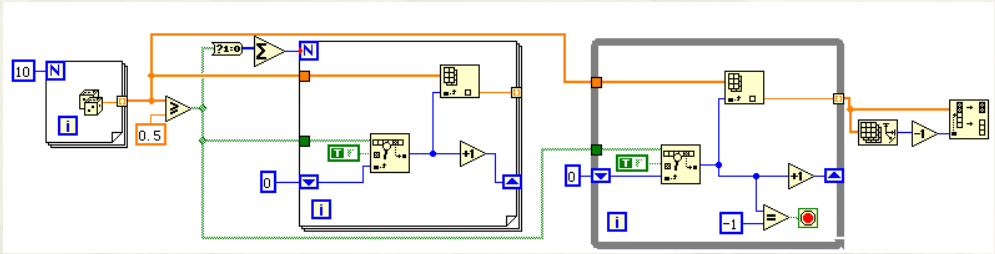


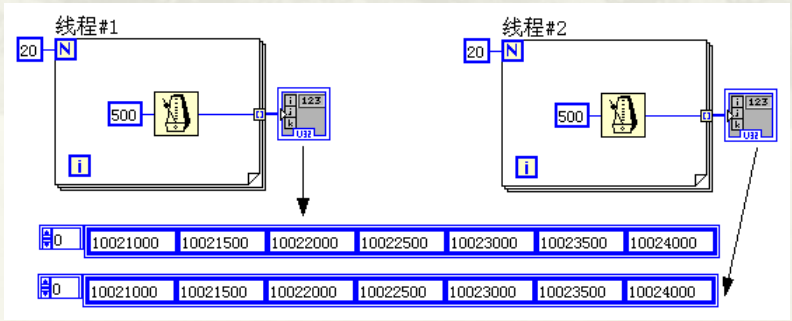


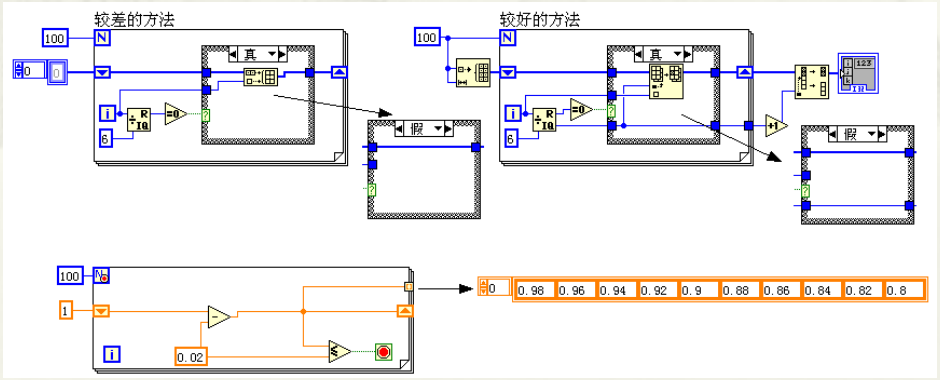
3.1 两种不同的循环结构

- 3.1.1 For循环的组成和特点
- 3.1.2 For循环与数组
- 3.1.3 For循环与移位寄存器
- 3.1.4 For循环中的continue和break
- 3.1.5 While循环不仅仅是循环
- 3.1.6 While循环与定时
- 3.1.7 反馈节点



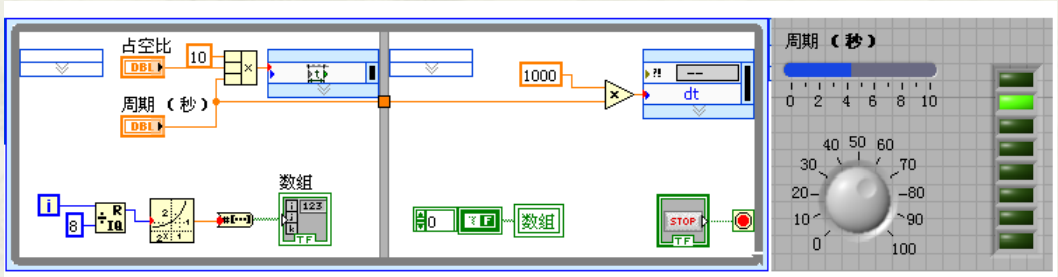


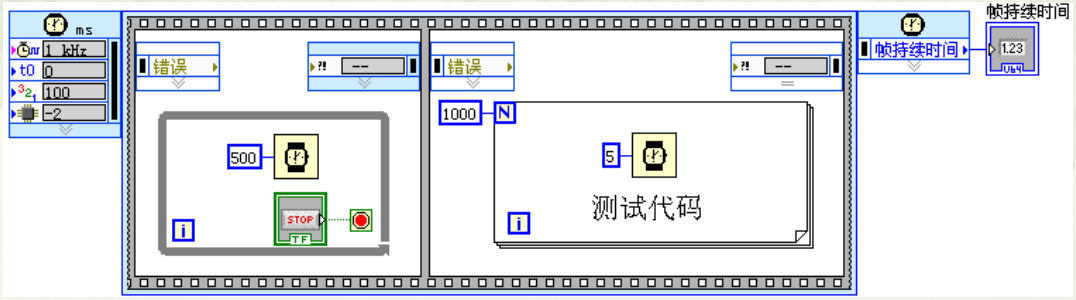




3.2 定时结构

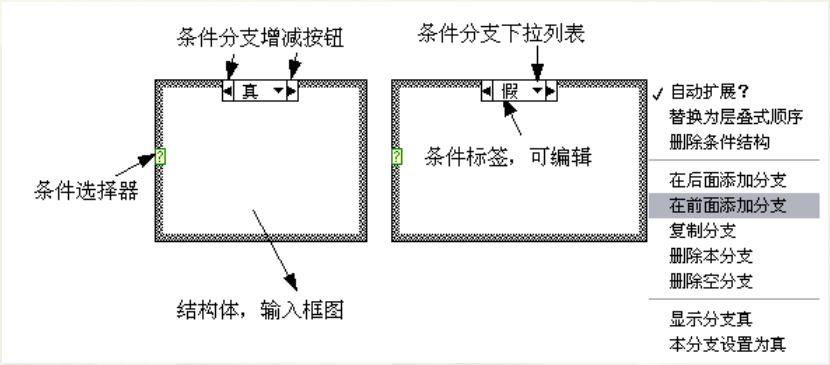
- 3.2.1定时循环的基本组成要素和配置对话框
- 3.2.2 定时顺序结构

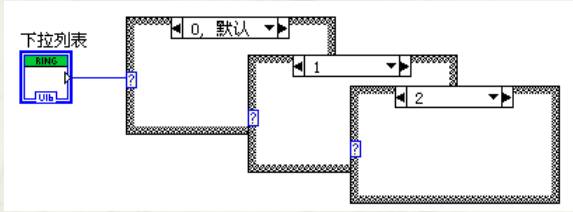


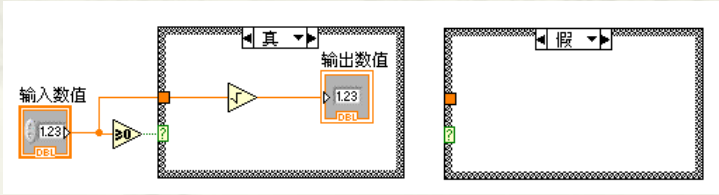


3.3 独特的条件结构

- 3.3.1 条件结构的基本结构
- 3.3.2 布尔型输入
- 3.3.3 错误簇输入
- 3.3.4 数值型输入
- 3.3.5 枚举型输入
- 3.3.6下拉列表输入
- 3.3.7 字符串和组合框输入
- 3.3.8 输入、输出隧道
- 3.3.9多重IF ELSE 的处理方法

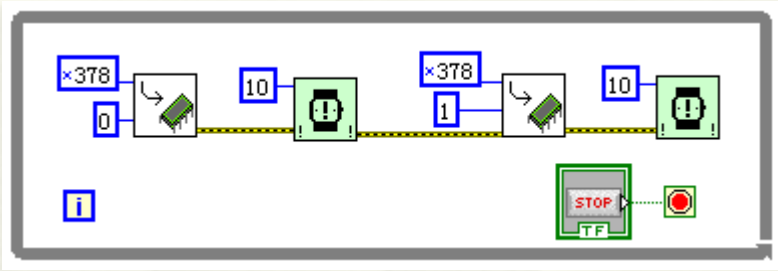


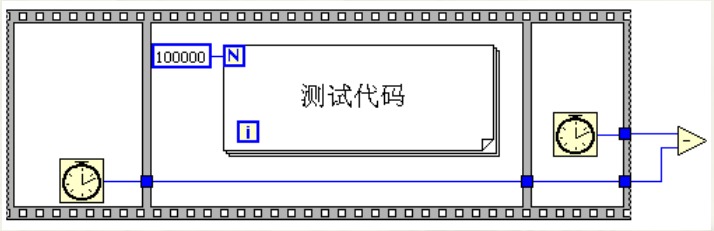


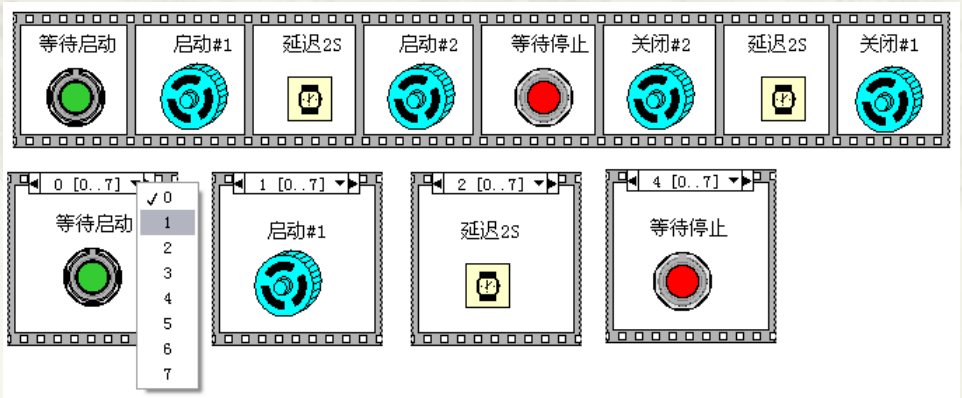


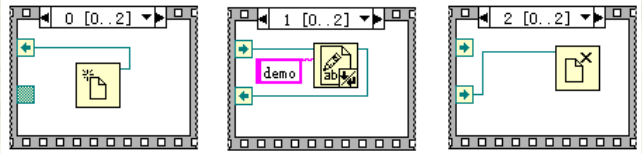
3.4 不和谐的顺序结构

- 3.4.1 多线程运行次序
- 3.4.2 两种不同的顺序结构
- 3.4.3 隧道与顺序局部变量
- 3.4.4 顺序结构的替代
- 3.4.5顺序结构的典型应用



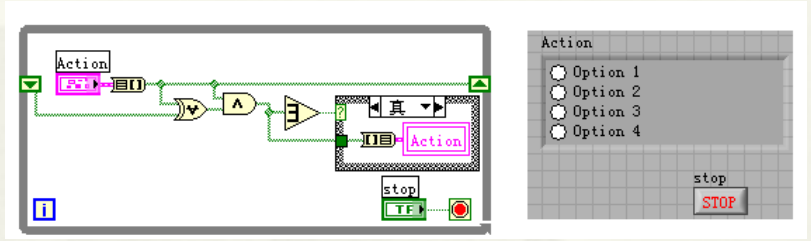


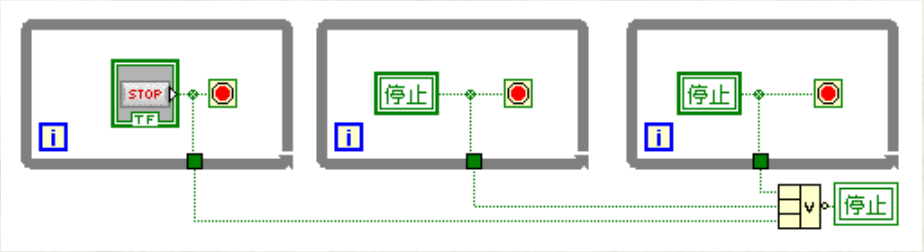


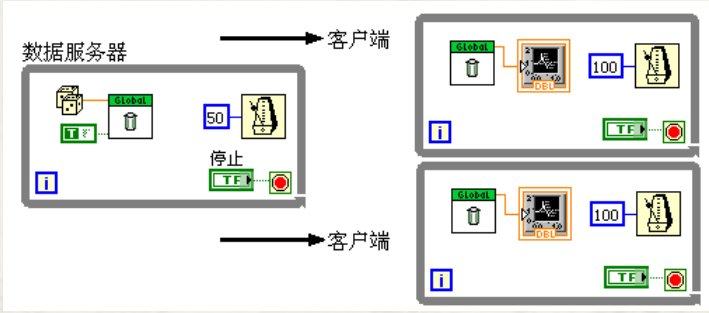


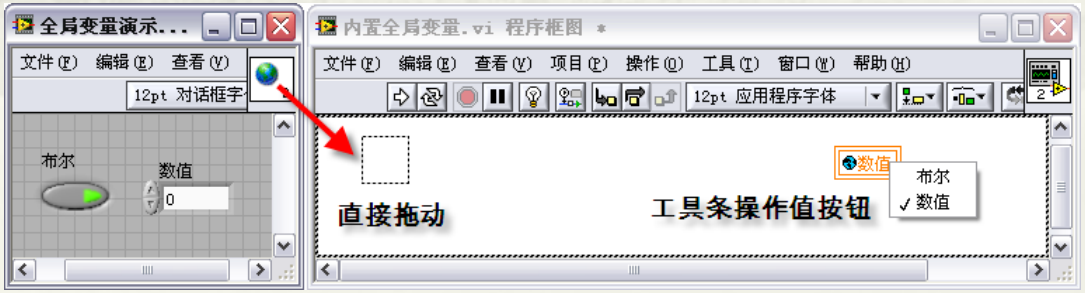
3.6 局部变量、内置全局变量和函数全局变量

* 3.6.1 局部变量
* 3.6.2 内置全局变量
* 3.6.3 函数全局变量



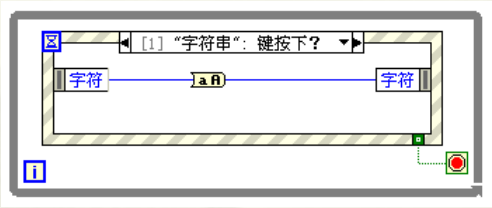


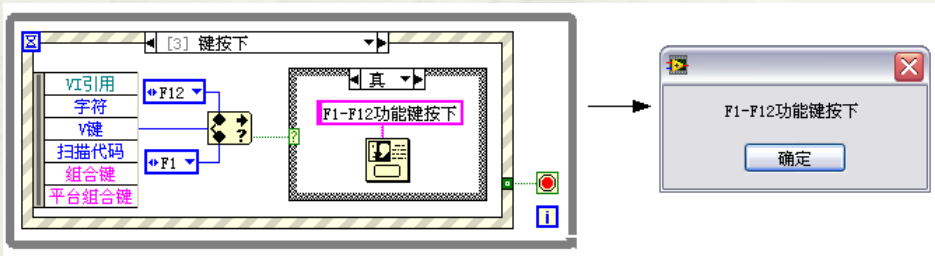


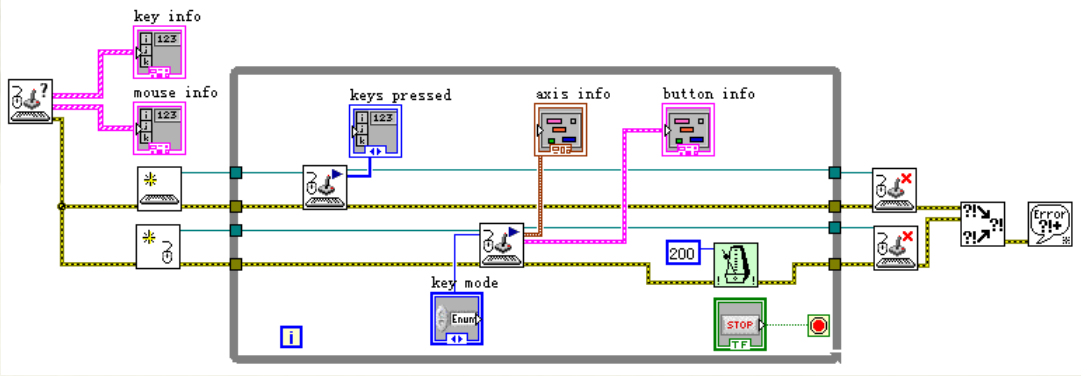


3.7 事件结构

- 3.7.1 事件结构的基本构成和创建方法
- 3.7.2 事件的分类及其特点
- 3.7.3 事件结构之间的数据传送与共享
- 3.7.4 事件发生的次序、过滤和转发
- 3.7.5 正确地使用事件结构

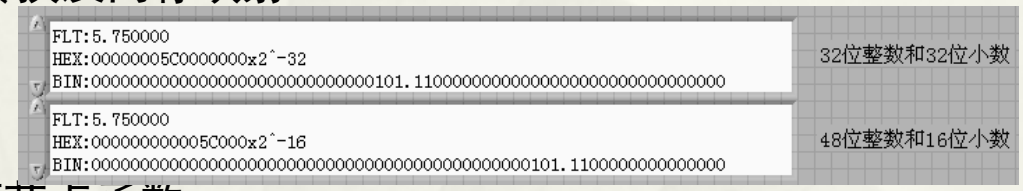


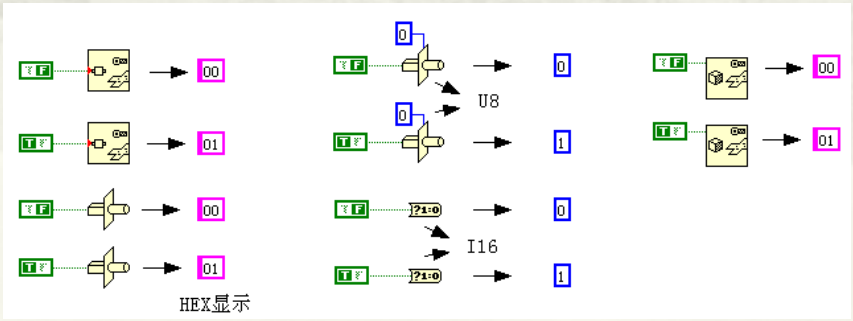


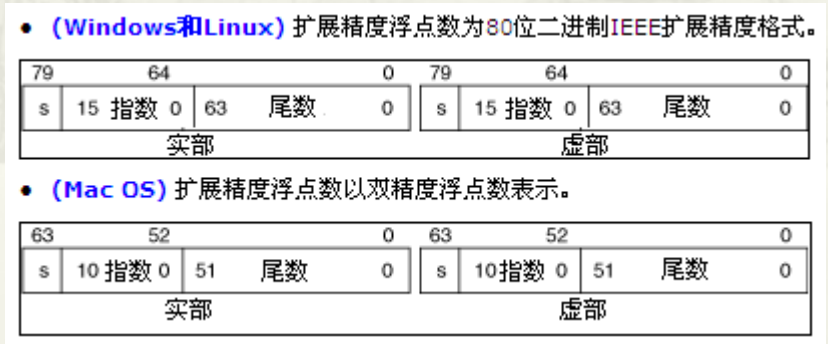


第4章 LabVIEW的数据结构及内存优化

- 4.1 几种常用的数据类型转换节点函数
- 4.2 整数的类型转换及内存映射
- 4.3 其它标量数据类型的类型转换及内存映射
- 4.4 复合数据类型
- 4.5 簇的内存映射
- 4.6 类型描述符
- 4.7 Openg中的有关类型描述符节点函数
- 4.8 几种常用的内存分析工具和方法
- 4.9 影响VI运行速度的因素

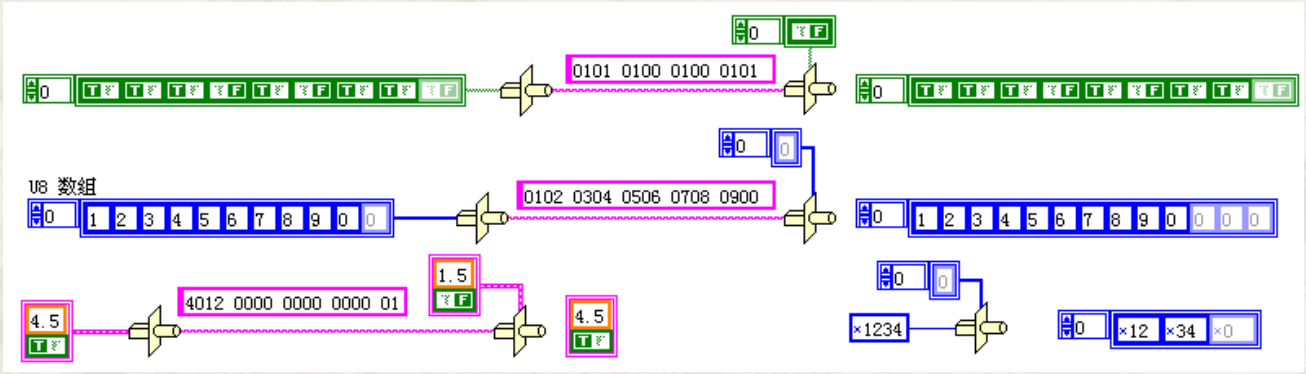


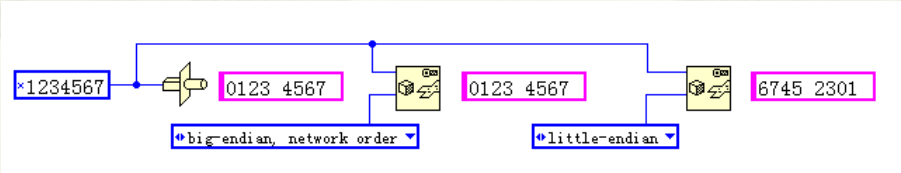




4.1 几种常用的数据类型转换节点函数

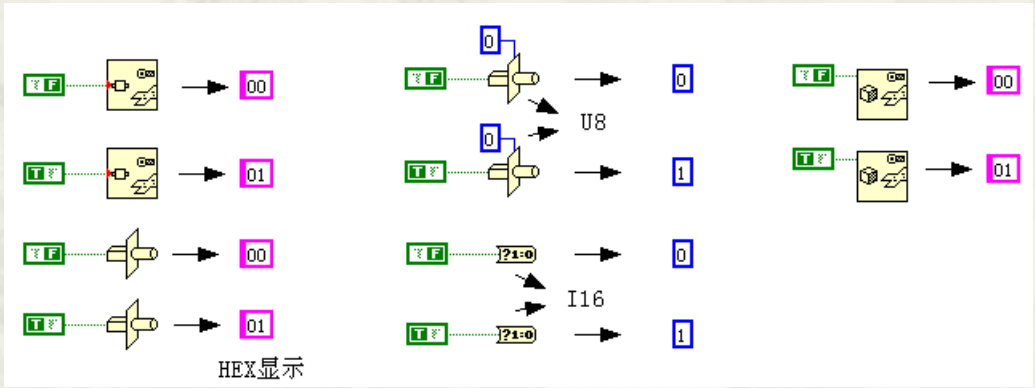
- 4.1.1 强制类型转换函数
- 4.1.2 平化数据至字符串及字符串还原平化数据函数
- 4.1.3 变体类型数据

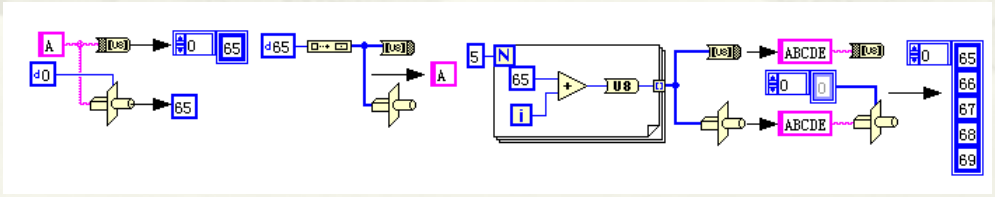




4.2 整数的类型转换及内存映射

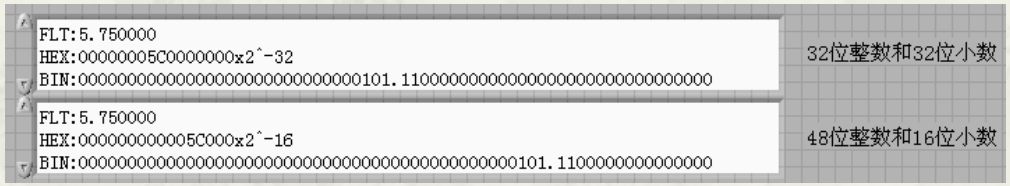
- 4.2.1 布尔类型与字符串和数值的相互转换
- 4.2.2 U8类型与字符串
- 4.2.3 其它整数的相互转换





4.3 其它标量数据类型的类型转换及内存映射

- 4.3.1定点数和浮点数的类型转换和内存映射
- 4.3.2复数的类型转换及内存映射
- 4.3.3时间标识符内存映射

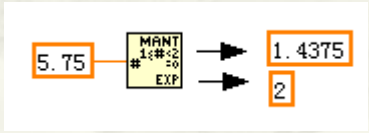


IEEE 单精度浮点数

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

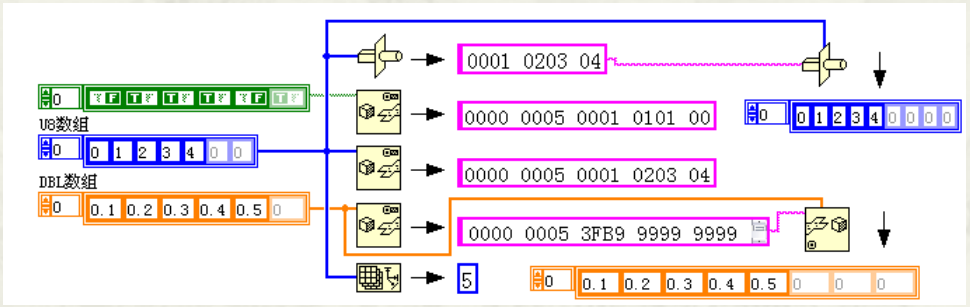
IEEE 双精度浮点数

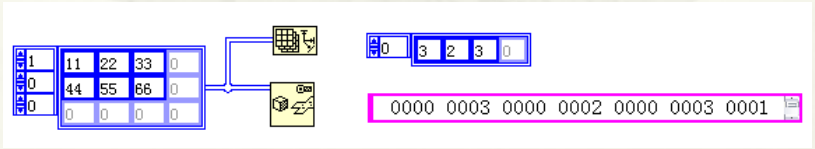
符号 Sign	指数 Exponent	尾数 Mantissa
1 bit	11 bits	52 bits

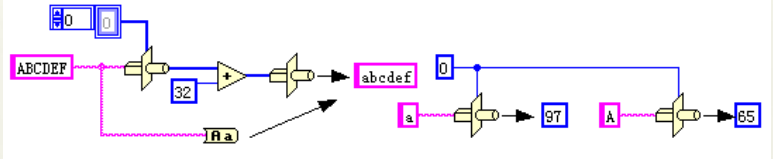


4.4 复合数据类型

- 4.4.1 标量数组及其内存映射
- 4.4.2字符串、路径和字符串数组的内存映射
- 4.4.3 LabVIEW使用的编码

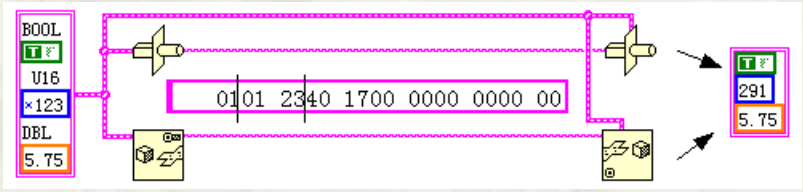


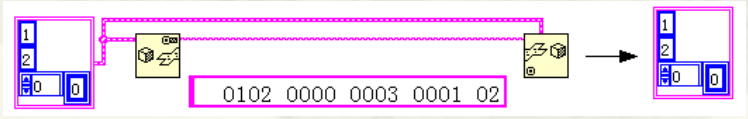


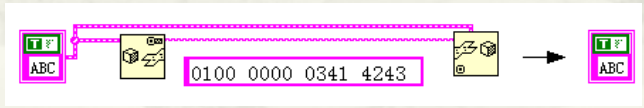


4.5 簇的内存映射

- 4.5.1 由标量组成的簇
- 4.5.2 包含数组和字符串的簇

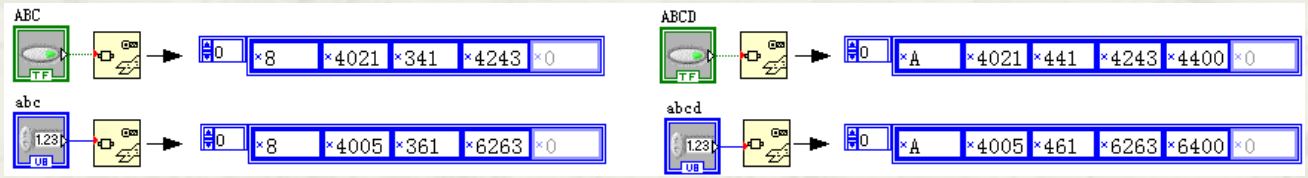


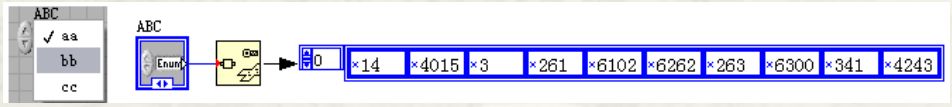


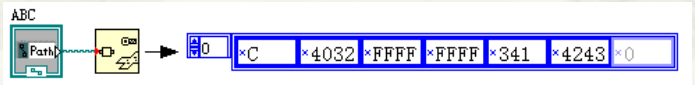


4.6 类型描述符

- 4.6.1 类型描述符的基本构成要素
- 4.6.2 常用类型描述符列表
- 4.6.3 常见数据类型的类型描述符结构

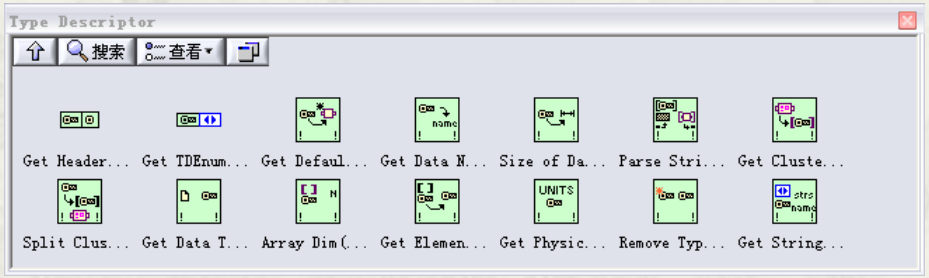


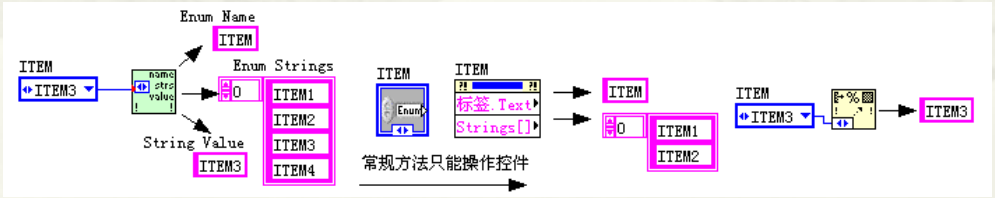




4.7 Openg中的有关类型描述符节点函数

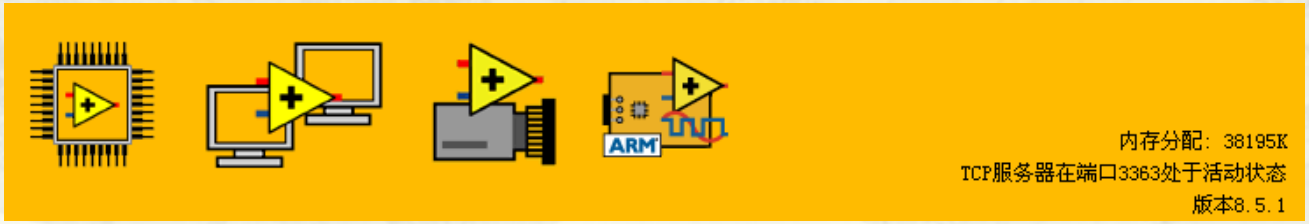
- 4.7.1 类型描述符节点
- 4.7.2 利用类型描述符处理枚举数据类型
- 4.7.3 利用类型描述符处理簇

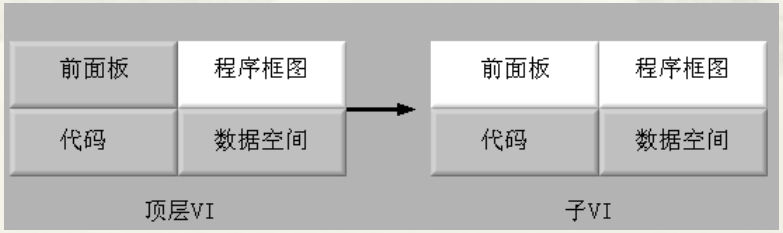




4.8 几种常用的内存分析工具和方法

- 4.8.1 内存的重要性
- 4.8.2 内存和性能查看工具
- 4.8.3 VI 使用的内存
- 4.8.4 优化内存的一般注意事项
- 4.8.5 数组处理与内存优化
- 4.8.6 避免循环中不必要的计算、读写控件或者变量





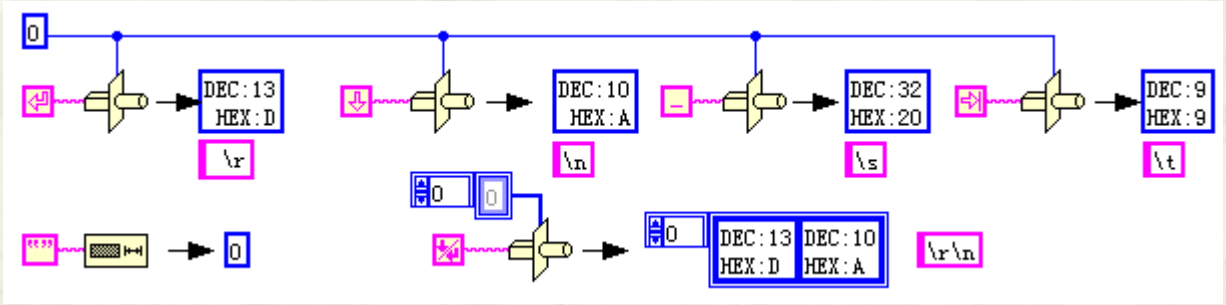
4.9 影响VI运行速度的因素

- 4.9.1 硬件输入输出
- 4.9.2 屏幕显示

屏幕显示更新是影响程序运行速度的另一个关键因素。LabVIEW会智能化的决定对一般的控件是否更新，当LabVIEW判断当前控件的值未发生变化时，是不会更新屏幕显示的。正因为这样，LabVIEW必须时刻关注并判断是否数据发生变化，这在一定程度上也是会影响速度的

第5章 字符串与文件存储

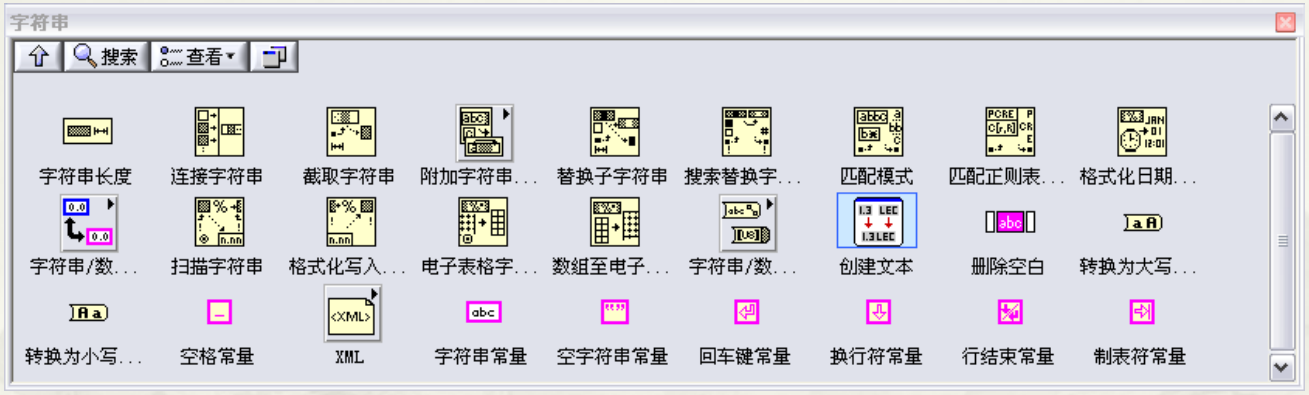
- 5.1 字符串
- 5.2 文件存储

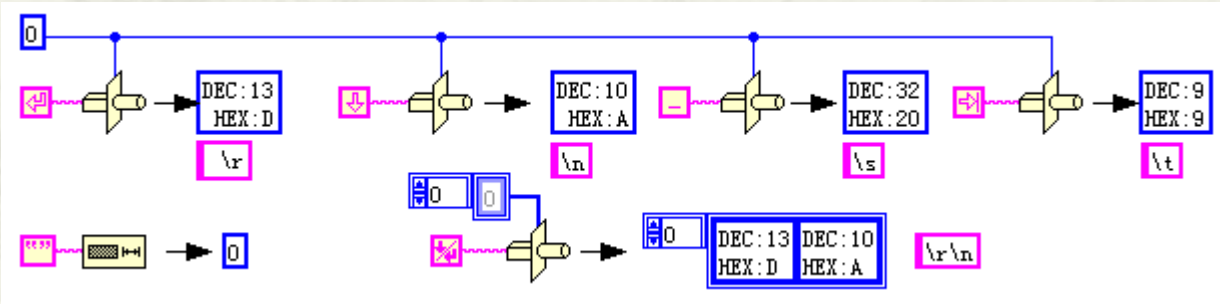


5.1 字符串

- 5.1.1 几种常用的字符串常量
- 5.1.2 几种简单常用的字符串节点函数
- 5.1.3 匹配模式和匹配正则表达式
- 5.1.4 字符串与数值的相互转换
- 5.1.5 功能强大的格式化字符串函数和扫描字符串函数
- 5.1.6 数组与电子表格字符串
- 5.1.7 附加字符串函数

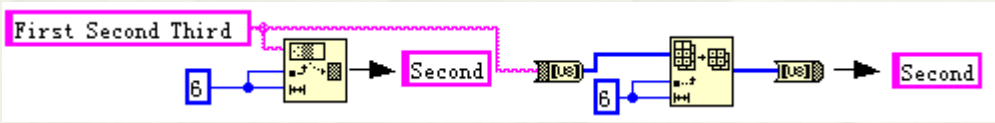
5.1.1 几种常用的字符串常量

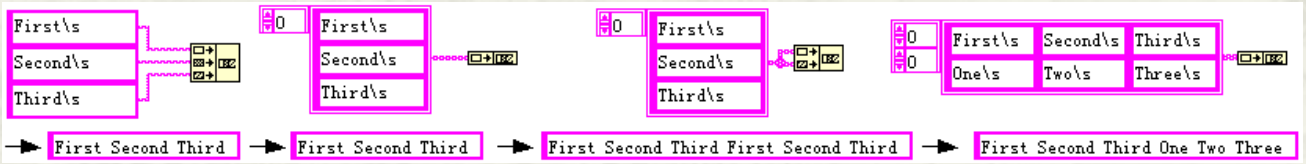


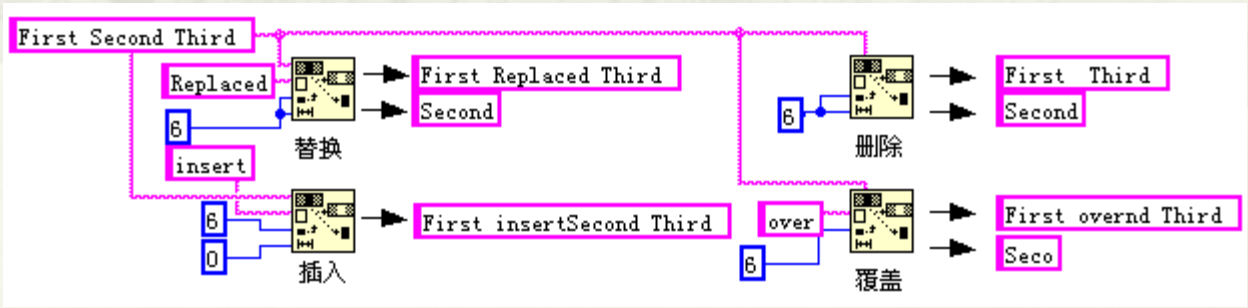


5.1.2 几种简单常用的字符串节点函数

- 字符串长度函数
- 连接字符串函数
- 截取字符串函数
- 替换子字符串函数
- 搜索替换子字符串函数

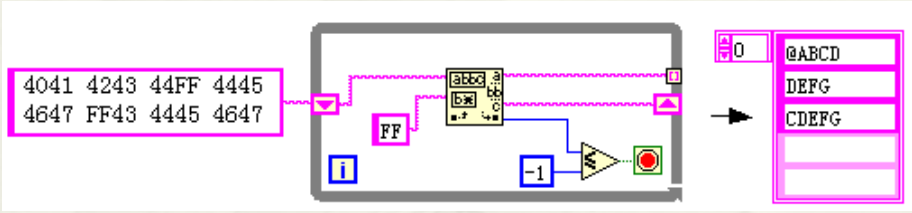


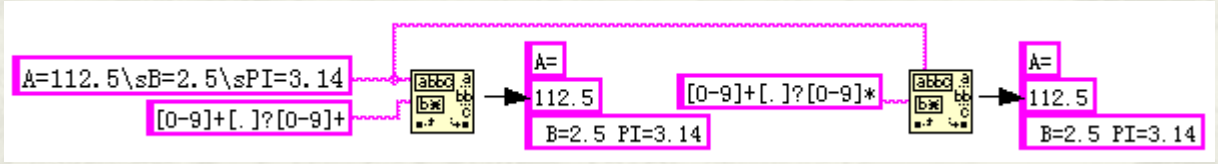


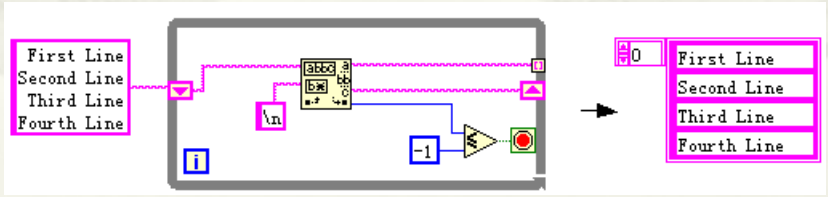


5.1.3 匹配模式和匹配正则表达式

- 特征字符串
- 匹配确定字符串
- 匹配数字
- 提取字符串中多个数值

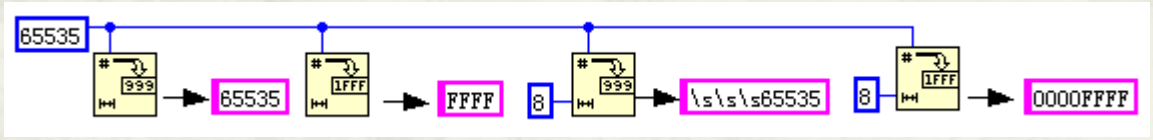


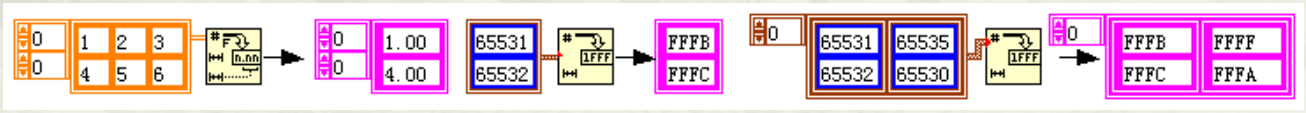


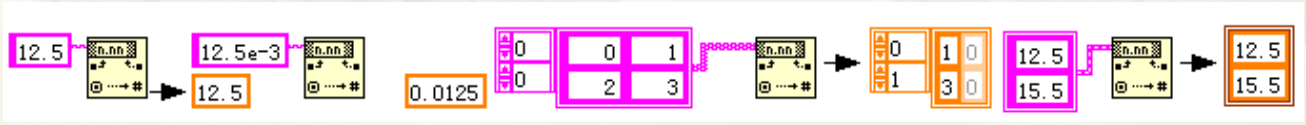


5.1.4 字符串与数值的相互转换

- 整数转换成字符串
- 浮点数转换成字符串
- 字符串转换成数值
- 字符串转换成数值的多态操作

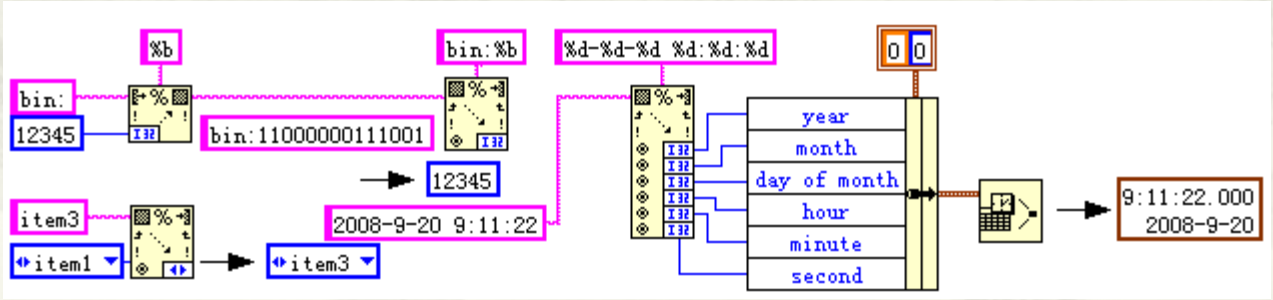




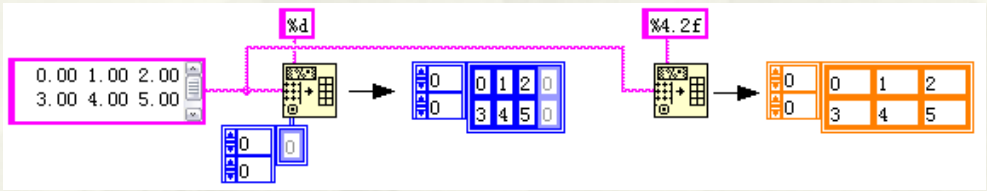


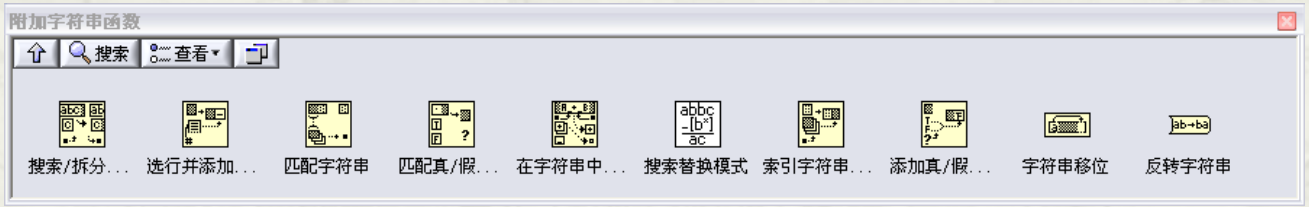
5.1.5功能强大的格式化字符串函数和扫描字符串函数

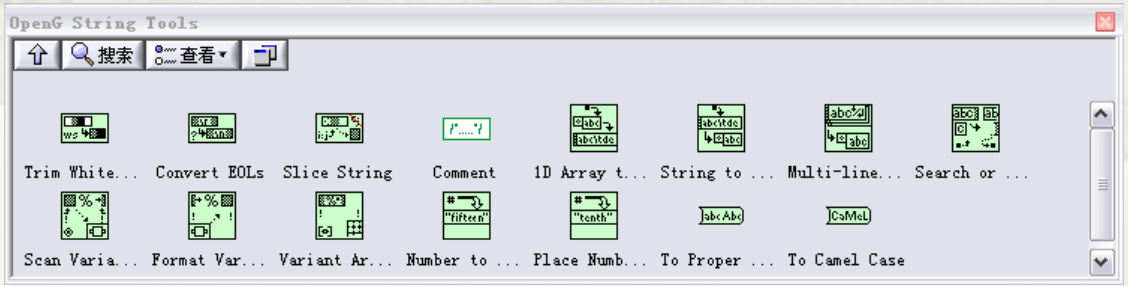
- 格式化值函数和格式化字符串函数
- 格式字符串
- 配置格式化字符串对话框
- 扫描值函数和扫描字符串函数



5.1.6 数组与电子表格字符串
5.1.7附加字符串函数





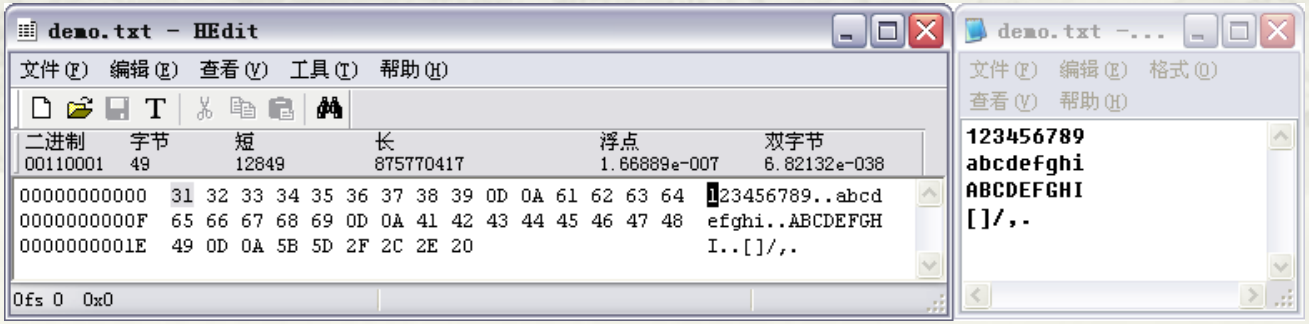


5.2 文件存储

- 5.2.1 文本文件与二进制文件的区别
- 5.2.2 文件常量和通用目录、文件节点函数
- 5.2.3 构造路径的方法
- 5.2.4 文本文件的读写
- 5.2.5 数据记录文件的读写
- 5.2.6 读写二进制文件
- 5.2.7 INI文件的读写
- 5.2.8 注册表的读写
- 5.2.9 TDM文件的读写
- 5.2.10 TDMS文件的读写

5.2.1 文本文件与二进制文件的区别

* 文本文件是以ASCII方式存储的文件，自然读取这个文件采取ASCII解读的方式。文本文件除了可显示字符，比如字母和数字外以及标点符号，还包括不可显示字符，比如空格、回车、换行等等。VI文件用记事本打开出现乱码，就是因为包含了很多不可显示字符。

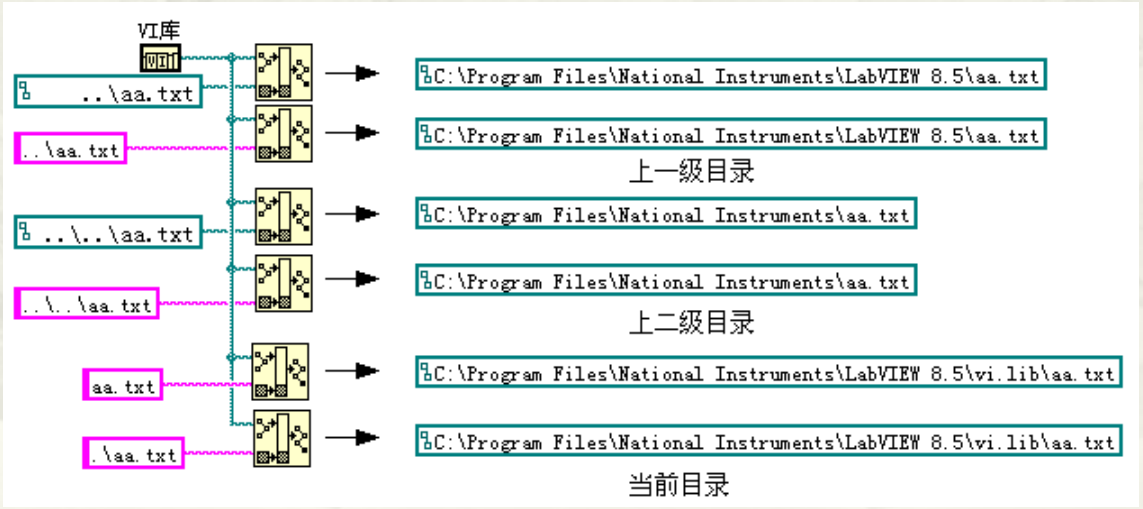


5.2.2 文件常量和通用目录、文件节点函数

* 除了文件常量之外，在高级文件函数选板中还提供了常用的文件操作函数，比如拷贝、删除、移动、创建文件夹和罗列文件夹等。

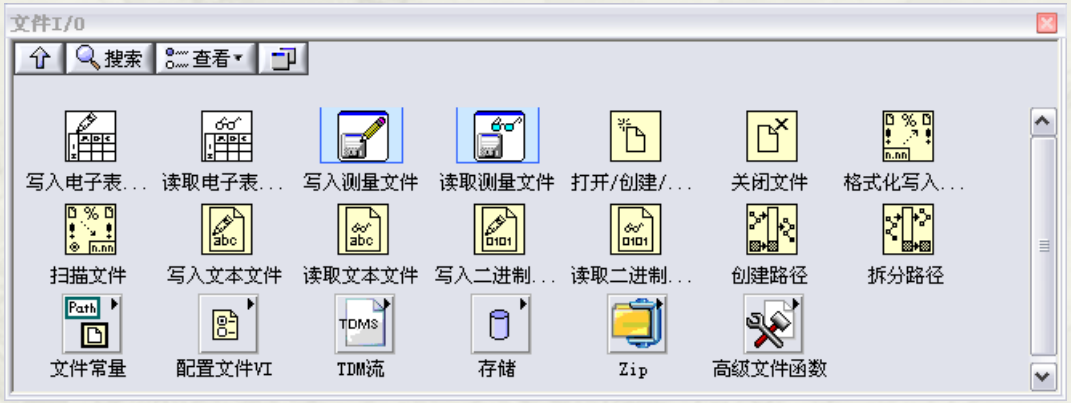
5.2.3 构造路径的方法

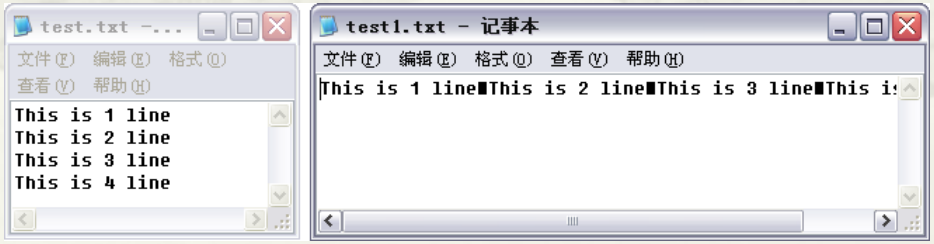
- 创建和拆分路径
- 当前VI路径函数在编辑和运行时的区别
- 解决开发和运行环境路径问题的几种方法



5.2.4 文本文件的读写

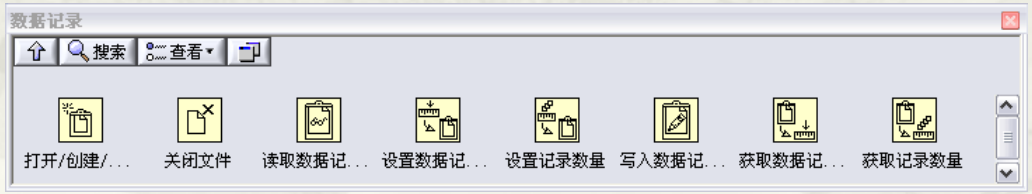
* 操作文件的基本过程为：打开文件、读写文件、关闭文件。读写文本文件和读写二进制文件的节点函数集中在文件I/O函数选板中





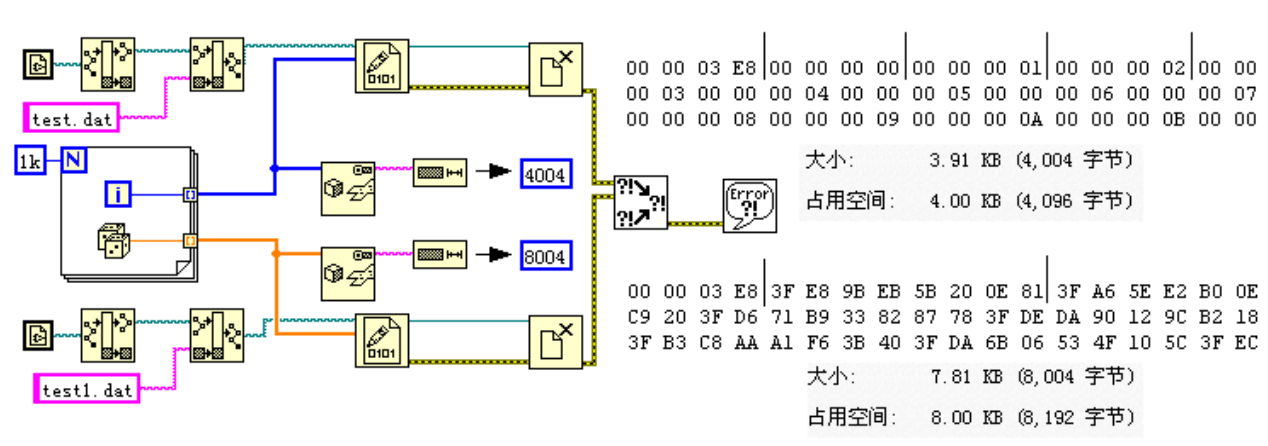
5.2.5 数据记录文件的读写

* 由于二进制文件格式的不确定性，使用非常困难，因此LabVIEW提供了具有格式的二进制文件，即数据记录文件。数据记录文件特别适合于数据块的存储。数据记录文件内部是以记录的方式存储数据的，一个记录就是一个完整的数据块，文件位置定位采用记录号，因此寻址非常快。



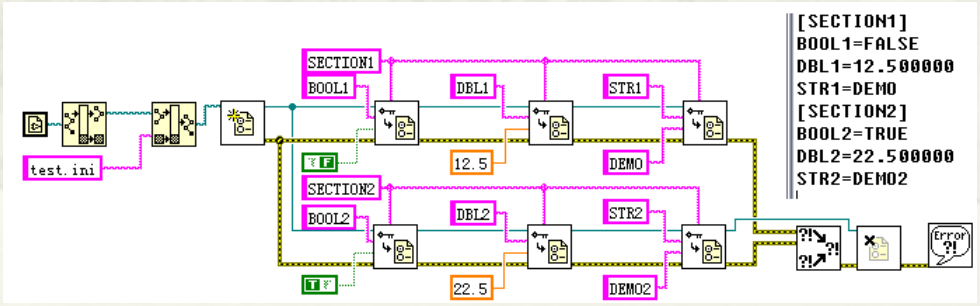
5.2.6 读写二进制文件

* 二进制文件是计算机文件中最常见的文件。它占用空间最小，适合于连续存储大量数据。同时，它的存储格式与数据在内存中的存储格式一致或者类似，很多情况下甚至是内存的映射。因此，无论是存储还是读取都是速度最快的，而且还具有非常高的安全性。如果不知道数据的格式，很难分析出文件的格式。



5.2.7 INI文件的读写

- 在Win95版本以前，Windows操作系统还没有引入注册表的概念，当时Windows是利用INI文件来存储计算机相关配置的。在API里，Windows提供了丰富的API函数来操作INI文件，LabVIEW里也专门有个子类别封装了这些API函数。LabVIEW使用INI文件来存储配置，应用非常广泛，生成执行文档后也自动生成一个INI文件。

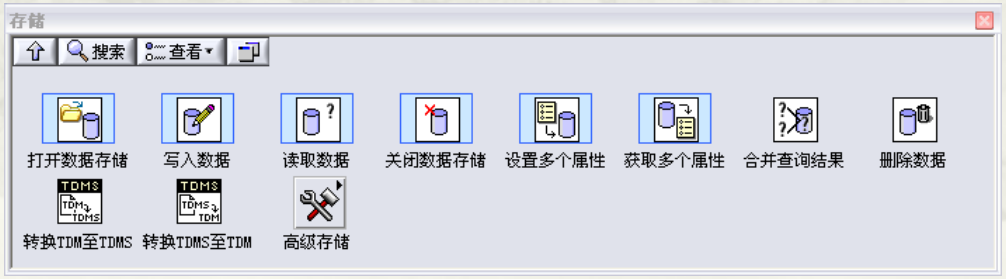


5.2.8 注册表的读写

- Windows注册表的重要性是不言而喻的，Windows绝大部分重要信息都记录在注册表中。读取注册表是Windows编程的常用操作。注册表是特定格式的二进制文件，可以由Windows的注册表编辑工具regedit.exe进行编辑。Windows同时也提供了常用的API函数操作注册表。。

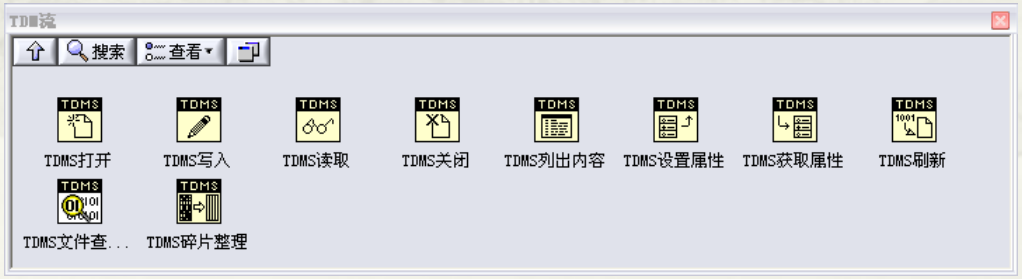
5.2.9 TDM文件的读写

- LabVIEW首先引入了TDM（Technical Data Management）数据管理技术，进而又引入了TDMS流式技术,这使得快速存储查询采集数据管理成为可能。



5.2.10 TDMS文件的读写

- TDMS文件以二进制方式存储数据，所以文件更小，速度更快。因此，它在具备二进制文件优点的同时，又具备关系型数据库的一些优点。据NI公司测试，TDMS格式文件存储速度能达到600MB/S。这样的存储速度能满足绝大多数数据采集系统存储的需要。

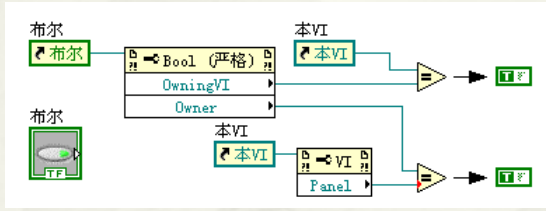


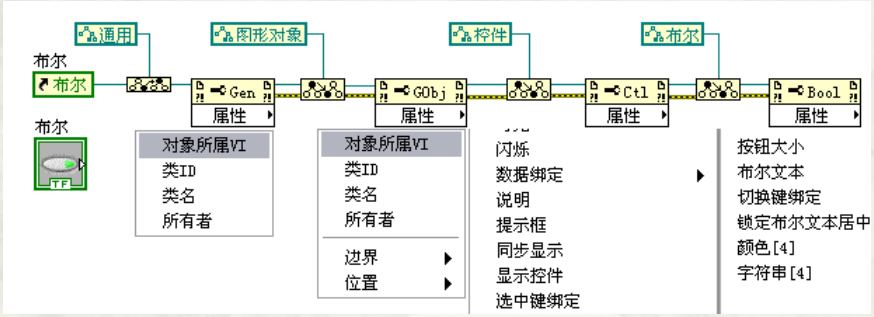
第6章 对象的属性、方法及引用

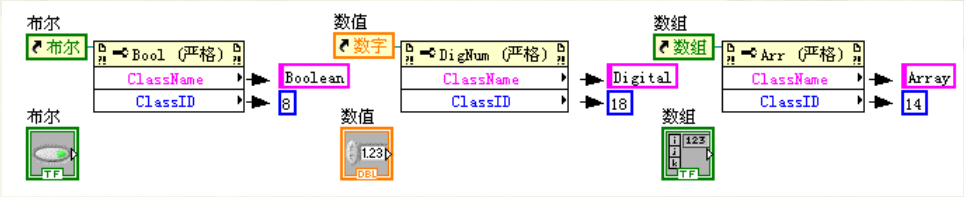
- 6.1 LabVIEW控件对象的层次继承结构
- 6.2 图形对象类的子类
- 6.3 控件类
- 6.4 常用控件的专用属性
- 6.5 引用句柄
- 6.6 VI的属性
- 6.7 常用VI方法
- 6.8 动态调用VI
- 6.9 应用程序的属性和方法
- 6.10 小结

6.1 LabVIEW控件对象的层次继承结构

- 6.1.1 布尔控件类的层次继承结构
- 6.1.2 通用类的属性
- 6.1.3 图形对象类

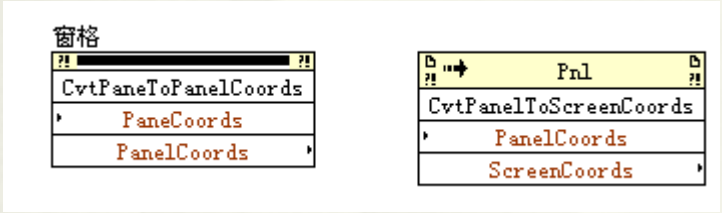


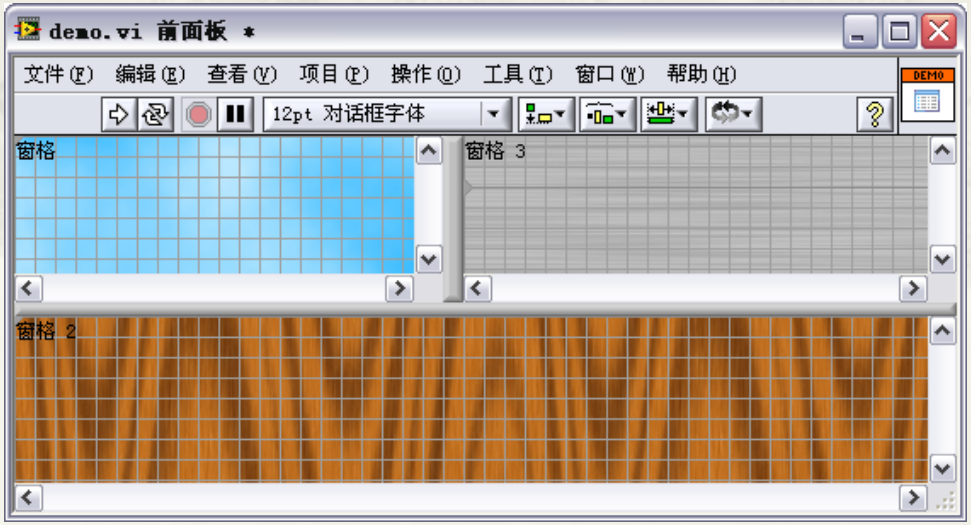


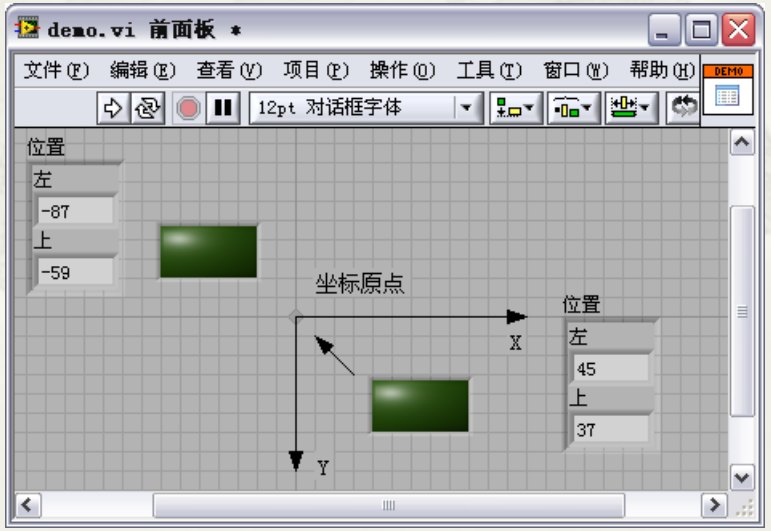


6.2 图形对象类的子类

- 6.2.1前面板类
- 6.2.2窗格类和分隔栏类
- 6.2.3 LabVIEW的坐标映射
- 6.2.4装饰类

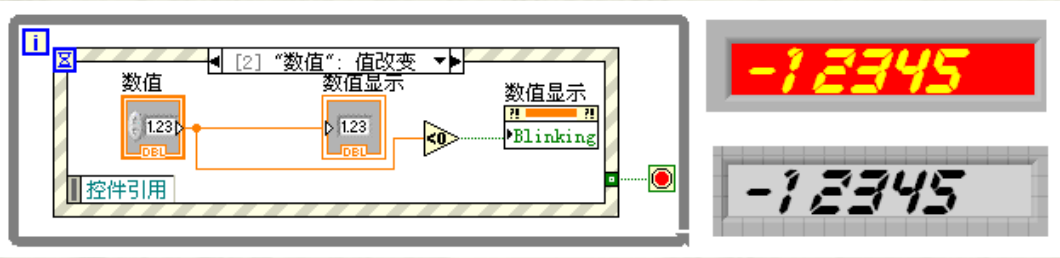






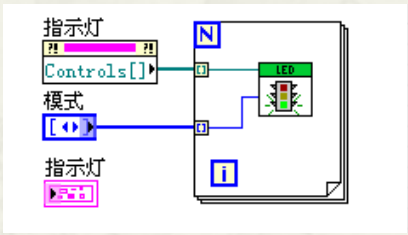
6.3 控件类

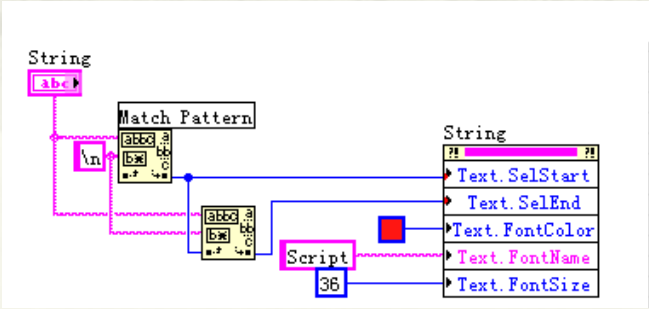
- 6.3.1控件类的常用属性
- 6.3.2控件类的常用方法
- 6.3.3数值控件类

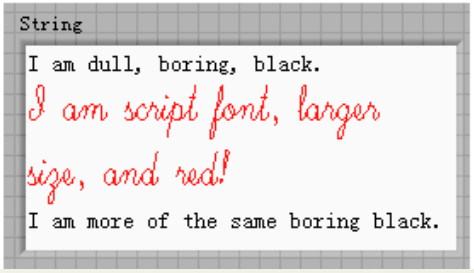


6.4 常用控件的专用属性

- 6.4.1 布尔控件的专用属性
- 6.4.2 枚举和下拉列表控件的专用属性
- 6.4.3 字符串、路径控件和组合框控件的专用属性
- 6.4.4 数组的属性和方法
- 6.4.5 簇的属性及方法

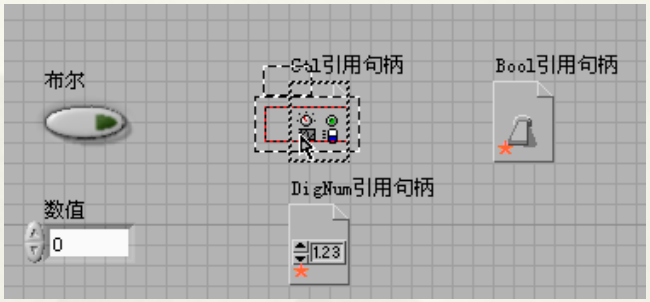


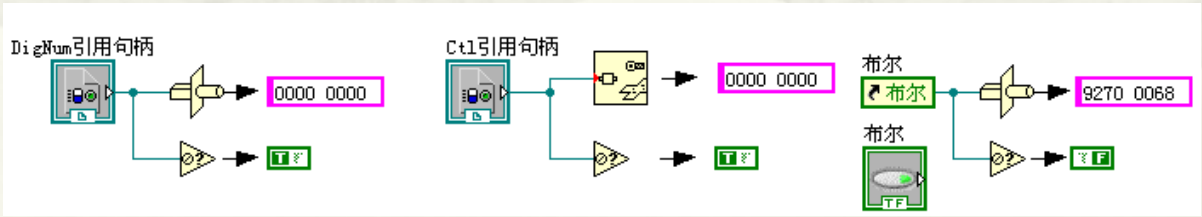




6.5 引用句柄

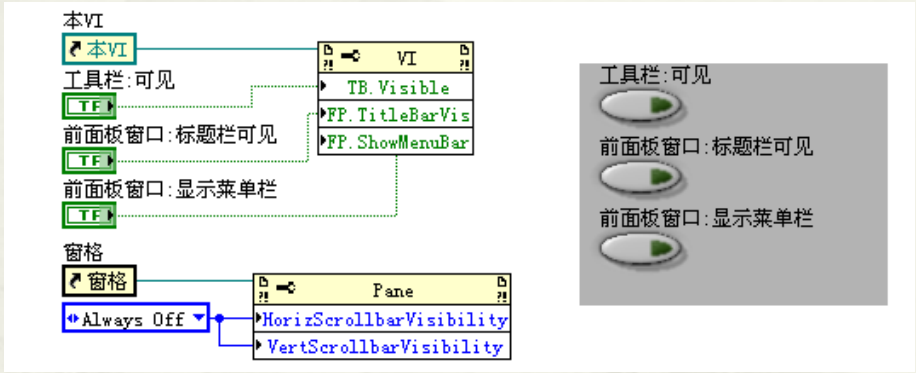
* 引用句柄与类的实例化
* 创建通用引用的方法

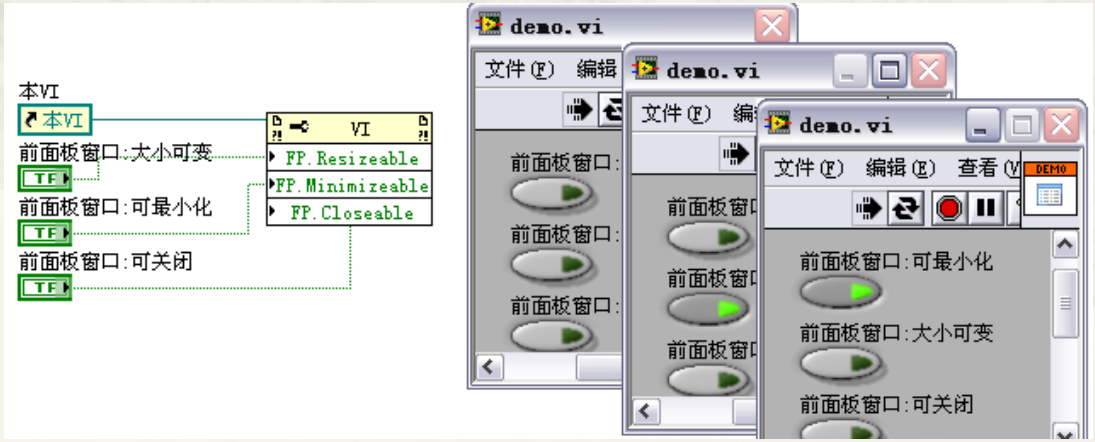


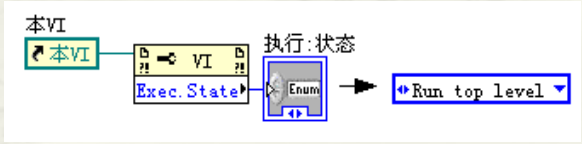


6.6 VI的属性

* 6.6.1 如何取得VI的引用
* 6.6.2 常用VI属性
* 6,6.3 VI前面板属性

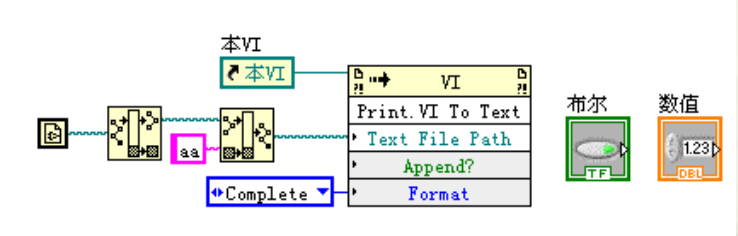


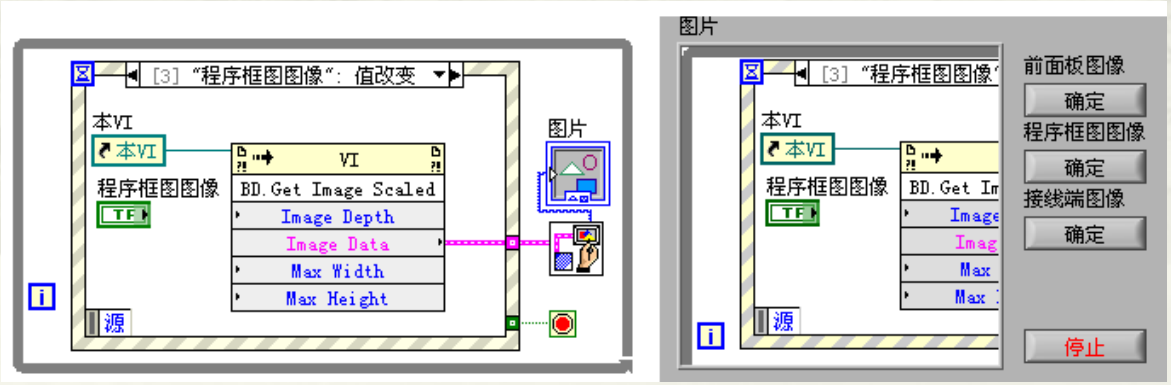




6.7 常用VI方法

- 6.7.1 获取前面板、程序框图和VI图标的图像
- 6.7.2 前面板的运行位置控制
- 6.7.3 打印控制
- 6.7.4 默认值方法





6.8 动态调用VI

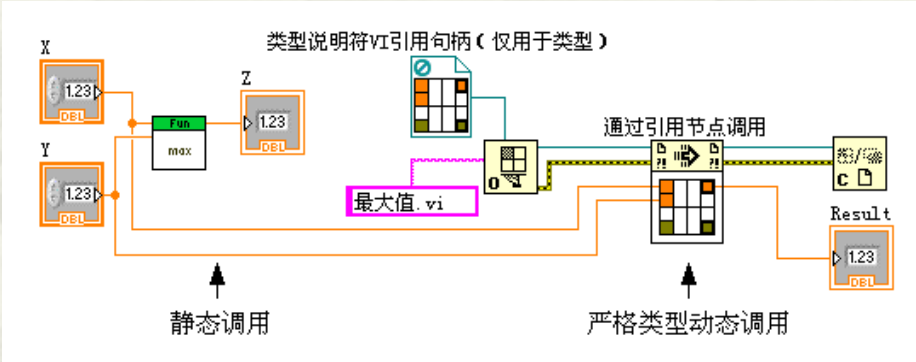
- 6.8.1 静态调用和动态调用的优劣
- 6.8.2 严格类型VI的动态调用
- 6.8.3 一般类型VI的动态调用
- 6.8.4 创建闪屏
- 6.8.5 创建后台运行程序
- 6.8.6 创建向导程序
- 6.8.7 动态调用VI之间的数据交换

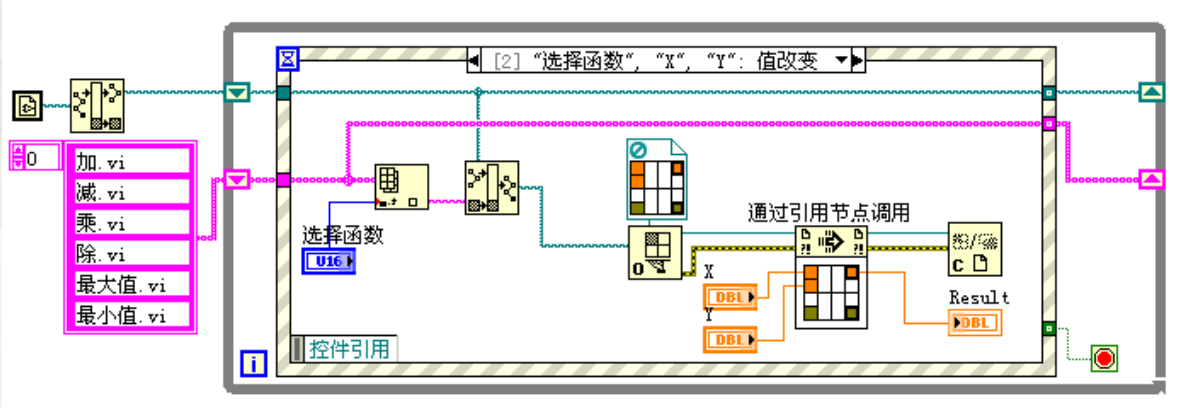
6.8.1静态调用和动态调用的优劣

- 动态加载VI、运行VI、关闭VI，有利于减少内存的使用。
- 动态控制VI的特性，如位置、外观等。
- 各VI之间灵活的数据交换，特别适合于不连续数据交换，比如监控。
- 主VI和子VI的并行运行。静态调用子VI时，主VI必须等待子VI运行完毕后才继续运行。
- 动态调用可以实现网络VI调用，即通过计算机网络，远程调用其它计算机上的VI。
- 强大的插件功能。通过动态调用，可以实现增功能。比较典型的是滤波器的使用。只要输入、输出参数相同，原有程序不需任何改动，就可以增加新的滤波器

6.8.2 严格类型VI的动态调用

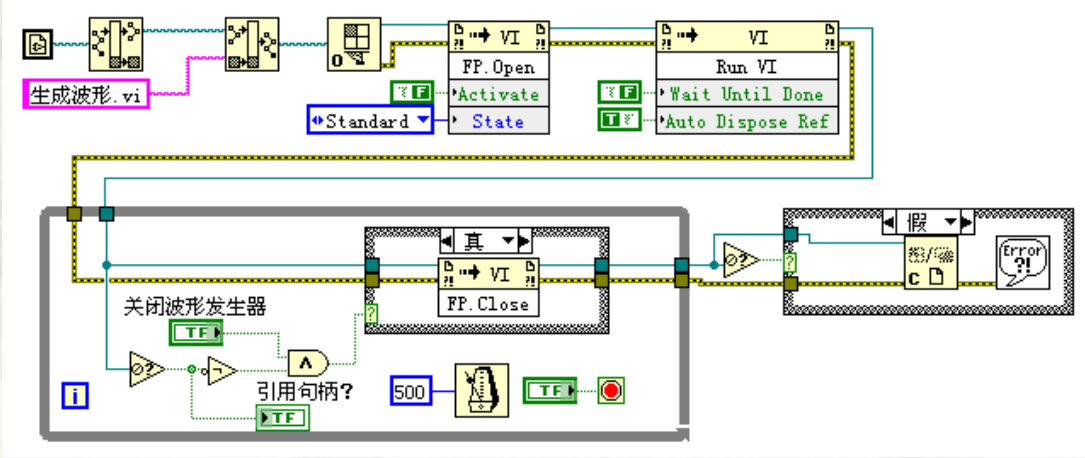
- 严格类型VI的含义
- 严格类型VI应用举例





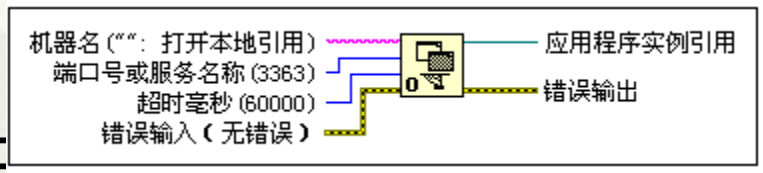
6.8.3 一般类型VI的动态调用

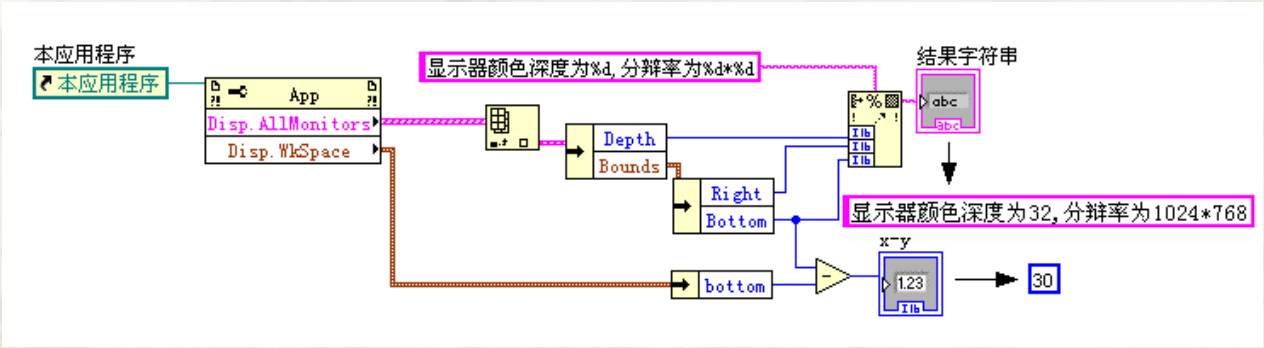
- 获取VI引用句柄
- 通过VI引用句柄动态调用VI
- 动态调用的过程分析

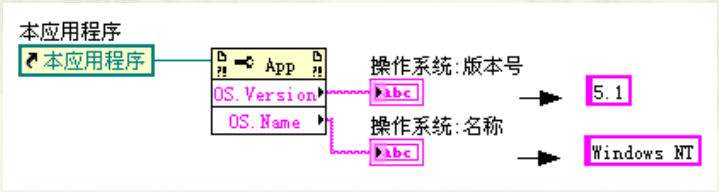


6.9 应用程序的属性和方法

* 6.9.1 获取应用程序句柄
* 6.9.2 常用应用程序的属性





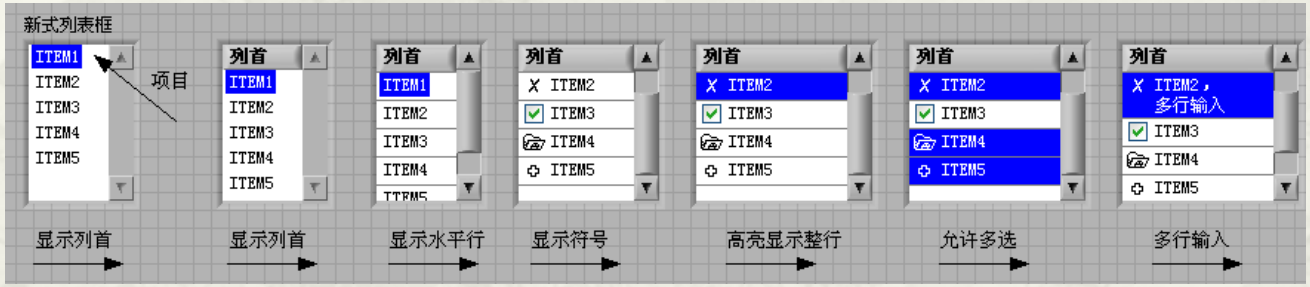


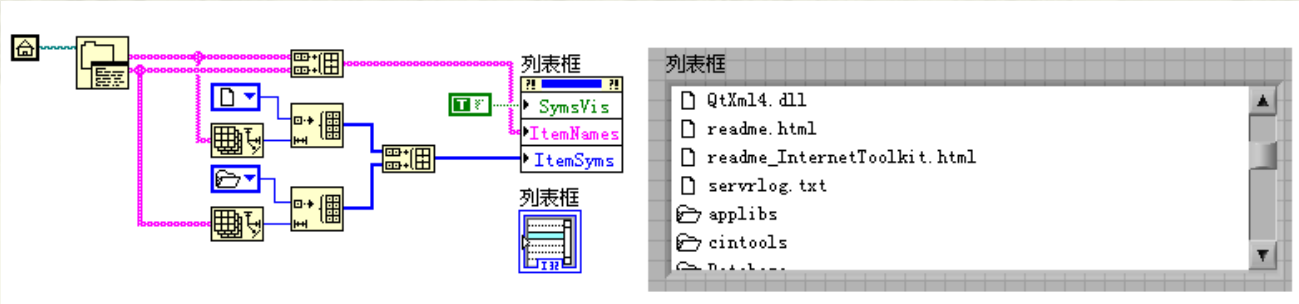
第7章 高级控件的运用

* 7.1 列表框
* 7.2 多列列表框
* 7.3 表格
* 7.4 树形控件
* 7.5 波形图表
* 7.6 波形图
* 7.7 XY图
* 7.8 强度图表和强度图
* 7.9 数字数据、数字波形与数字波形图
* 7.10图片控件
* 7.11小结

7.1 列表框

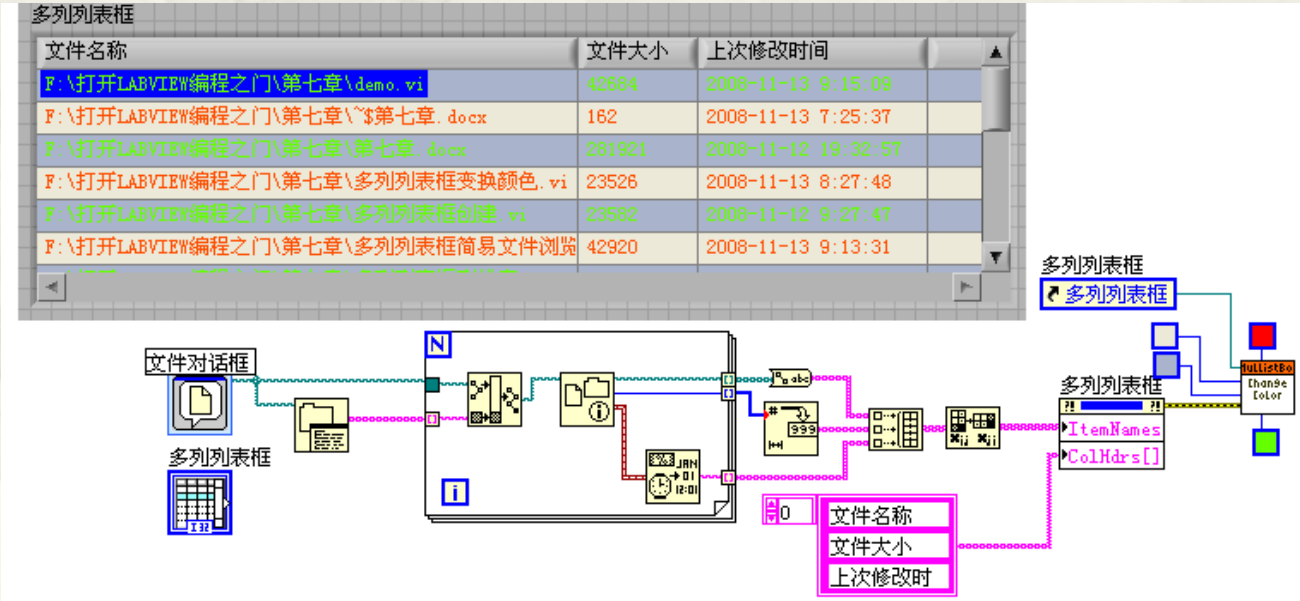
- 7.1.1 列表框的创建及显示风格
- 7.1.2 列表框常用属性、方法与事件
- 7.1.3 列表框的应用举例





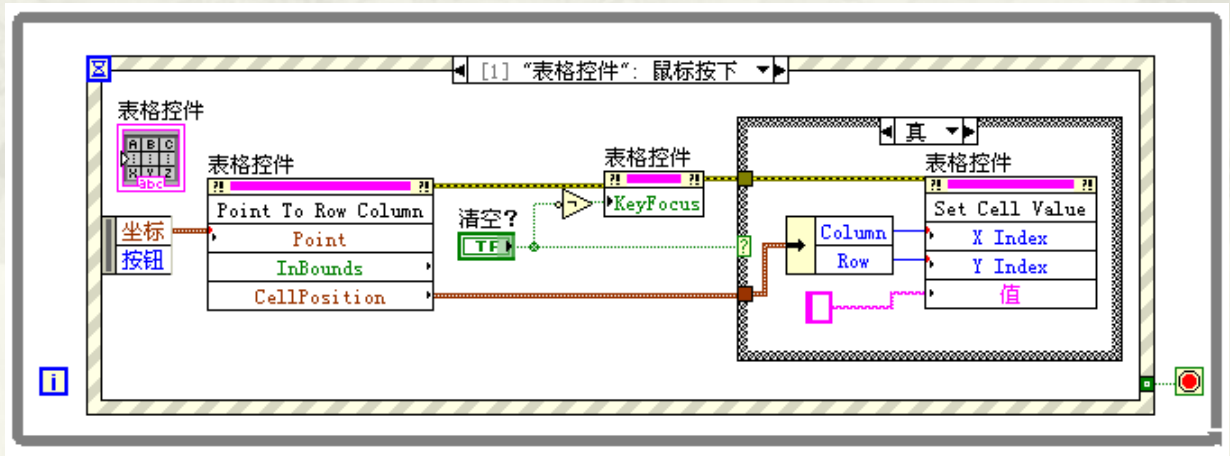
7.2 多列列表框

- 7.2.1 显示多列条目并排序
- 7.2.2 多列列表框的特效制作



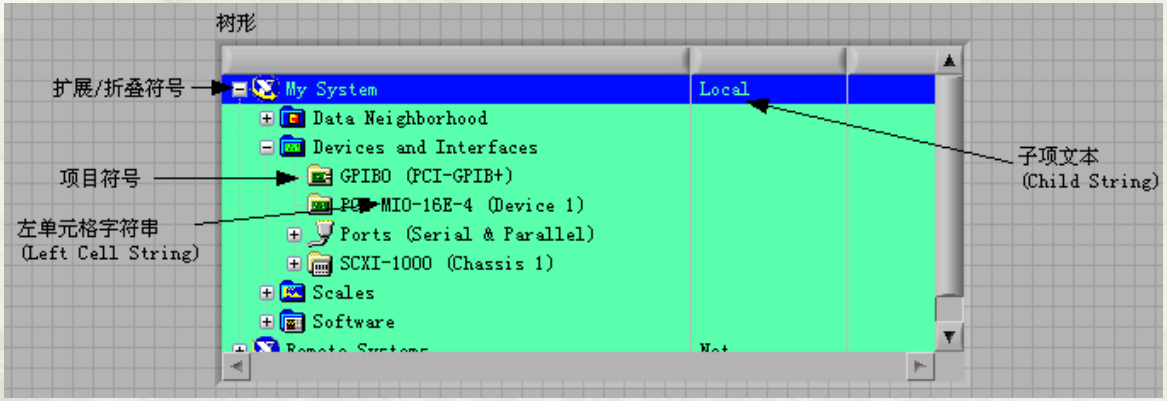
7.3 表格

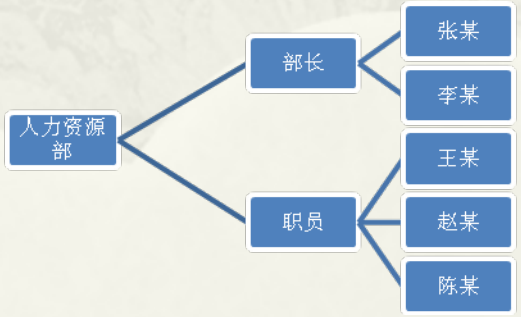
- 7.3.1 表格的常用属性和方法
- 7.3.2 表格的应用举例



7.4 树形控件

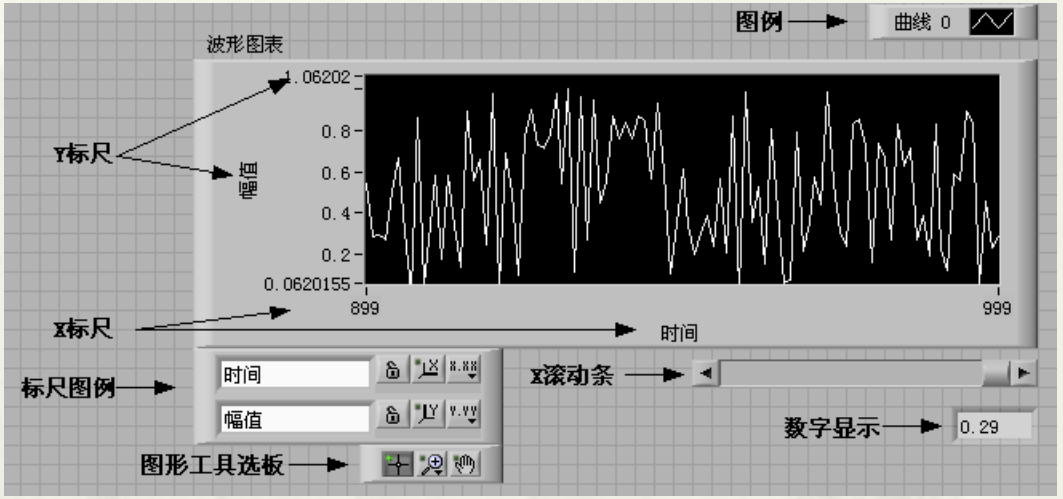
- 7.4.1 树形控件的创建与静态编辑
- 7.4.2 树形控件常用属性、方法和事件
- 7.4.3 树形控件高级应用举例

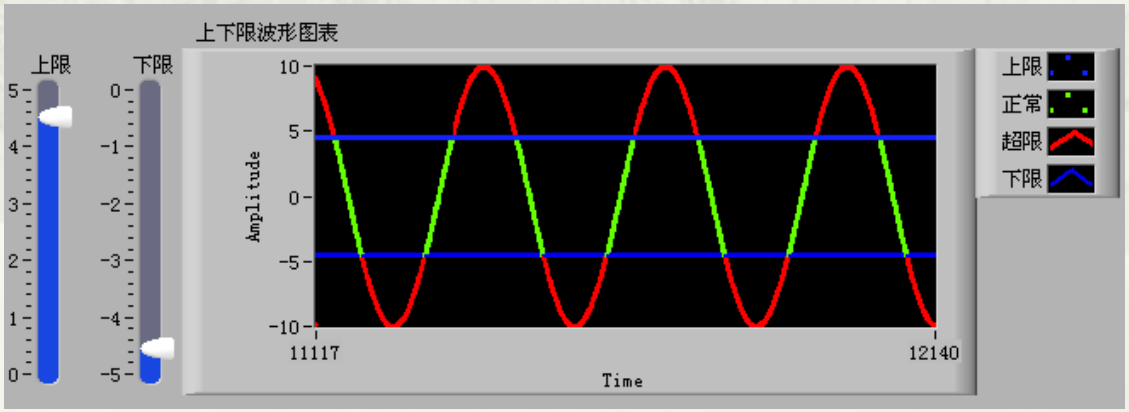




7.5 波形图表

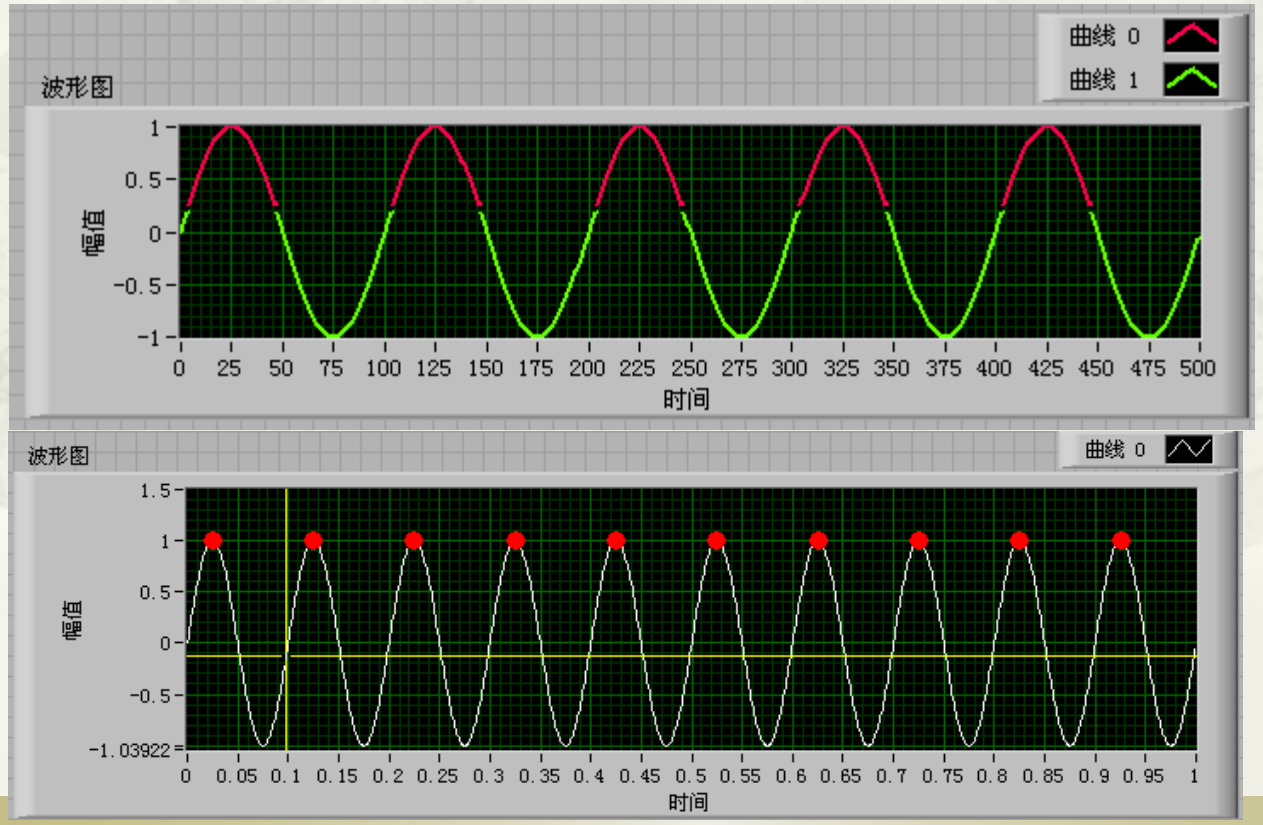
- 7.5.1 波形图表的组成要件
- 7.5.2 波形图表的输入类型
- 7.5.3 波形图表常用属性
- 7.5.4 波形图表应用举例





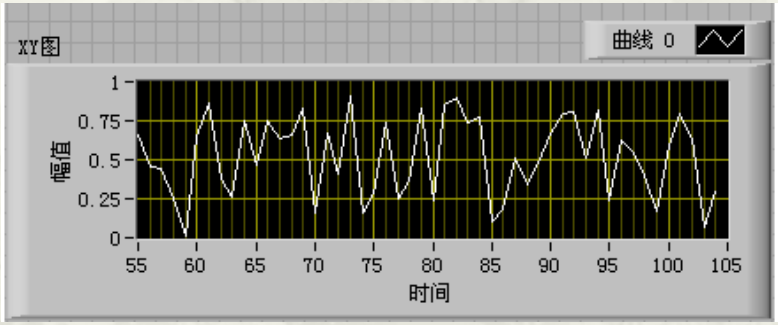
7.6 波形图

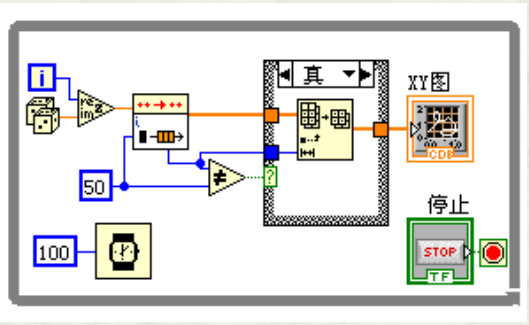
- 7.6.1 波形图控件的创建和组成要件
- 7.6.2 波形图控件的输入类型
- 7.6.3 波形图控件的专用属性
- 7.6.4 波形图控件的高级应用举例

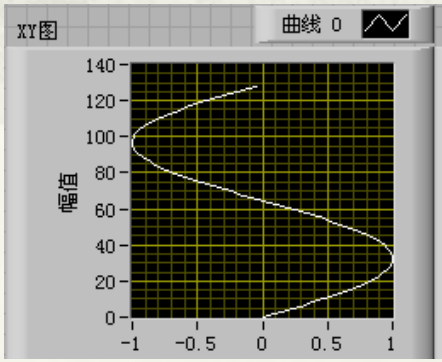


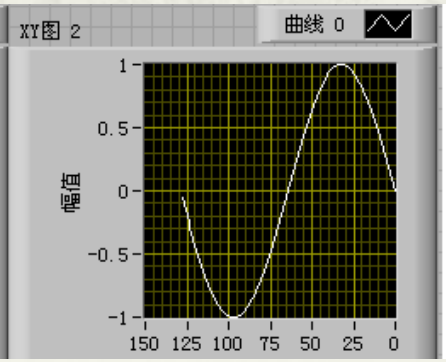
7.7 XY图

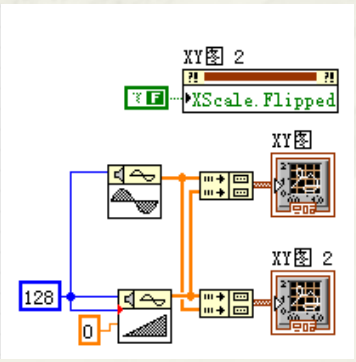
- 7.7.1 XY图的输入数据类型
- 7.7.2 XY图的高级应用





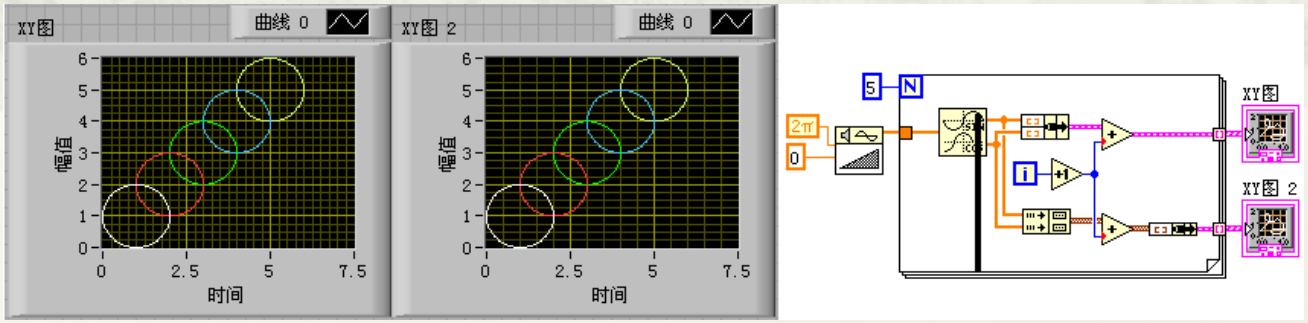






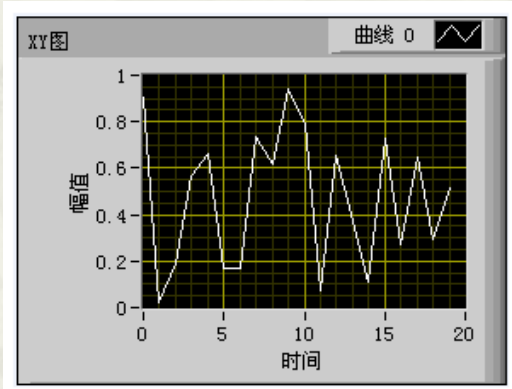
7.7.1 XY图的输入数据类型

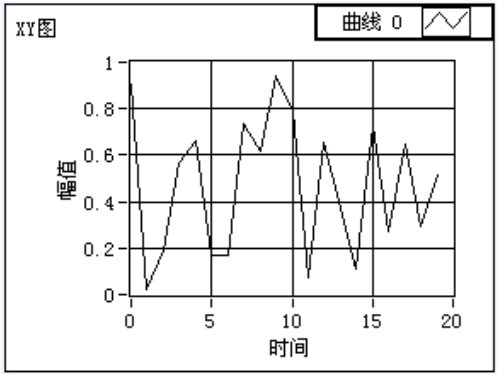
* 复数数组输入
* 点簇构成的数组输入
* 一维数组捆绑输入
* 使用系统时间作为X轴
* 复数簇数组显示多条曲线
* 簇数组显示多条曲线
* XY图显示两条曲线的特殊方法

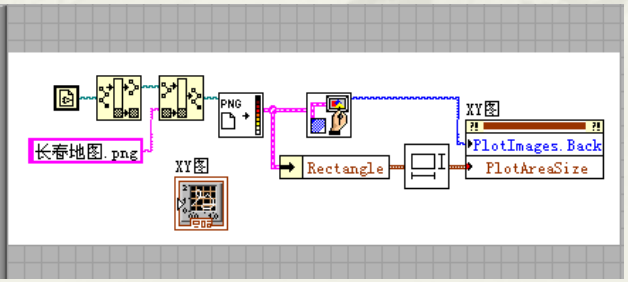


7.7.2 XY图的高级应用

* XY图表实现波形图表显示效果
* 曲线的纵向显示
* 动态指定标尺
* 导出图像
* XY图显示背景图片

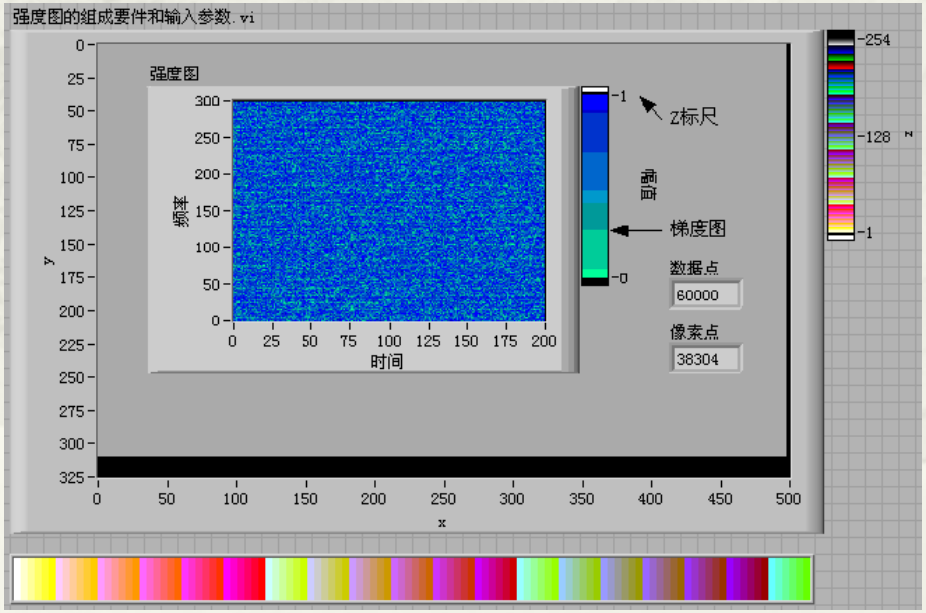






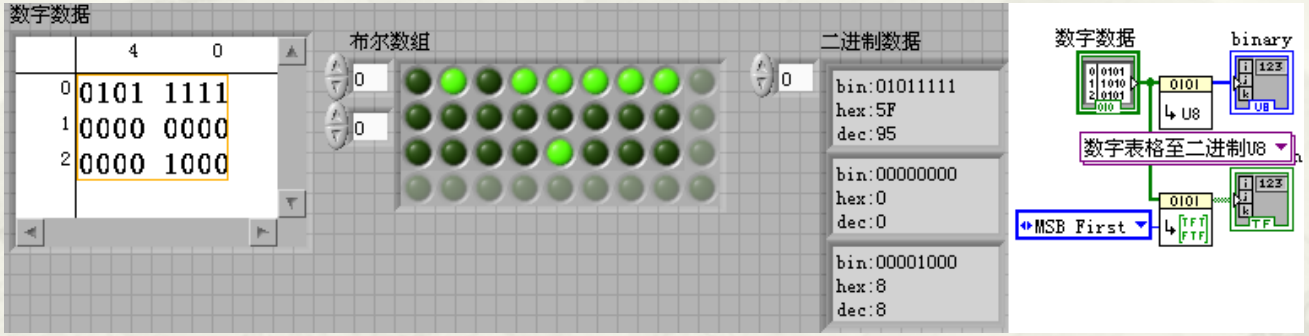
7.8 强度图表和强度图

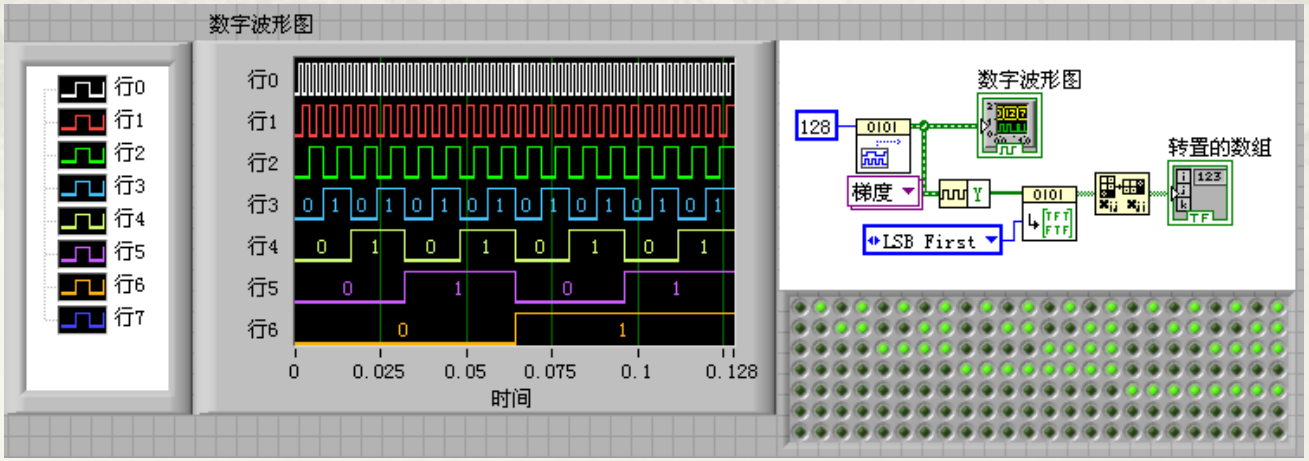
* 强度图不同于波形图的最大区别，是强度图由三个坐标轴组成X、Y、Z组成，默认标签为时间、频率和幅值。其中X、Y坐标确定位置，而Z坐标表示当前位置的值，这实际上就是二维数组的表示方式，所以强度图的输入参数是二维数组。



7.9 数字数据、数字波形与数字波形图

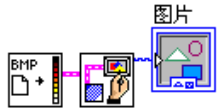
- 7.9.1 数字数据
- 7.9.2 数字波形数据和数字波形图

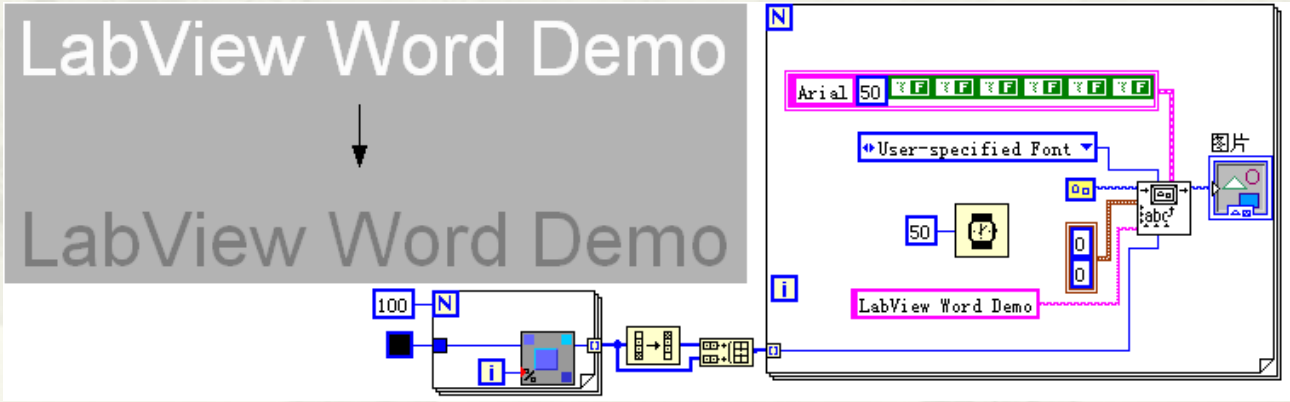




7.10 图片控件

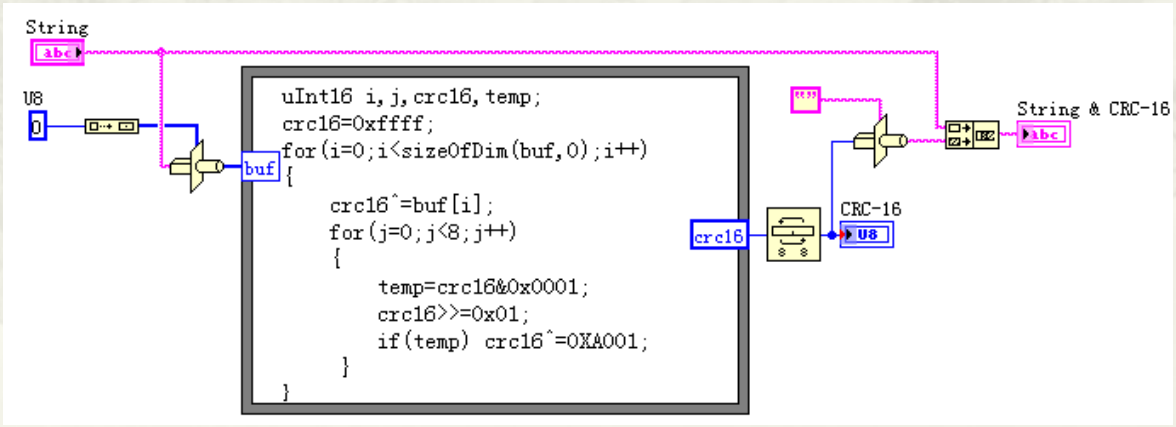
- 7.10.1 利用图片控件显示图片
- 7.10.2 常用绘图操作函数
- 7.10.3 图片控件的高级应用





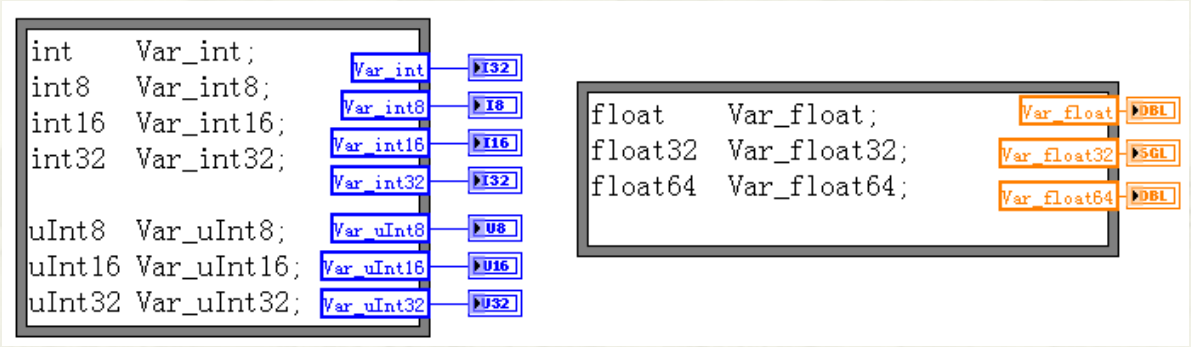
第8章 文本式编程与外部程序接口

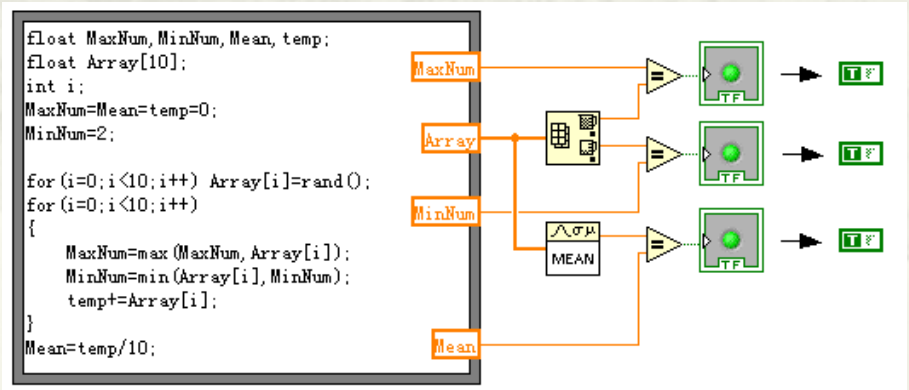
- 8.1 公式节点
- 8.2 调用库函数
- 8.3 CIN
- 8.4 系统命令
- 8.5 剪切板
- 8.6 DDE库
- 8.7 ActiveX控件与ActiveX文档
- 8.8 .NET技术
- 8.9 小结



8.1 公式节点

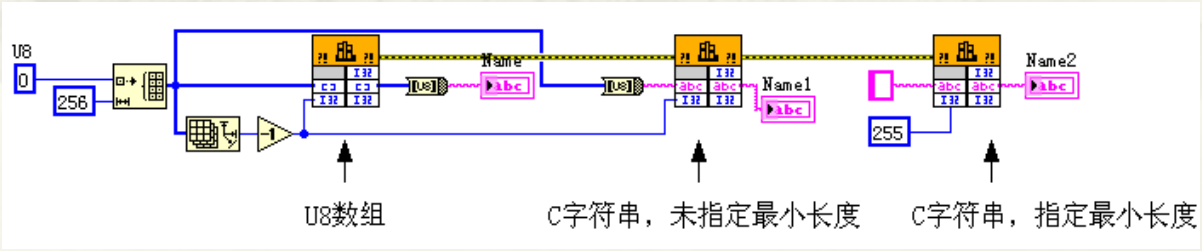
- 8.1.1 公式节点的数据类型、语法与控制结构
- 8.1.2 公式节点的应用举例





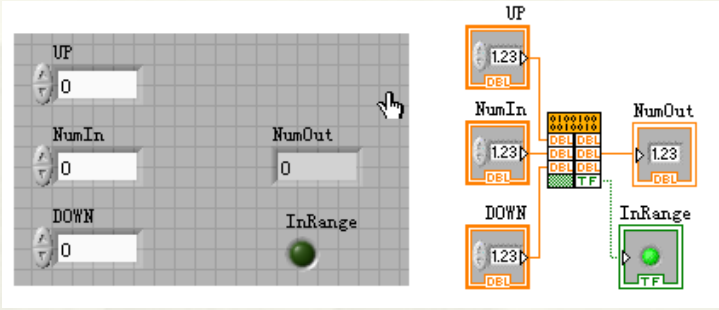
8.2 调用库函数

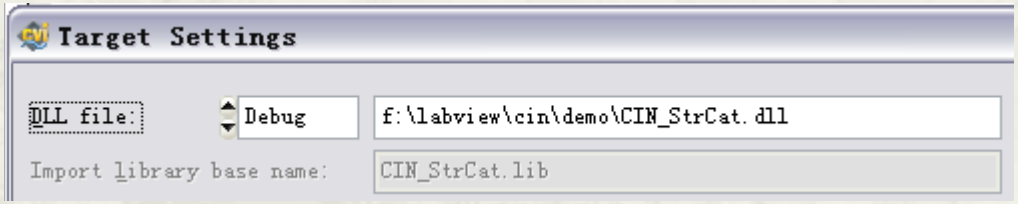
- 8.2.1 动态链接库与API函数
- 8.2.2 如何调用DLL函数
- 8.2.3 常用API函数的调用
- 8.2.4 LabVIEW调用DLL的局限性

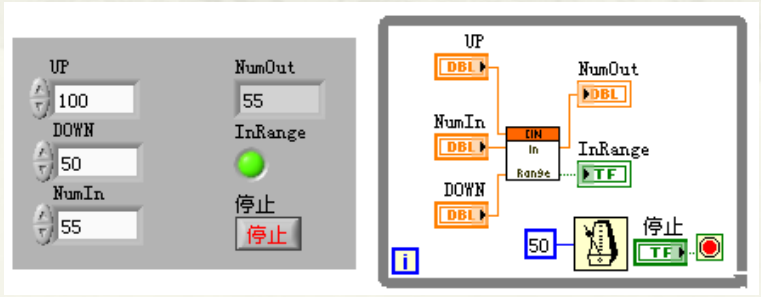


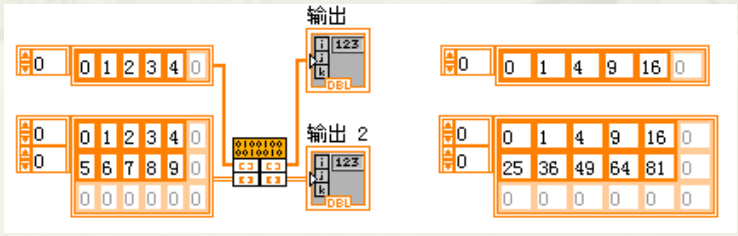
8.3 CIN

* 8.3.1 CIN创建的一般过程
* 8.3.2 CIN的数据类型和常用函数
* 8.3.3 CIN与内存管理器
* 8.3.4 CIN的运行过程和数据共享



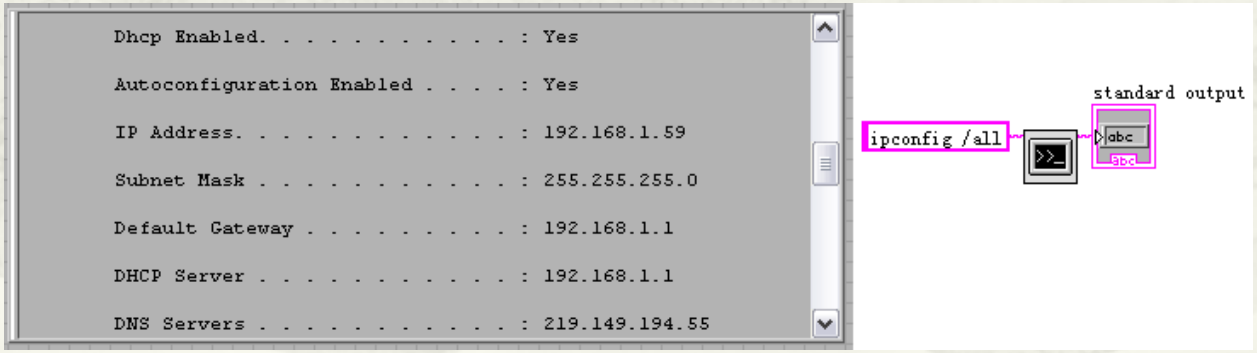






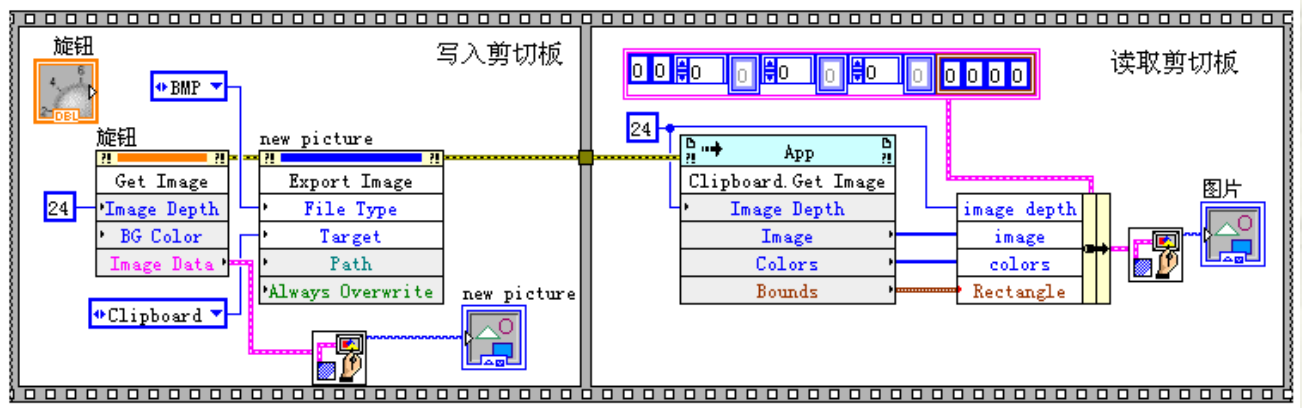
8.4 系统命令

* 调用内部DOS命令
* 调用外部DOS命令
* 调用一般执行文件
* 复杂参数的命令行输入



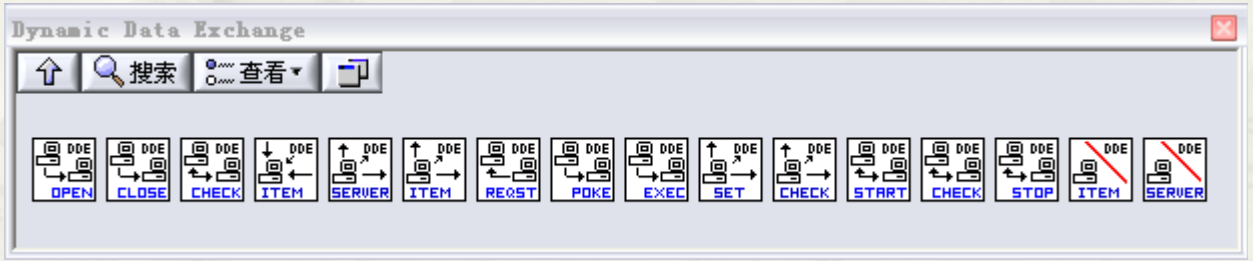
8.5 剪切板

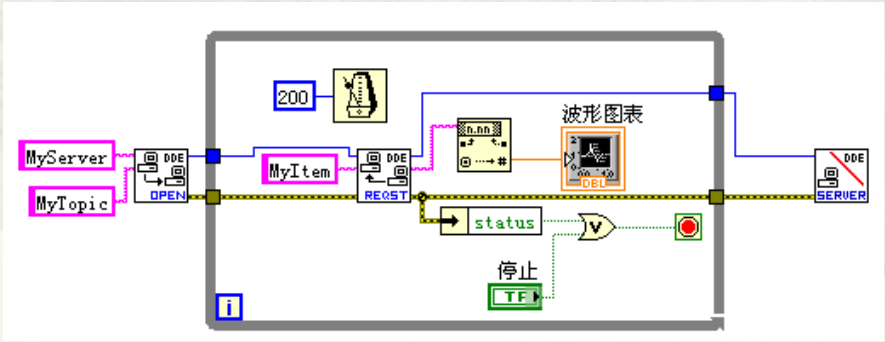
- 剪贴板内置于Windows中，使用系统的内部资源RAM或虚拟内存来临时保存剪切和复制的信息。剪切或复制时保存在剪贴板上的信息，只有在再次剪贴或复制其他信息、断电或有意地清除时，才可能更新或清除其内容。即剪切或复制一次，就可以粘贴多次。



8.6 DDE

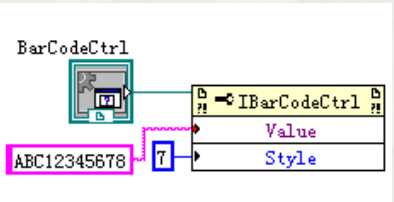
- 8.6.1 DDE概述
- 8.6.2 LabVIEW中DDE的常用方法

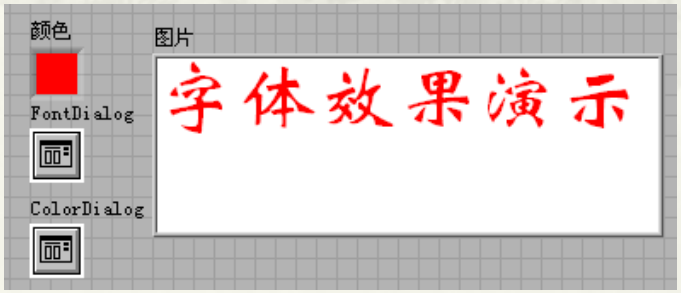




8.7 ActiveX控件与ActiveX文档

- 8.7.1 ActiveX的基本概念
- 8.7.2 ActiveX控件的调用过程
- 8.7.3 ActiveX应用实例
- 8.7.4 ActiveX自动化服务器





$$Y = AX^2 + BX + C$$

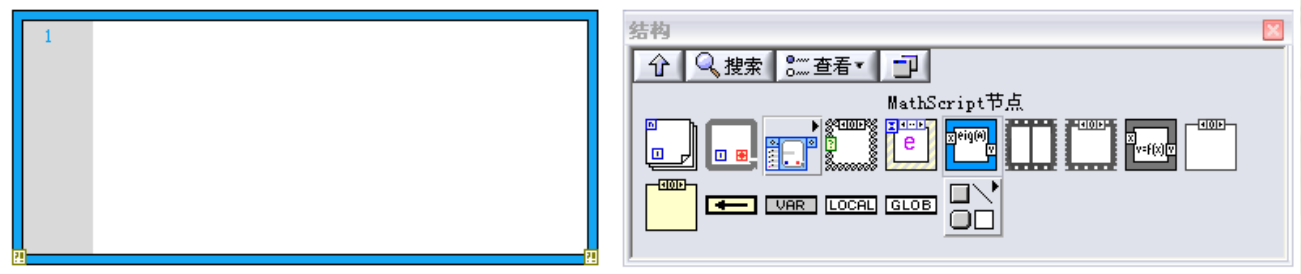
$$u = \frac{1}{N}\sum_{i=0}^{N-1} X_i$$

8.8 .NET技术

- 8.8.1 .NET控件
- 8.8.2 NET服务
- 8.8.3 利用.NET创建托盘程序

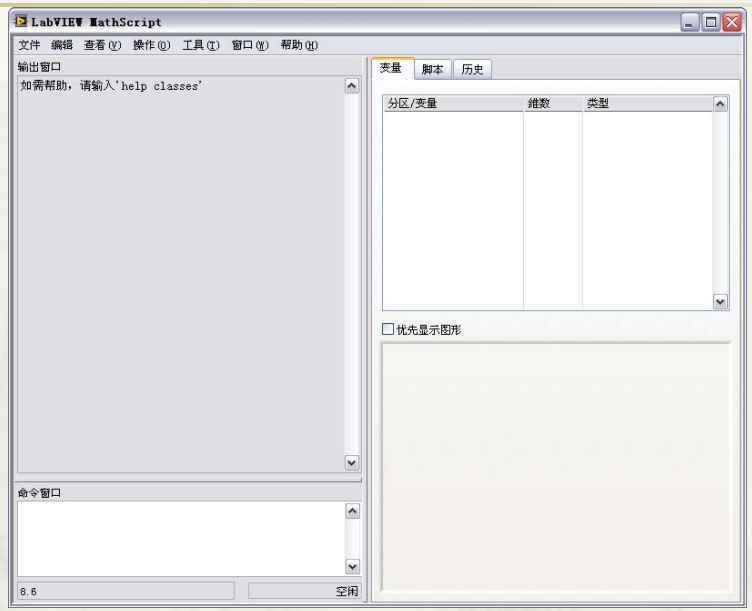
第9章 MathScript

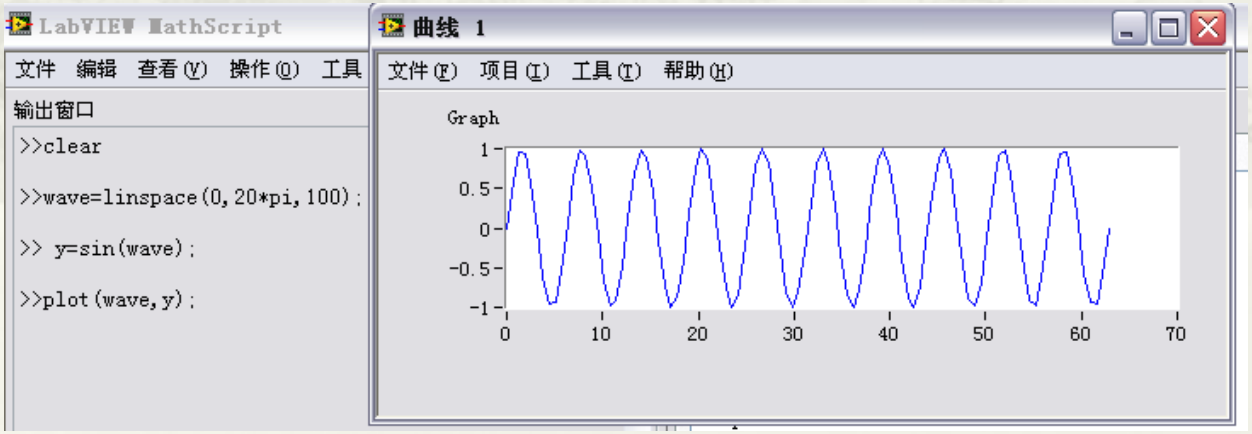
- 9.1 如何使用MathScript
- 9.2 MathScript常用命令
- 9.3 MathScript基础知识
- 9.4 程序控制结构与函数
- 9.5 数据统计和数据插值拟合
- 9.6 多项式、优化、积分和微分
- 9.7 数据的图形显示
- 9.8 小结



9.1 如何使用MathScript

- 9.1.1 使用MathScript节点
- 9.1.2 使用MathScript交互窗口





9.2 MathScript常用命令

- MathScript是文本式编程语言，每输入一个函数立即执行，是一种基于命令的输入方式。其中的一部分是系统命令，类似于DOS命令，还有一部分是控制MathScript交互窗口的命令。在学习MathScript编程之前，熟悉这些命令是非常重要的。下面详细介绍下各种常用命令。
- **help命令**。MathScript众多函数分成了许多分类。如果无法牢记所有函数的用法，使用help命令可以直接打开在该函数或者分类的帮助文档上。
- 例如，使用help advanced命令将显示advanced分类函数帮助。使用help sin命令将显示正弦函数的帮助。
- **cd和dir命令**。类似于DOS命令。cd命令显示或者改变文件夹，dir命令显示当前文件夹中的内容，可以使用通配符。
- 例如，使用cd命令将显示当前文件夹，使用cd ‘..’命令将进入到上一级文件夹，使用cd ‘f:\abc’命令将改变文件夹到 “f:\abc”；使用Dir命令将显示所有文件及文件夹，使用dir ‘*.vi’命令将显示所有vi类型的文件。
- **disp和display命令**。使用Disp命令将显示变量的内容，不含名称。使用display命令将显示变量的内容和名称。

9.3 MathScript基础知识

- 9.3.1 创建向量和矩阵的基本方法
- 9.3.2 矩阵的基本运算
- 9.3.3 标准矩阵
- 9.3.4 矩阵元素的插入、替换、删除和提取
- 9.3.5 矩阵元素的排序和搜索特征值
- 9.3.6 矩阵常用变换函数
- 9.3.7 矩阵中元素的数据类型及其转换
- 9.3.8 关系运算、逻辑运算和位操作
- 9.3.9 集合函数
- 9.3.10 时间、日期和计时函数

```
>>x=rand(3,4)
>>sortrows(x)
>>sortrows(x,1)
>>sortrows(x,3)
```

```
>>x=rand(3,4)
>>max(x)
>>min(x)
>>max(x(:))
>>min(x(:))
>>mean(x)
```

```
>>eye(3)
ans =
     1     0     0
     0     1     0
     0     0     1
```

```
>>x=rand(3,4)
>>k=find(x>0.5)
>>x(k)
```

```
>>A=[1 2 3;4 5 6;7 8 9]
>>A(2,2)=8
>>A(2,5)=10
```

9.4 程序控制结构与函数

- 9.4.1 For循环和While循环
- 9.4.2 if条件结构和switch分支条件结构
- 9.4.3 函数和脚本文件

```
for variable = expression
    statement1,
    ....
    statementn,
end
```

```
while expression
    statement1
    ....
    statementn
end
```

```
if expression
    statement, ... , statement
elseif expression
    statement, ... , statement
else
    statement, ... , statement
end
```

```
switch expression
case expression
statement, ... , statement
...
otherwise
```

9.5 数据统计和数据插值拟合

- 9.5.1常用数据统计函数
- 9.5.2数据插值

```
Data=[ 0.80708      0.28597      0.86297
0.54432;
    0.6114       0.12997      0.024445
0.61139;
    。 。 。 。 。 。 。 。 。 。 。 。 。 。 。 。
    0.44786      0.08477      0.63748
0.15612;
    0.25372      0.39714      0.5067
0.72956;
    0.73856      0.21347      0.85222
0.15819];
```

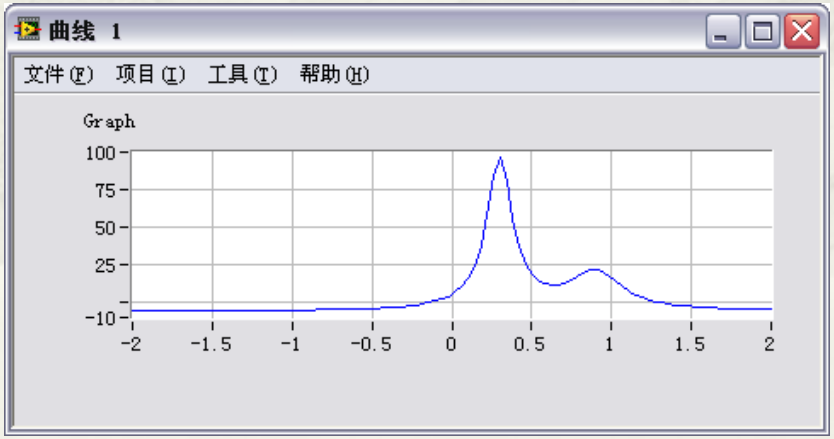
```
>>x=1:5;
>>y=rand(1,5);
>>z=x+y;
>>detrend(z)
ans =
          0.09273    -0.28715     0.35145
-0.21239     0.05535
>>z
z =
          1.805      2.3538      3.9211
4.286      5.4824
```

9.6多项式、优化、积分和微分

- 9.6.1 多项式
- 9.6.2 优化
- 9.6.3 积分和微分

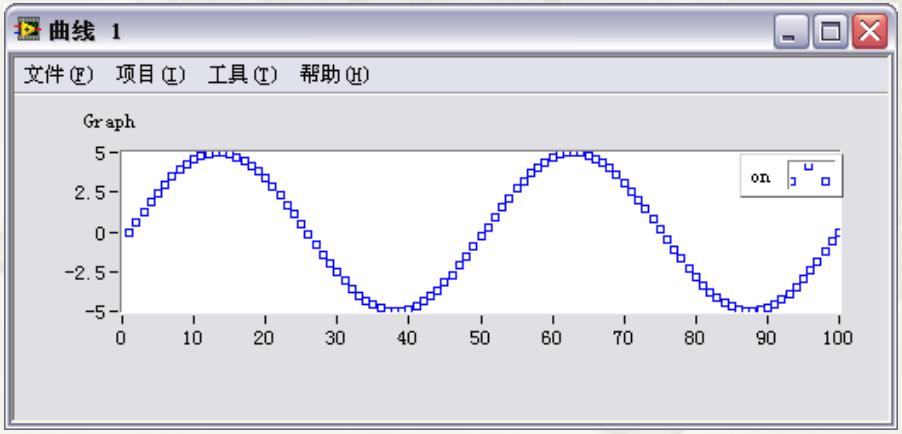
```
>>a=[1 2 3 4];b=[2 3];
>>y=conv(a,b)
>>y1=deconv(y,a)
```

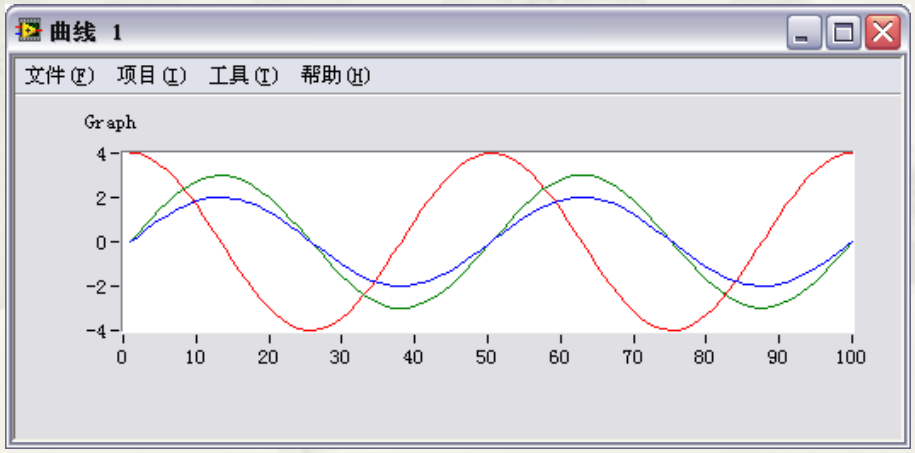
```
>>x=linspace(0,1,10000);
>>y=sin(x);
>>trapz(x,y)
```



9.7数据的图形显示

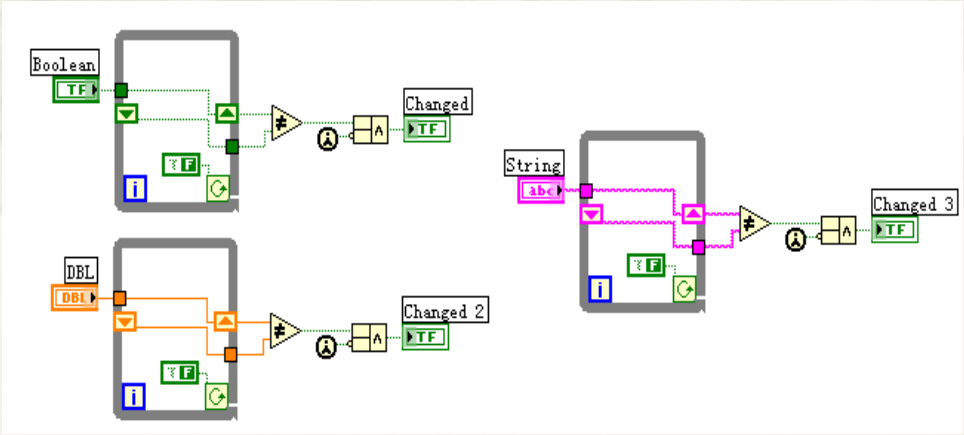
- 9.7.1 窗口类属性与常用窗口操作函数
- 9.7.2 绘图区域属性
- 9.7.3 线对象和文本对象的属性和常用函数
- 9.7.4 基本绘图函数





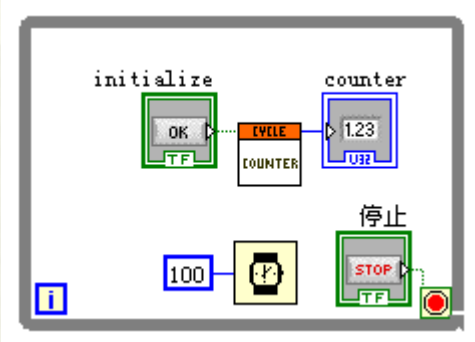
第10章 基于组件的程序结构

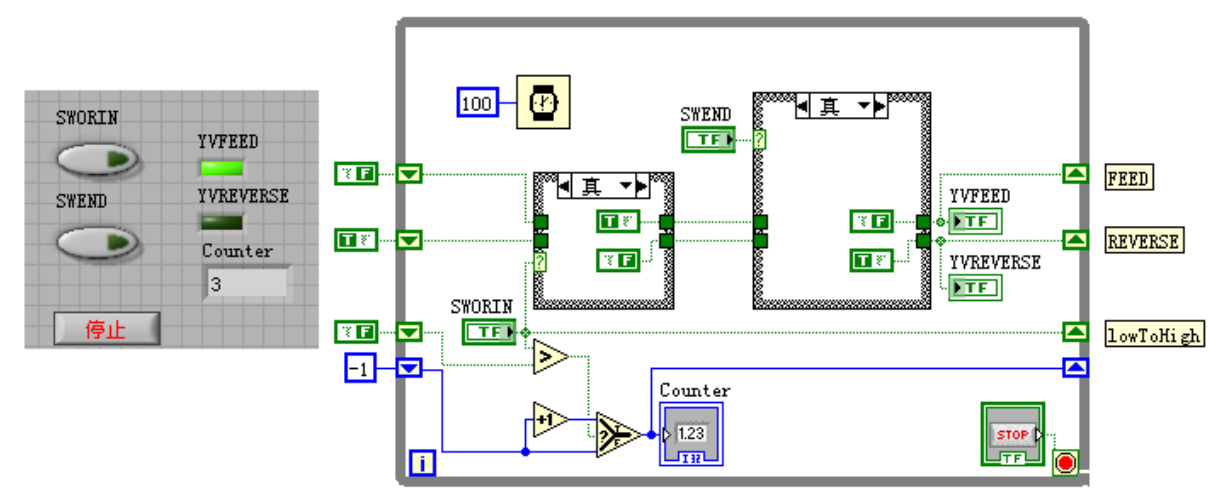
- 10.1 数据的封装与隔离
- 10.2 Action Engine
- 10.3 用户事件与动态注册事件
- 10.4 堆栈、数据缓冲区
- 10.5 同步控制技术
- 10.6 项目管理器
- 10.7 面向对象编程
- 10.8 小结



10.1 数据的封装与隔离

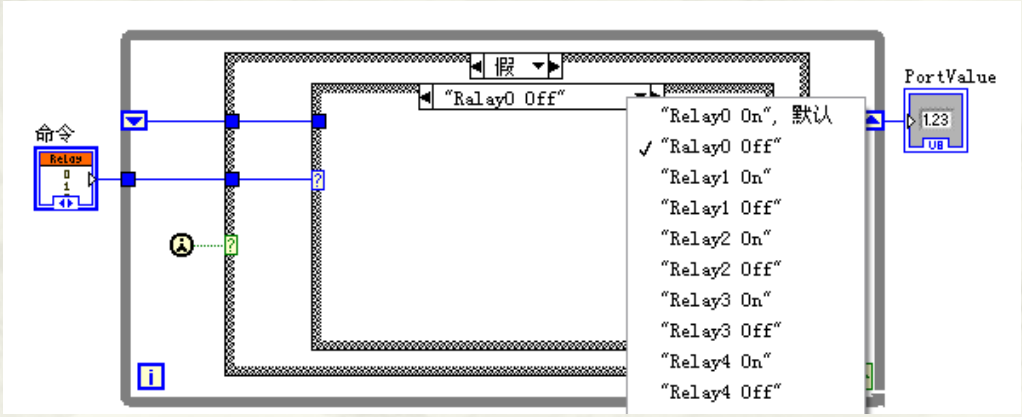
- 10.1.1 合理地使用数据流
- 10.1.2 LV2全局变量
- 10.1.3 值变化与上升下降沿
- 10.1.4 定时触发与计数器





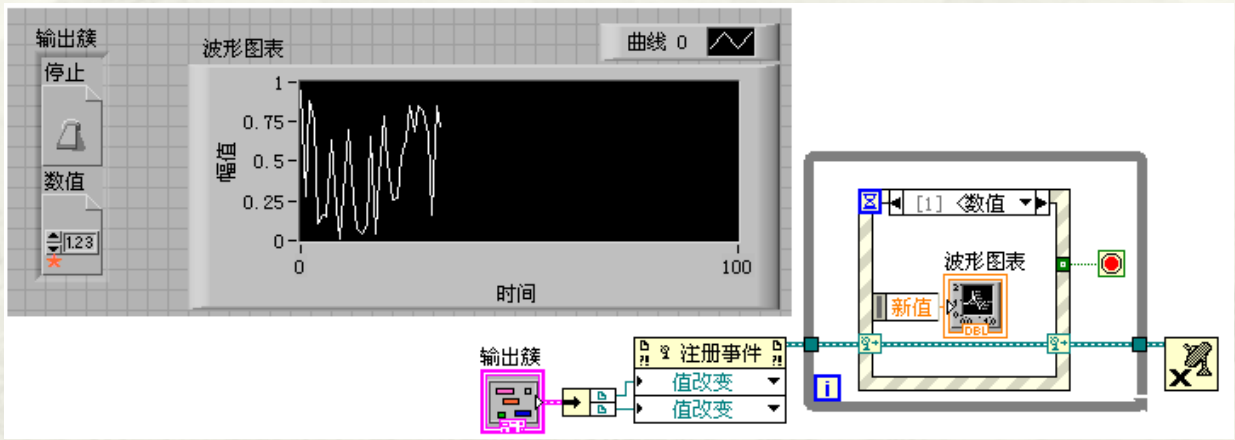
10.2 Action Engine

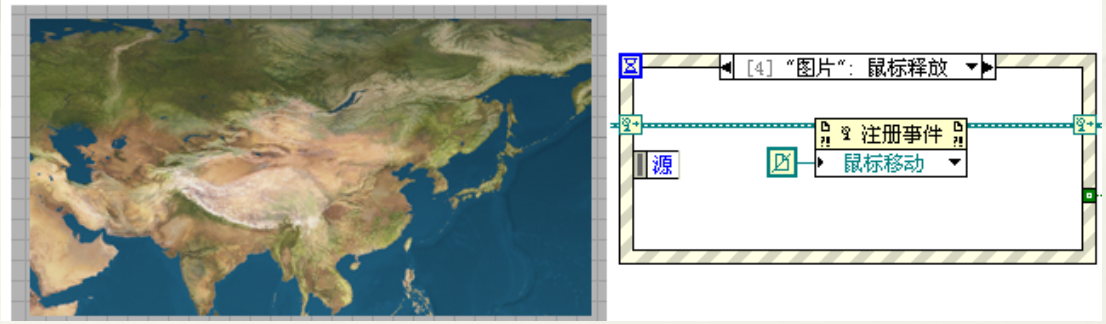
- 10.2.1 准备建立动作机
- 10.2.2 建立动作机的步骤



10.3 用户事件与动态注册事件

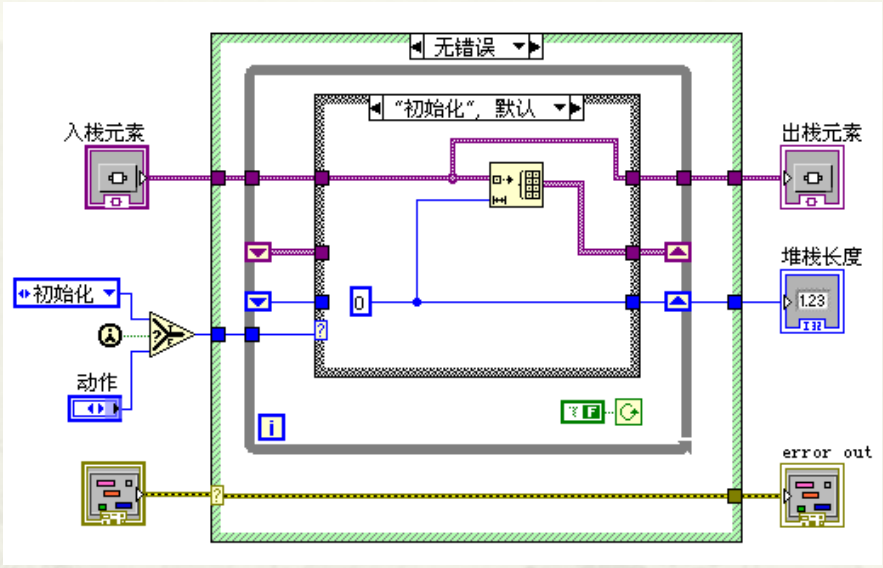
- 10.3.1 用户事件
- 10.3.2 动态注册事件

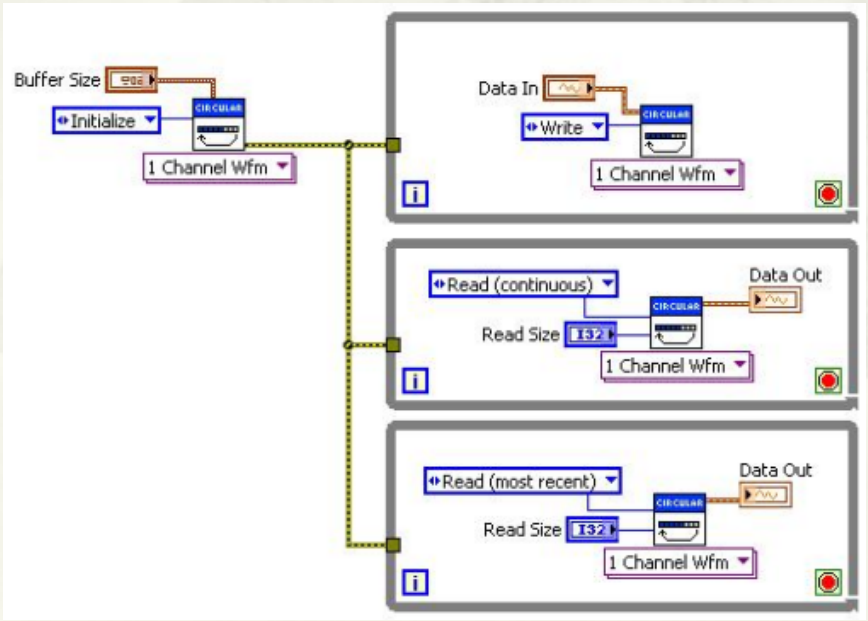


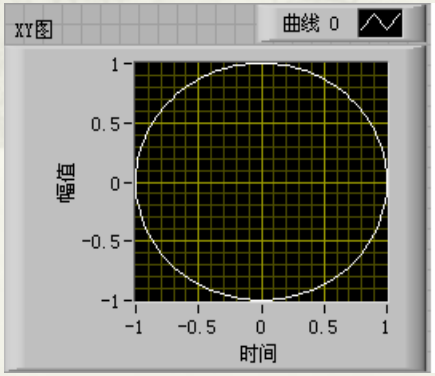


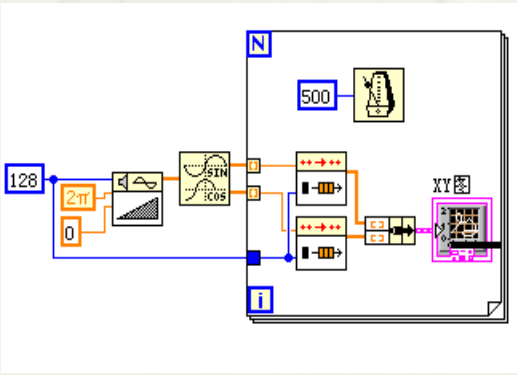
10.4堆栈、数据缓冲区

- 10.4.1 堆栈的实现
- 10.4.2 数据缓冲区



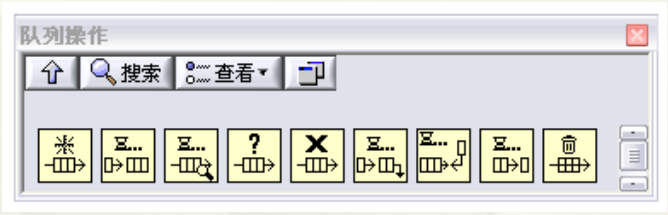


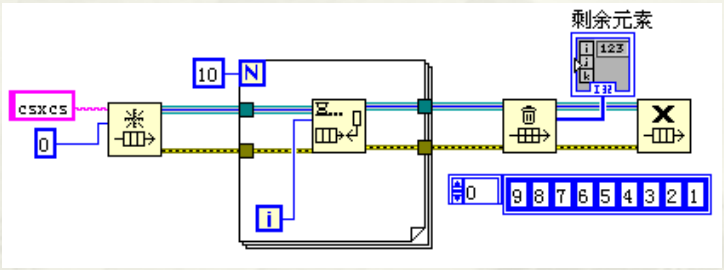


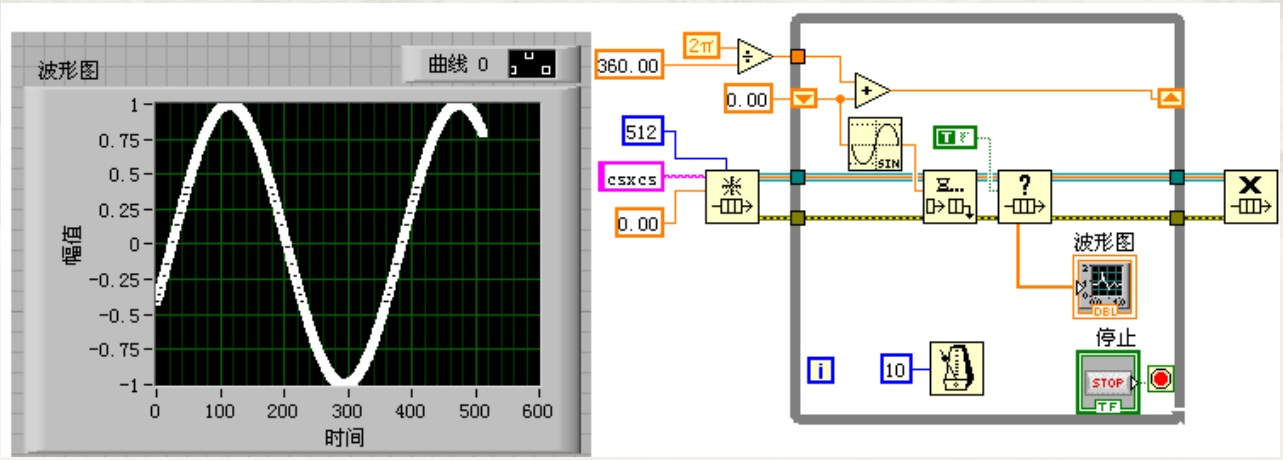


10.5 同步控制技术

- 10.5.1 队列
- 10.5.2 通知器与全局变量
- 10.5.3 信号量与集合点

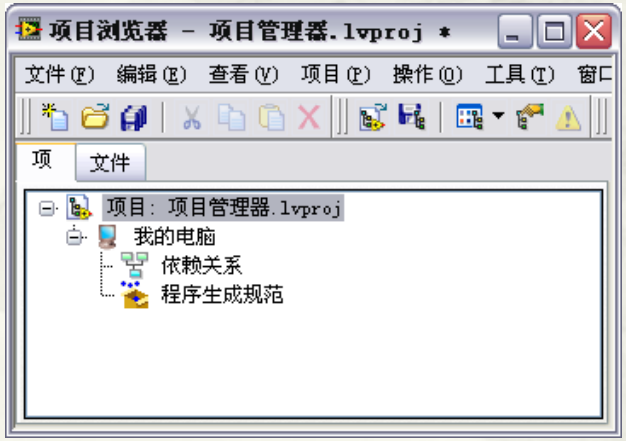


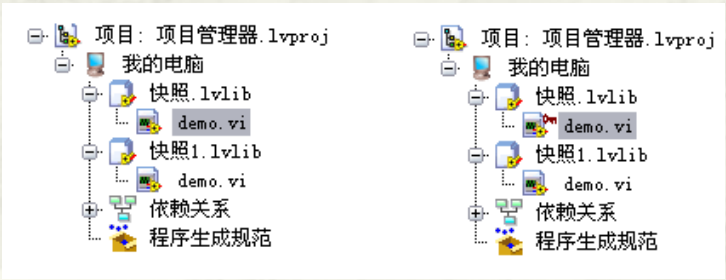




10.6 项目管理器

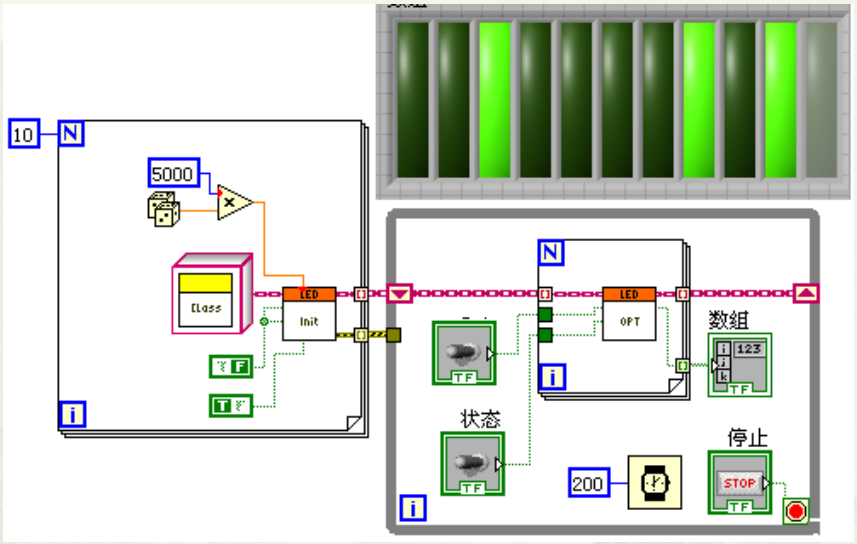
- 10.6.1 项目管理器的结构
- 10.6.2 虚拟文件夹与项目库

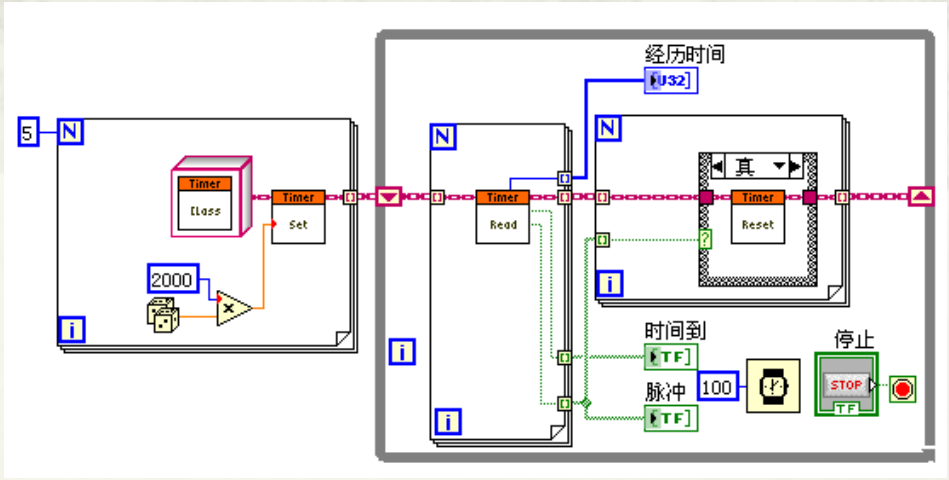


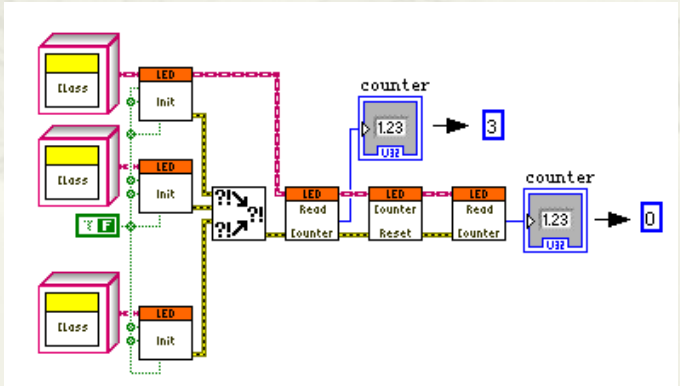


10.7 面向对象编程

- 10.7.1 面向对象编程的基本概念
- 10.7.2 类的封装特性
- 10.7.3 类的继承特性
- 10.7.4 类的多态特性
- 10.7.5 类变量
- 10.7.6 动态加载类与引用转换

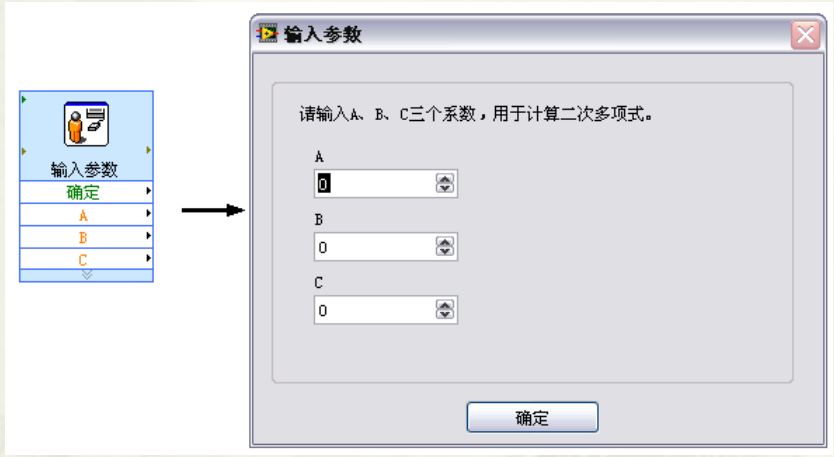






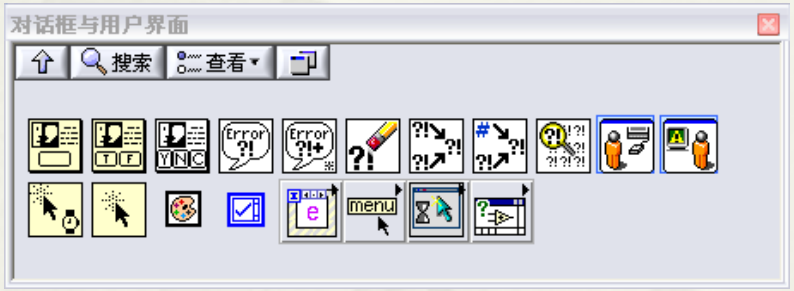
第11章 人机交互与编程风格

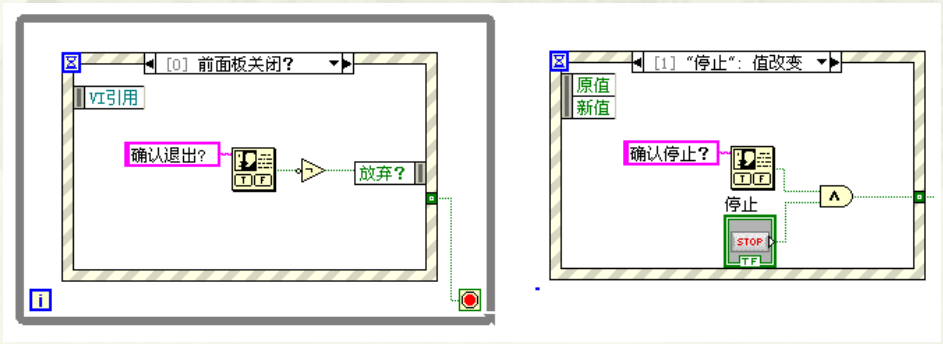
- 11.1 对话框
- 11.2 菜单
- 11.3 光标工具
- 11.4 选项卡、分隔栏与子面板
- 11.5 Xcontrol
- 11.6 错误处理
- 11.7 LabVIEW的编程风格
- 11.8 小结



11.1 对话框

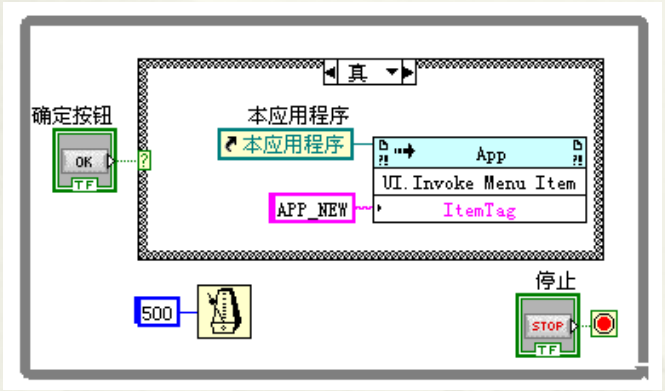
- 11.1.1 内置对话框
- 11.1.2 用户输入和显示对话框
- 11.1.3 定制VI对话框

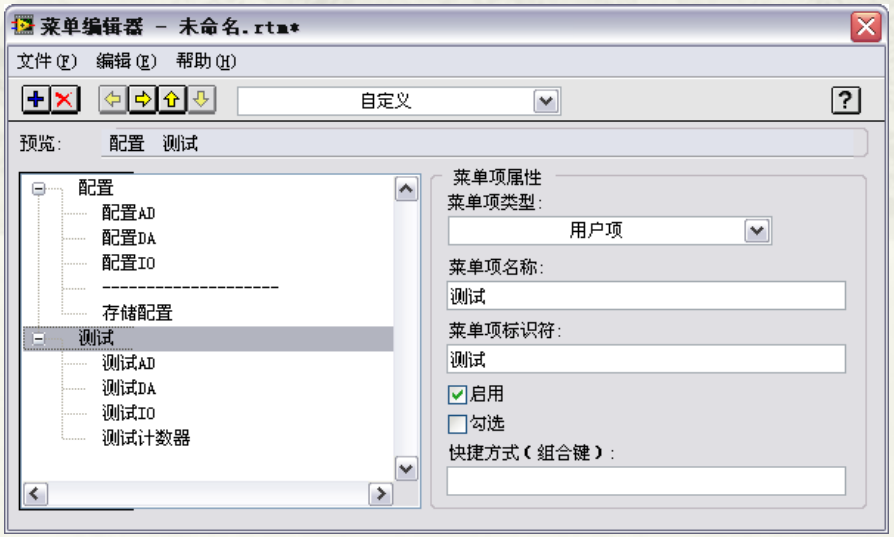


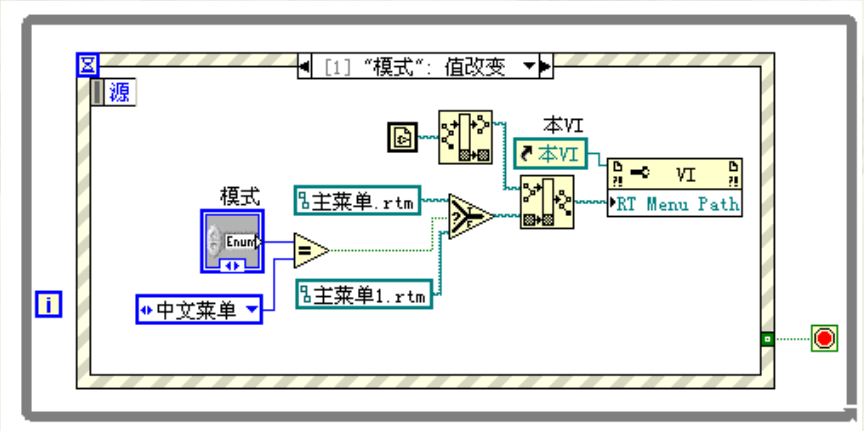


11.2 菜单

- 11.2.1 创建静态菜单
- 11.2.2 菜单相关函数
- 11.2.3 动态创建菜单函数
- 11.2.4 动态创建菜单
- 11.2.5 调用多个静态菜单
- 11.2.6 存储动态建立的菜单
- 11.2.7自动触发预定义菜单项

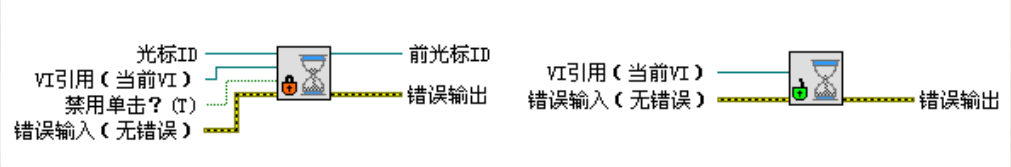


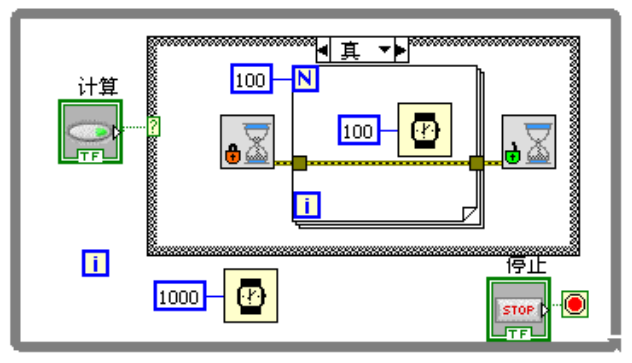


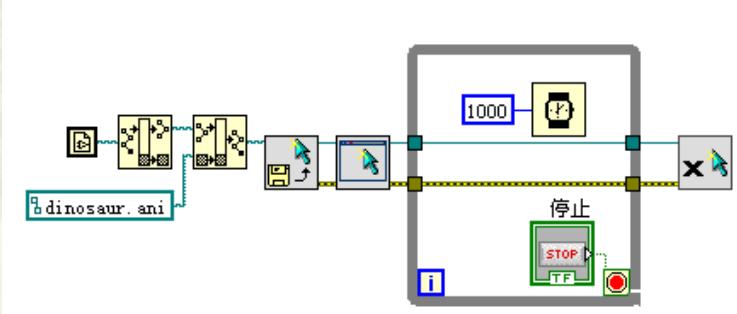


11.3 光标工具

- 11.3.1 设置忙碌状态与取消设置忙碌状态
- 11.3.2 使用光标文件

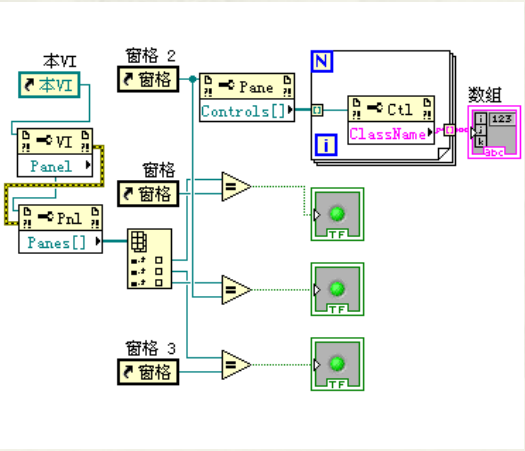


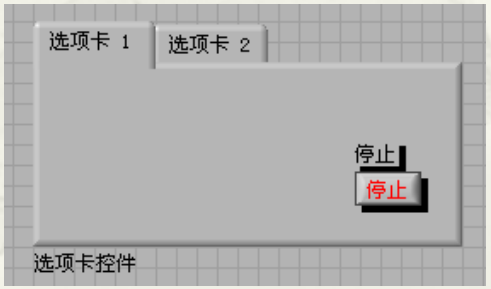




11.4 选项卡、分隔栏与子面板

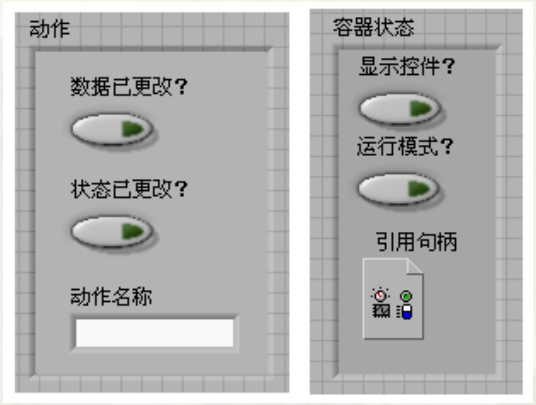
* 11.4.1 选项卡控件
* 11.4.2 分隔栏控件
* 11.4.3 子面板

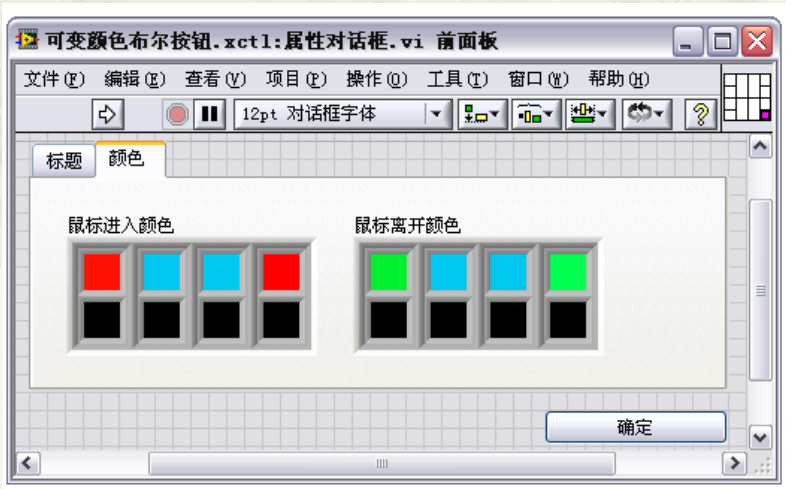


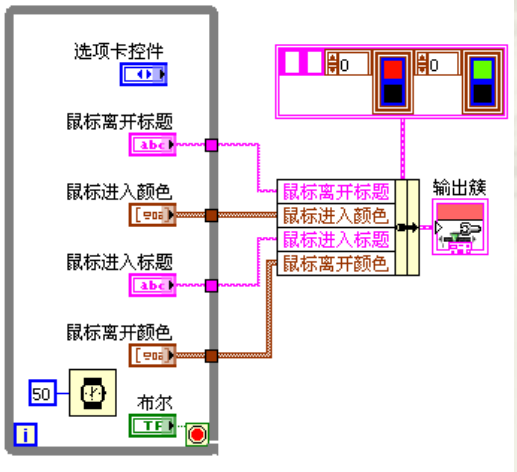


11.5 XControl

- 11.5.1 传统更改控件方式
- 11.5.2 新建Xcontrol
- 11.5.3 修改数据控件和状态控件
- 11.5.4 修改初始化VI和外观VI
- 11.5.5 创建属性和方法
- 11.5.6 调试Xcontrol
- 11.5.7 自定义属性对话框与快捷菜单

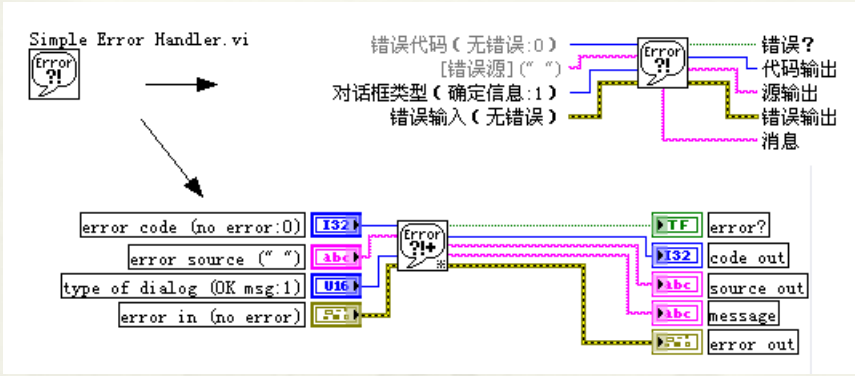


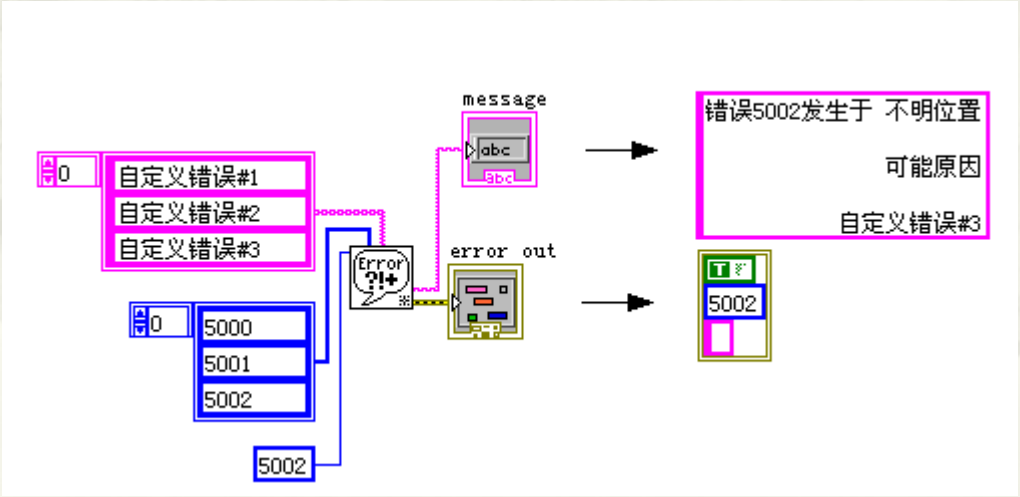




11.7 错误处理

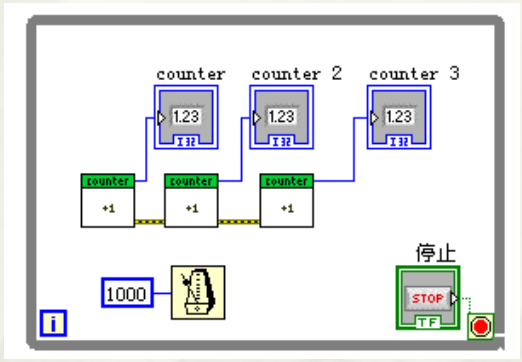
* 11.7.1 错误簇
* 11.7.2 常用错误处理函数

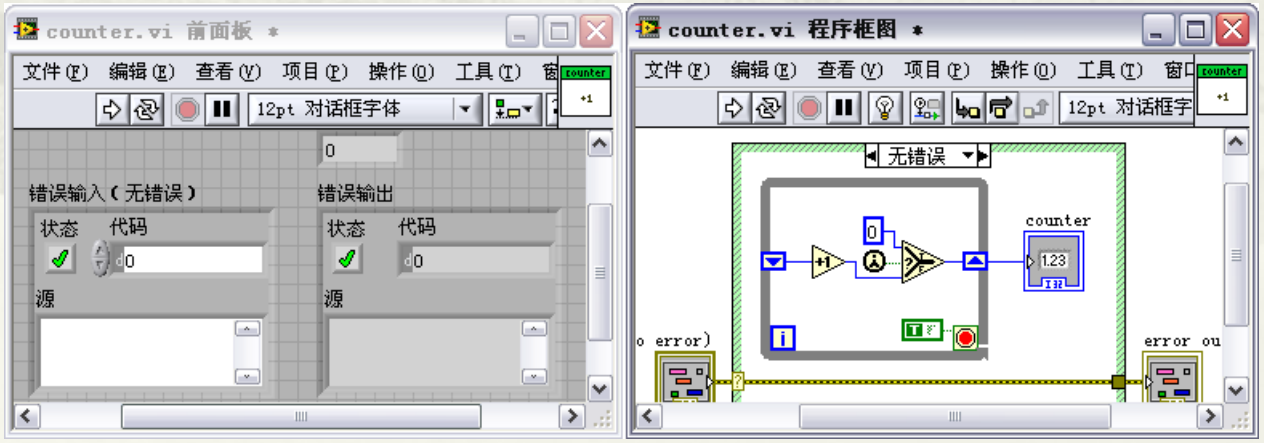




第12章 VI模板、设计模式、状态图

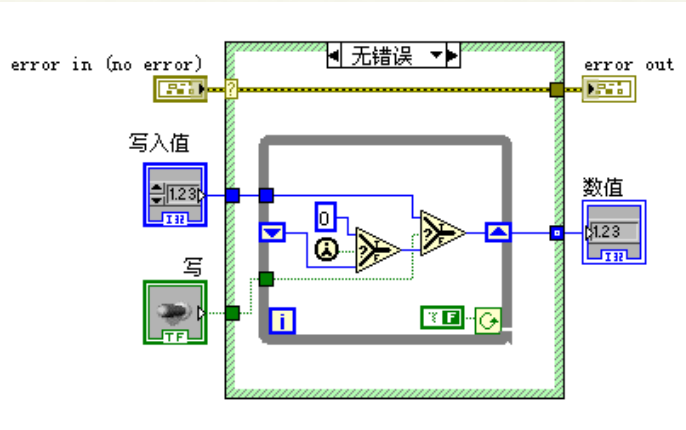
- 12.1程序的基本单元VI
- 12.2 LabVIEW标准设计模式
- 12.3 简单设计模式
- 12.4 古典状态机
- 12.5 状态机工具包（State diagram）
- 12.6 消息队列状态机
- 12.7状态图工具包（Statechart）

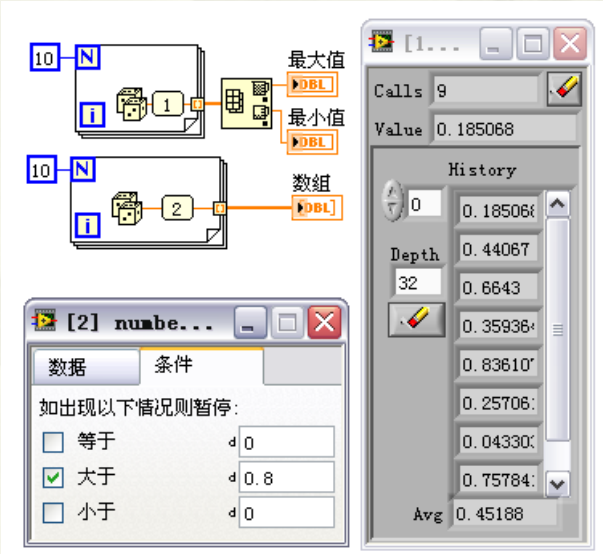


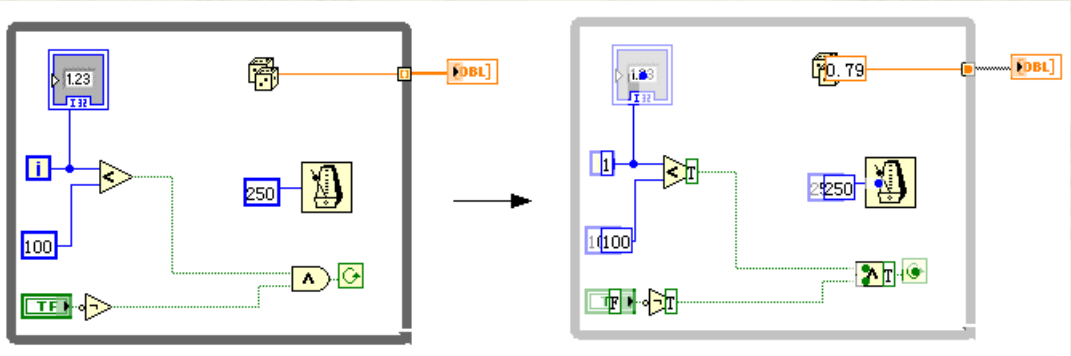


12.1 程序的基本单元VI

- 12.1.1 可重入VI
- 12.1.2 VI模板与代码重用
- 12.1.3 VI的调试
- 12.1.4 VI的重构

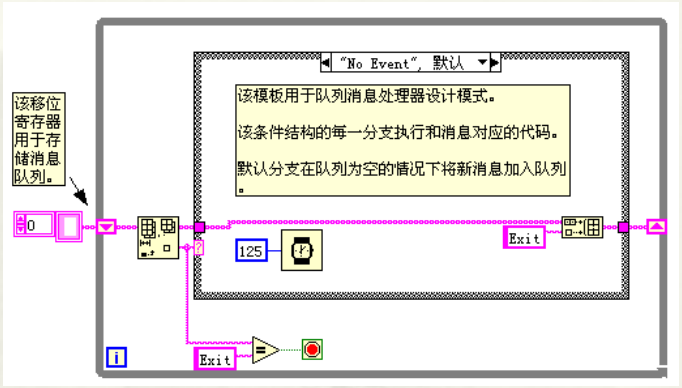


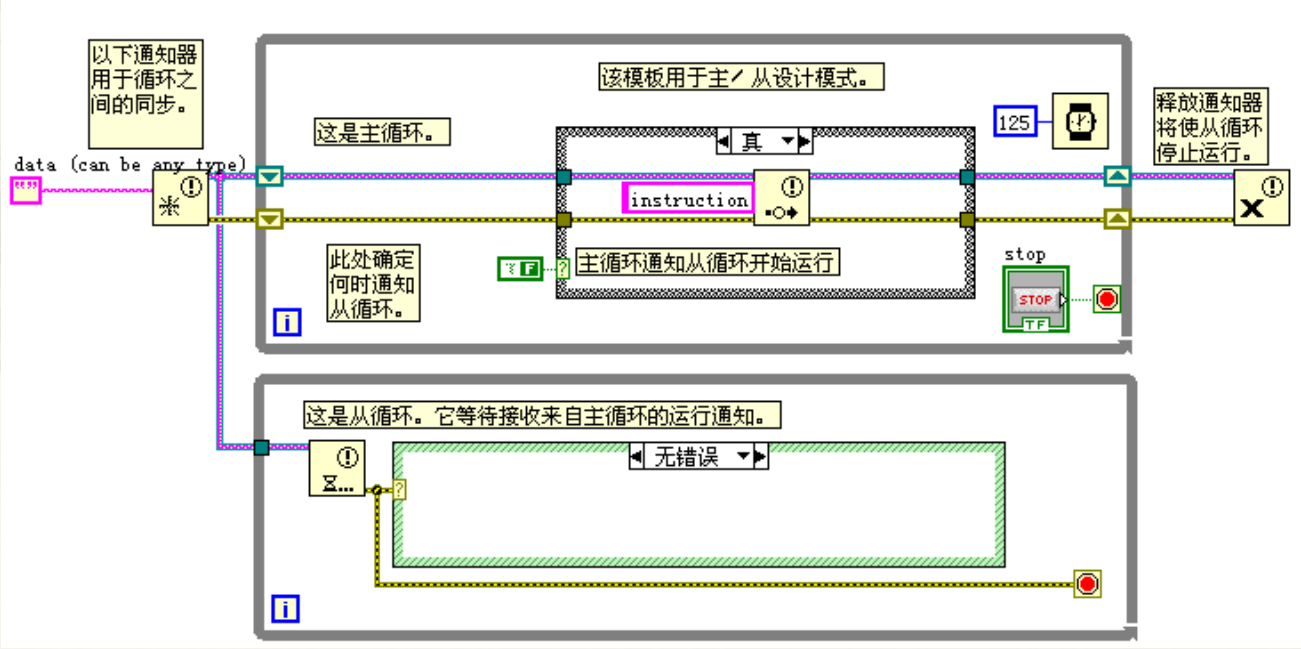




12.2 LabVIEW标准设计模式

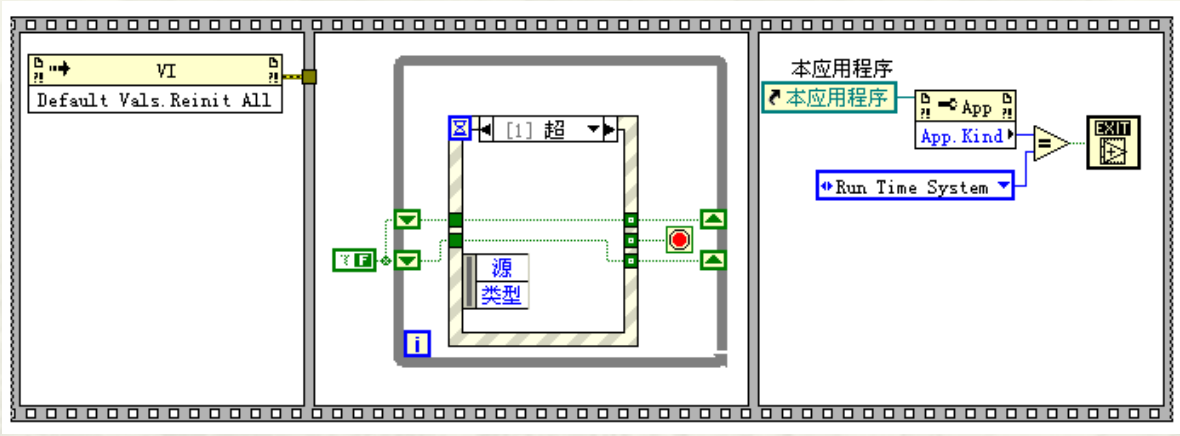
- 12.2.1 使用事件的顶层应用程序
- 12.2.2 使用事件的对话框
- 12.2.3 设计模式之标准状态机
- 12.2.4 设计模式之队列消息处理器
- 12.2.5 设计模式之用户界面事件处理器
- 12.2.6设计模式之生产者/消费者设计模式（事件）
- 12.2.7设计模式之生产者/消费者设计模式（数据）
- 12.2.8设计模式之主从设计模式（Master/Slave）

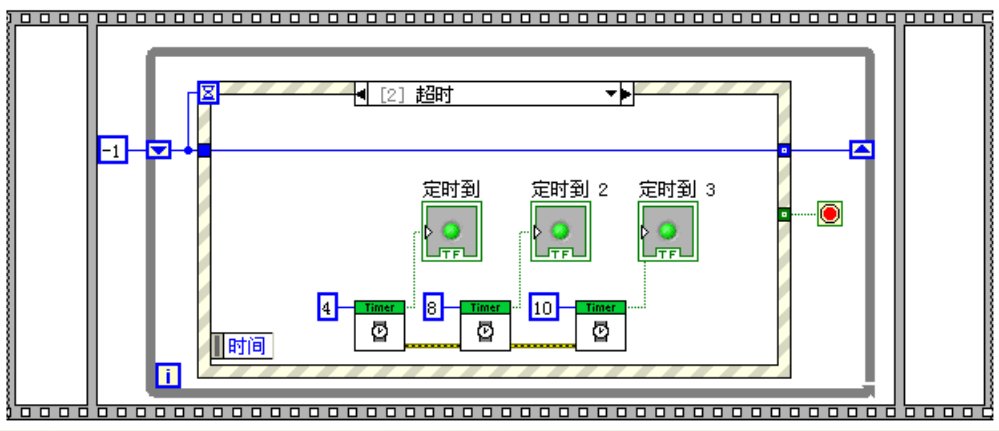




12.3 简单设计模式

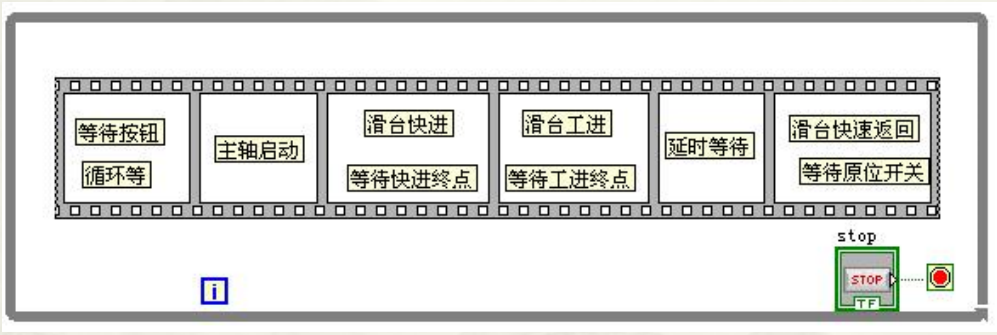
- 12.3.1 顺序结构简单设计模式
- 12.3.2事件结构与定时结构简单设计模式

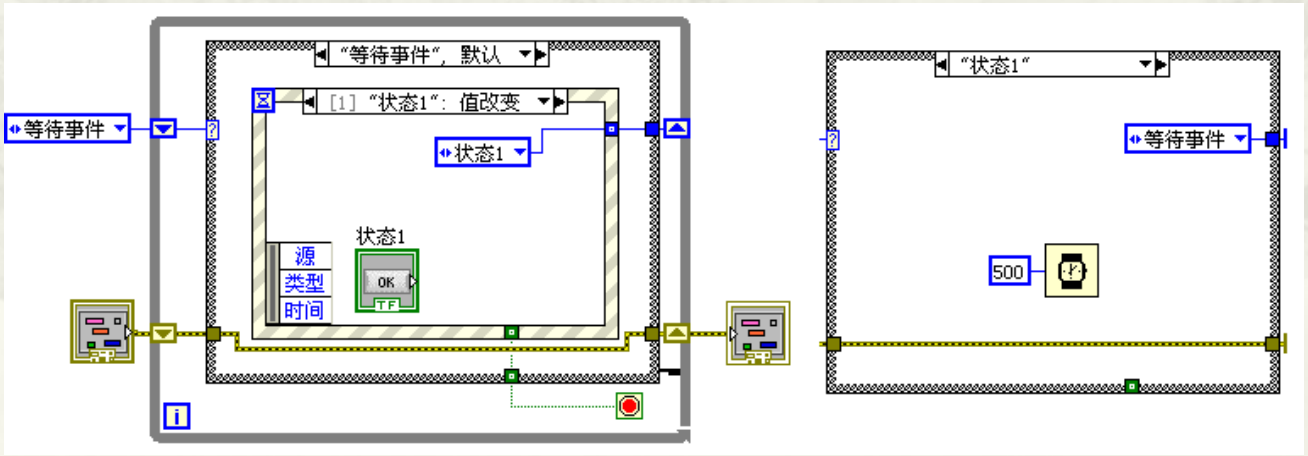




12.4 古典状态机

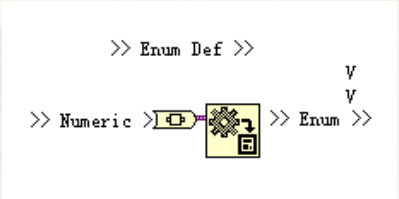
- 12.4.1 采用顺序结构
- 12.4.2 采用顺序状态机
- 12.4.3 处理公共状态
- 12.4.4 事件状态机
- 12.4.5 早期界面处理状态机

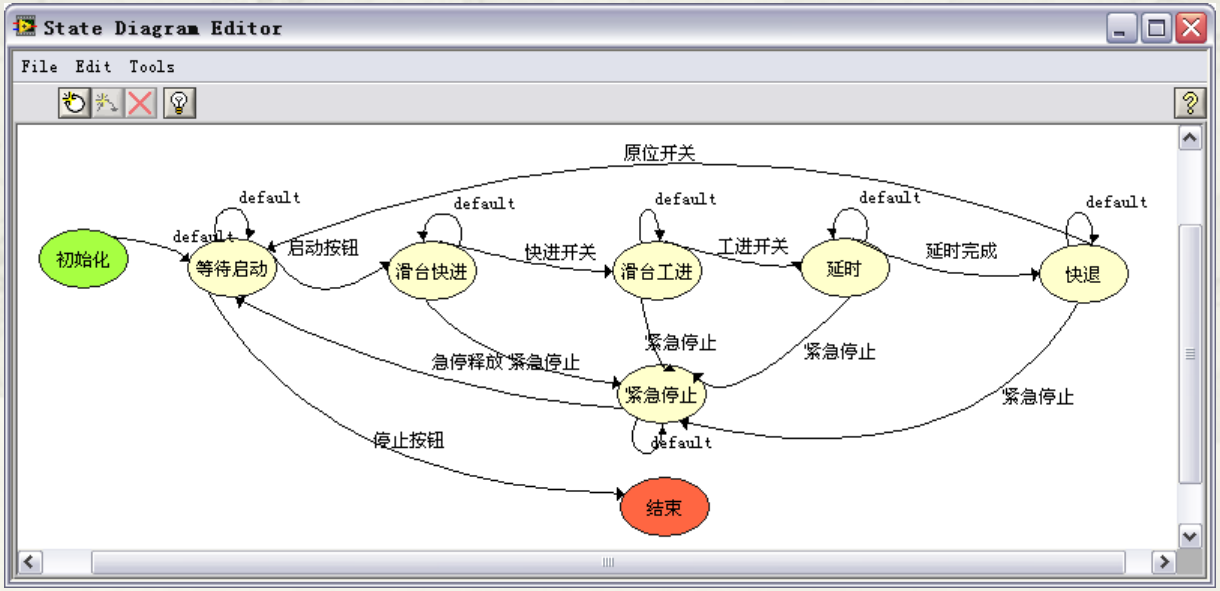




12.5 状态机工具包（State diagram）

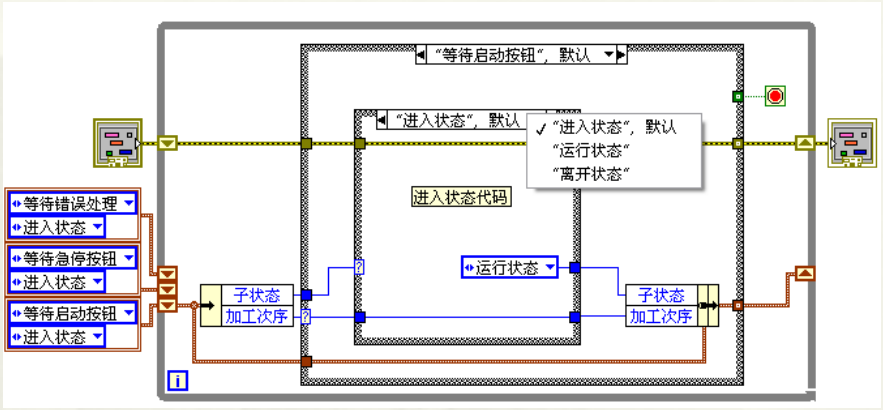
- 12.5.1 调用状态机工具
- 12.5.2 使用状态图编辑器
- 12.5.3 添加转换条件和状态代码
- 12.5.4 选择独立运行或者子VI方式

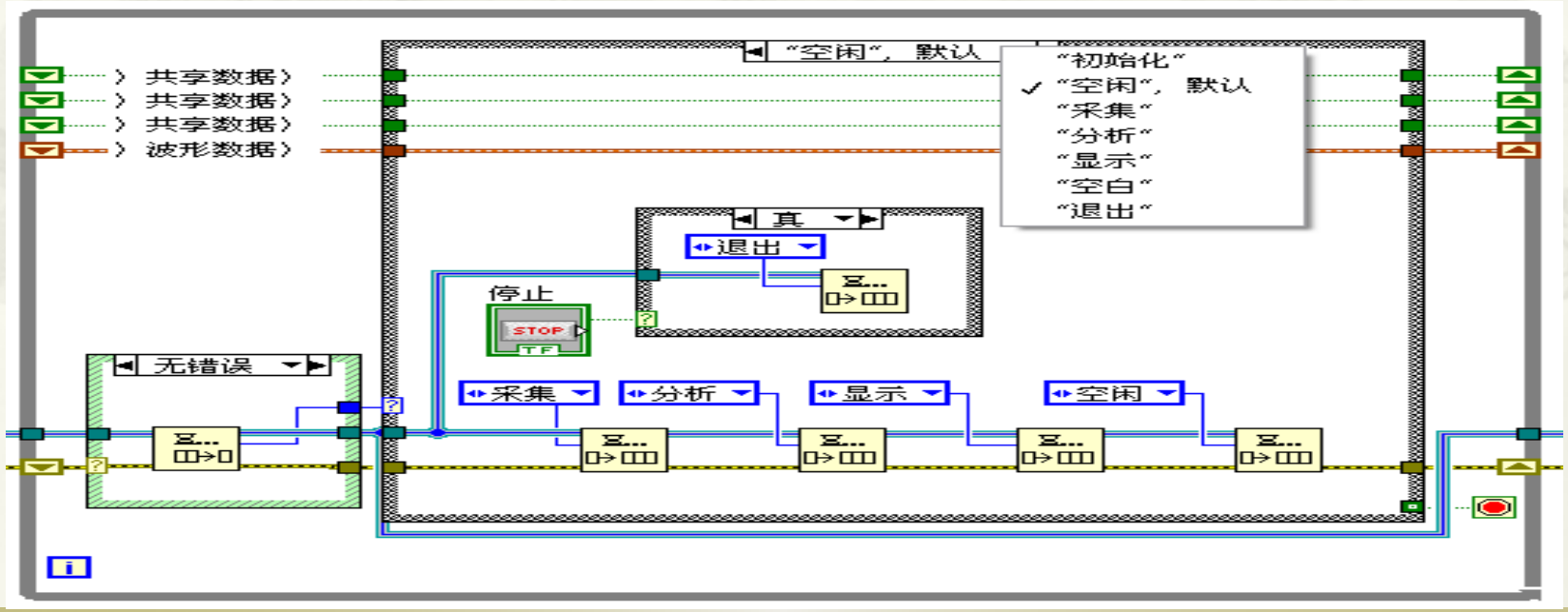




12.6 消息队列状态机

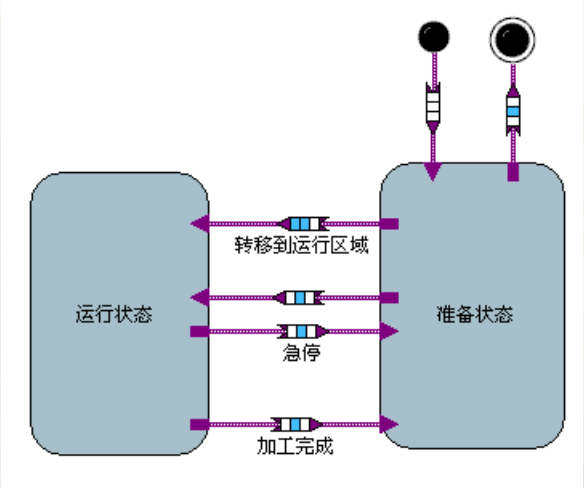
- 12.6.1 队列消息状态机的基本结构
- 12.6.2 命令方式队列消息状态机
- 12.6.3 命令数据方式消息队列状态机
- 12.6.4 进入、运行和离开状态的处理
- 12.6.5 事件驱动方式消息队列状态机

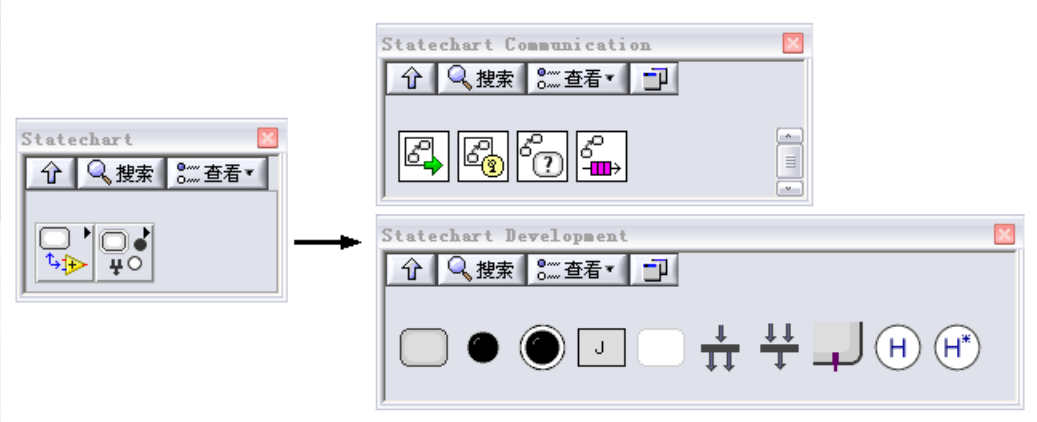




12.7 状态图工具包（Statechart）

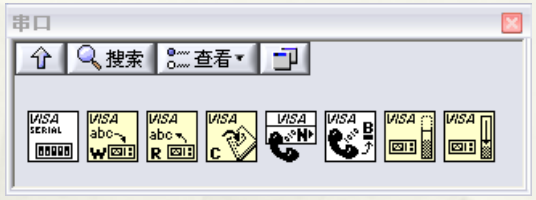
- 12.7.1 状态图工具包简介
- 12.7.2 同步和异步方式
- 12.7.3 创建状态图
- 12.7.4 同步型状态图
- 12.7.5 状态图的调用和调试
- 12.7.6 异步型状态图
- 12.7.7 区域、超级状态和子状态
- 12.7.8 多区域并发、连接、分叉与子图
- 12.7.9 高级应用函数

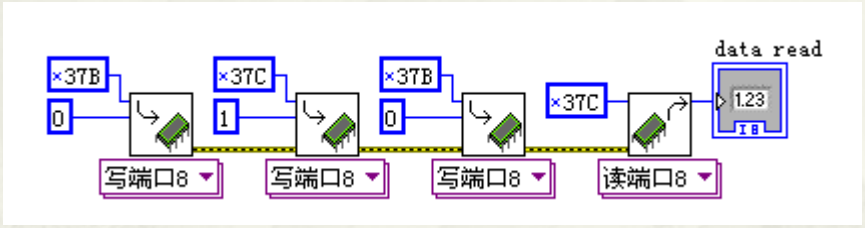


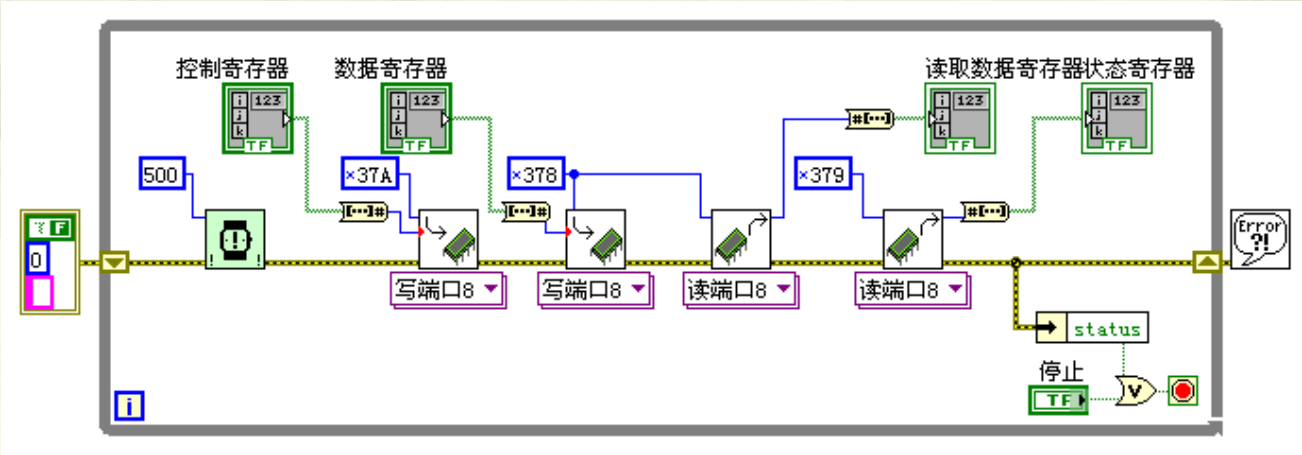


第13章 串并口通讯、网络与DSC

- 13.1 串口通讯
- 13.2 并口通讯
- 13.3 共享变量
- 13.4 DataSocket
- 13.5 TCP与UDP网络通讯
- 13.6 DSC工具包

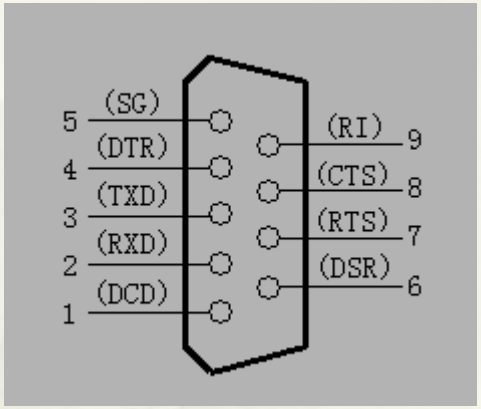


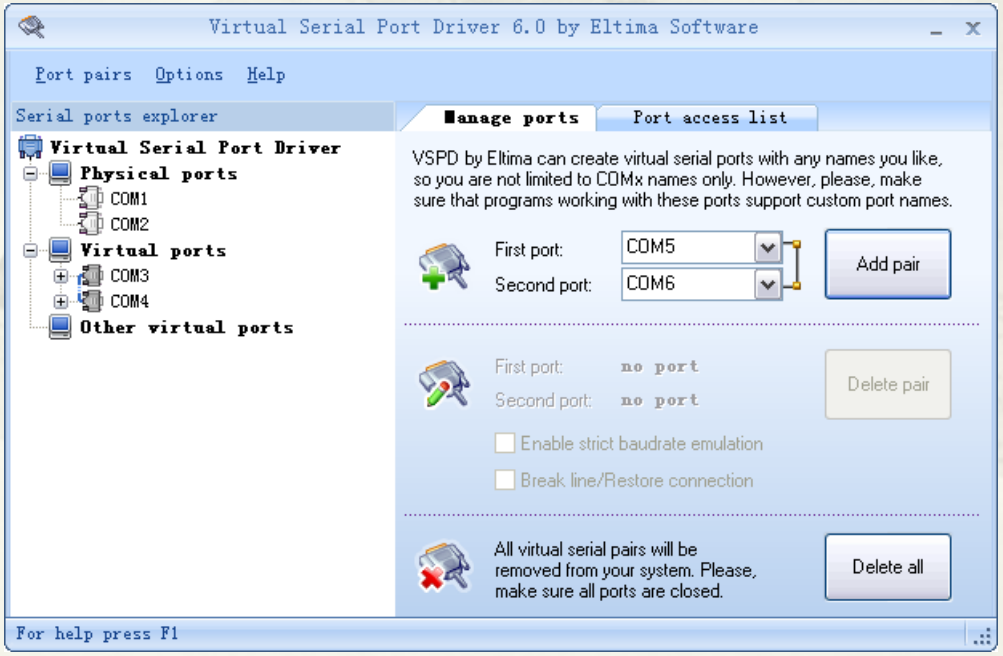




13.1 串口通讯

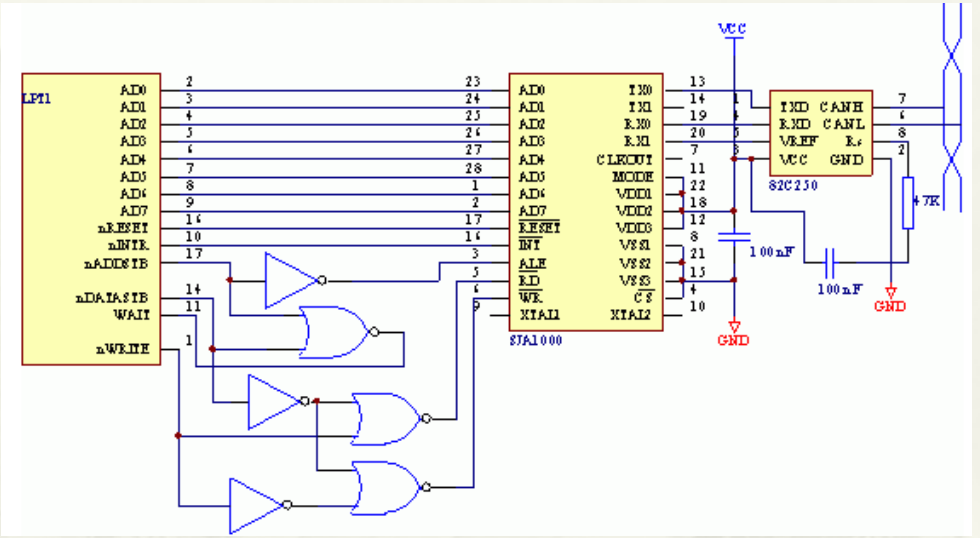
- 13.1.1 串口通信的基本概念
- 13.1.2 串口通讯的准备工作
- 13.1.3 串口通讯函数及其应用

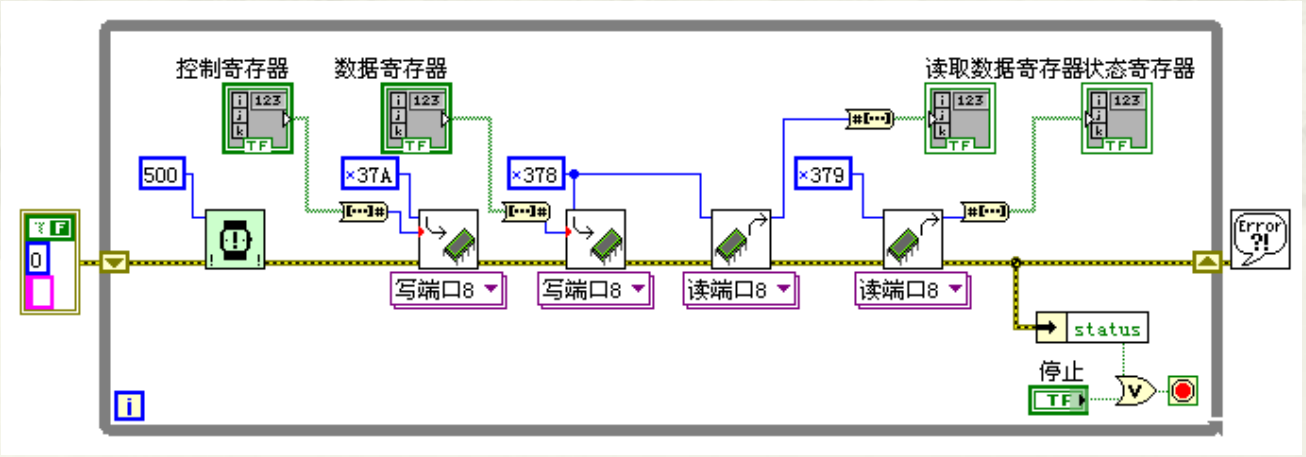




13.2 并口通讯

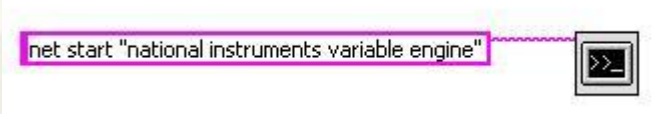
- 13.2.1 设置并口通讯模式
- 13.2.2 字节型数据传送
- 13.2.3 EPP数据传送

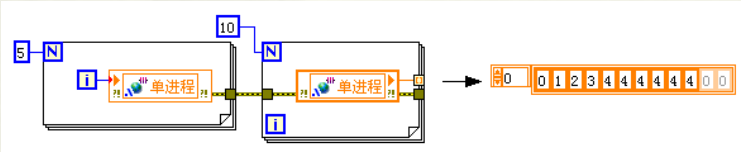


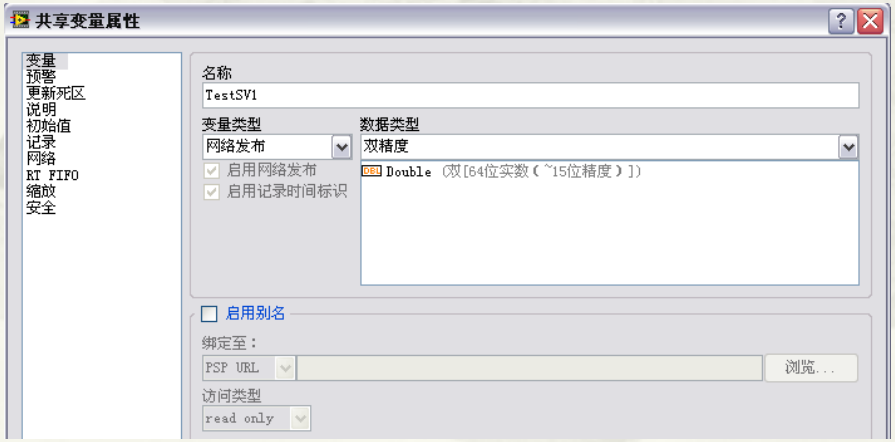


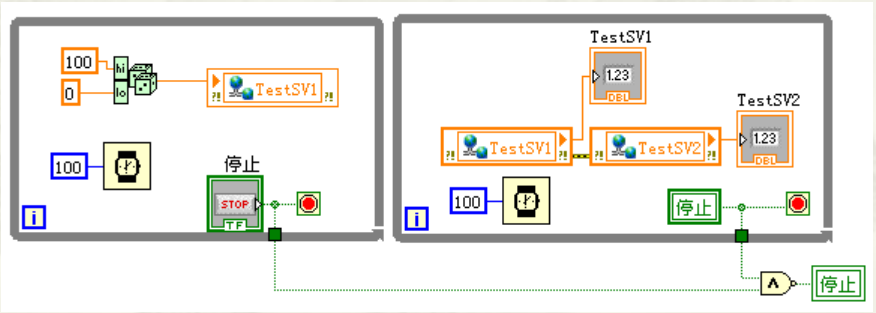
13.3 共享变量

- 13.3.1 共享变量与共享变量引擎
- 13.3.2 创建与监视共享变量
- 13.3.3 共享变量的内部缓冲机制
- 13.3.4 共享变量的批量创建、部署与引用



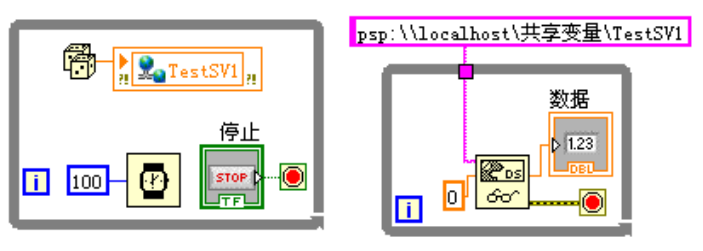


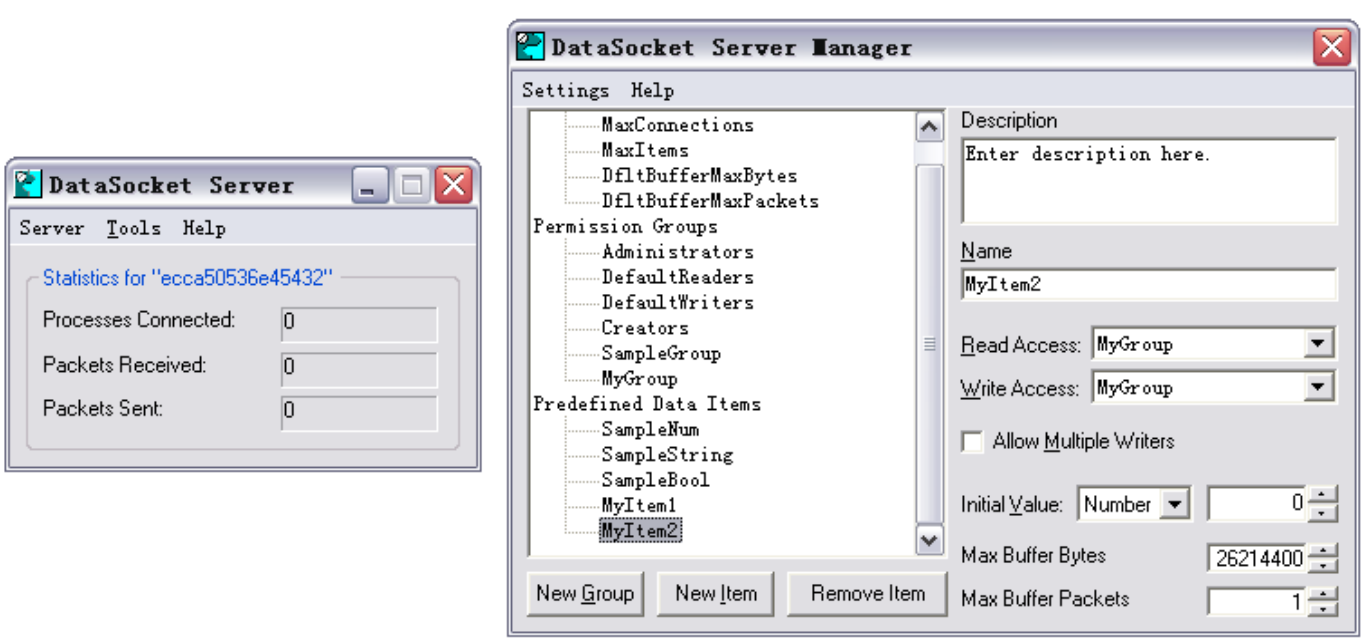




13.4 DataSocket

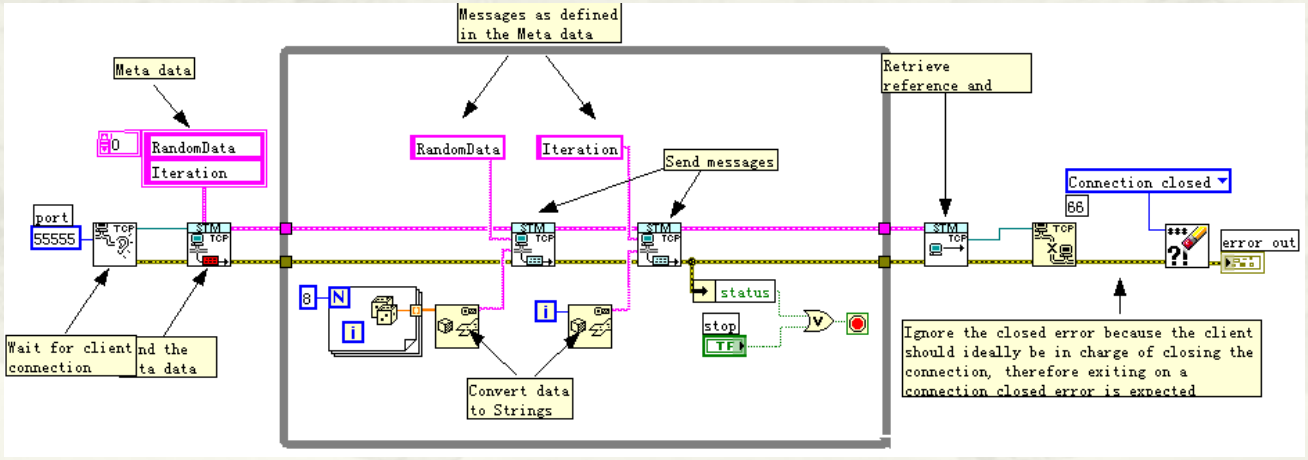
- 13.4.1 DataSocket支持的协议与URL
- 13.4.2 DataSocket服务器与服务管理器
- 13.4.3 DataSocket API与控件绑定





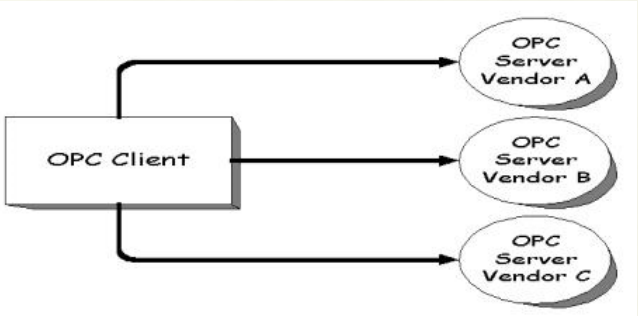
13.5 TCP与UDP网络通讯

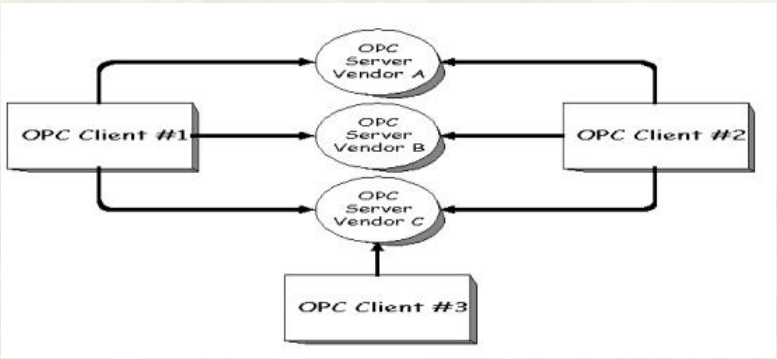
- 13.5.1 LabVIEW TCP函数
- 13.5.2 TCP STM库
- 13.5.3 LabVIEW UDP函数

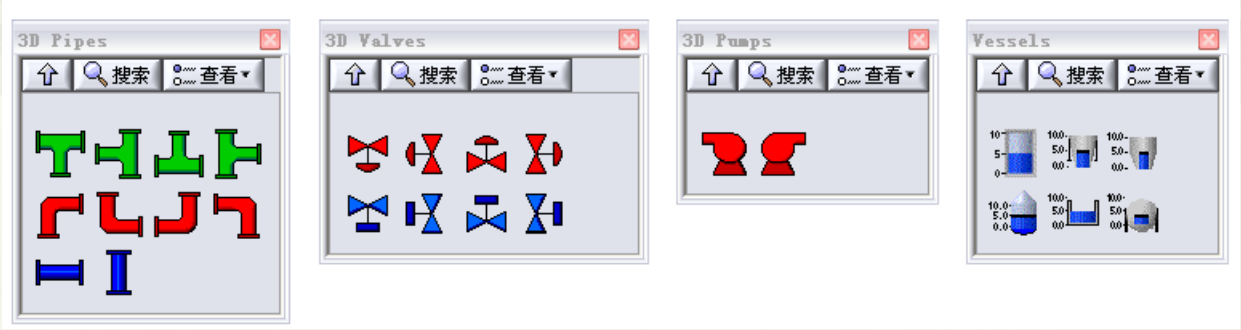


13.6 DSC工具包

- 13.6.1 OPC与DSC的基本概念
- 13.6.2 DSC强大的图形化能力
- 13.6.3 OPC配置与IO变量
- 13.6.4 Modbus
- 13.6.5 共享变量的属性
- 13.6.6 共享变量引擎SVE函数
- 13.6.7 预警与事件
- 13.6.8 数据记录
- 13.6.9 安全与权限管理

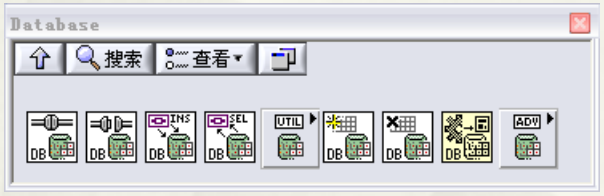


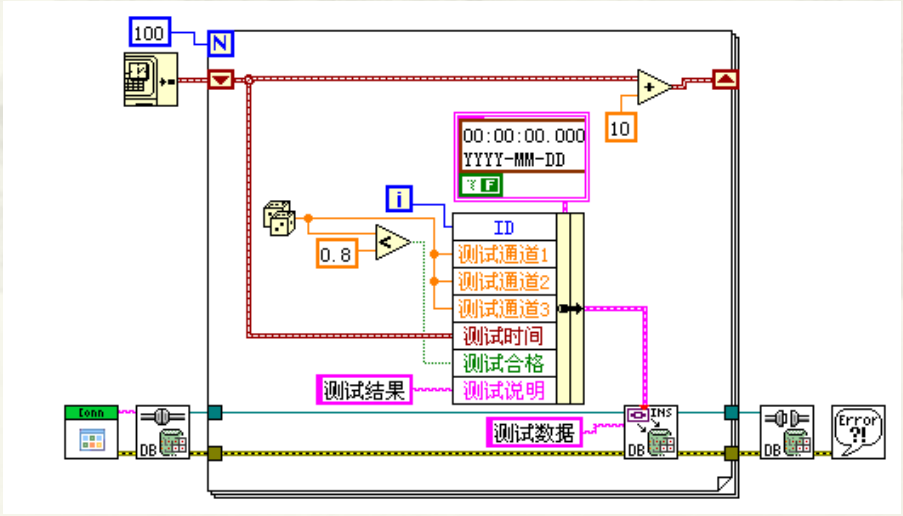


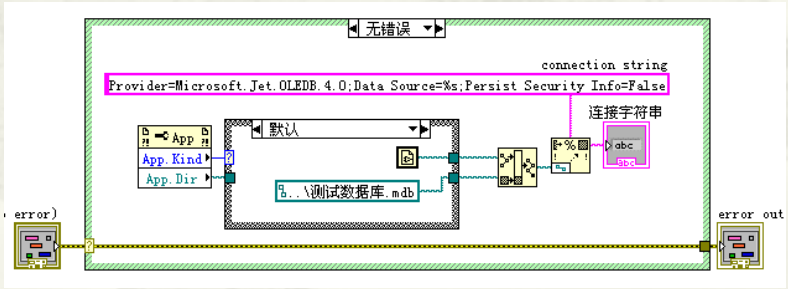


第14 章数据库、报表工具

- 14.1 准备使用数据库工具包
- 14.2 数据库基本操作
- 14.3 报表与报表生成工具包
- 14.4 利用报表工具操作Excel
- 14.5 利用报表工具操作Word

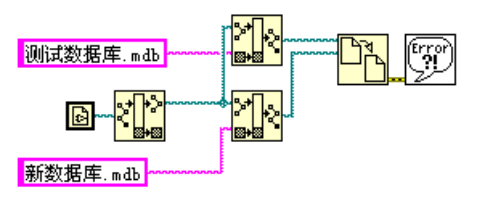


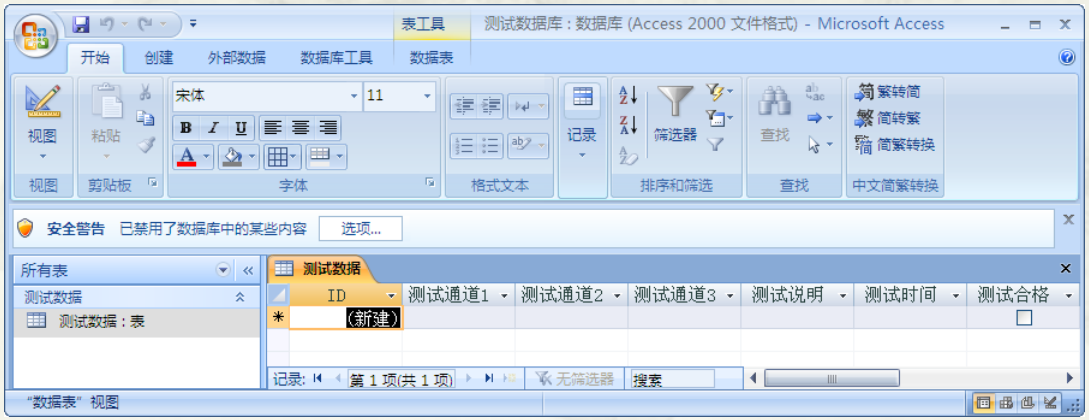




14.1 准备使用数据库工具包

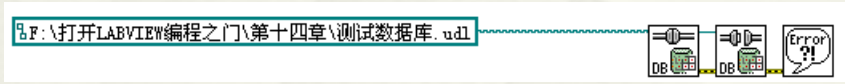
- 14.1.1 创建数据库
- 14.1.2 建立数据源
- 14.1.3 数据工具包支持的数据类型
- 14.1.4 ADO模型

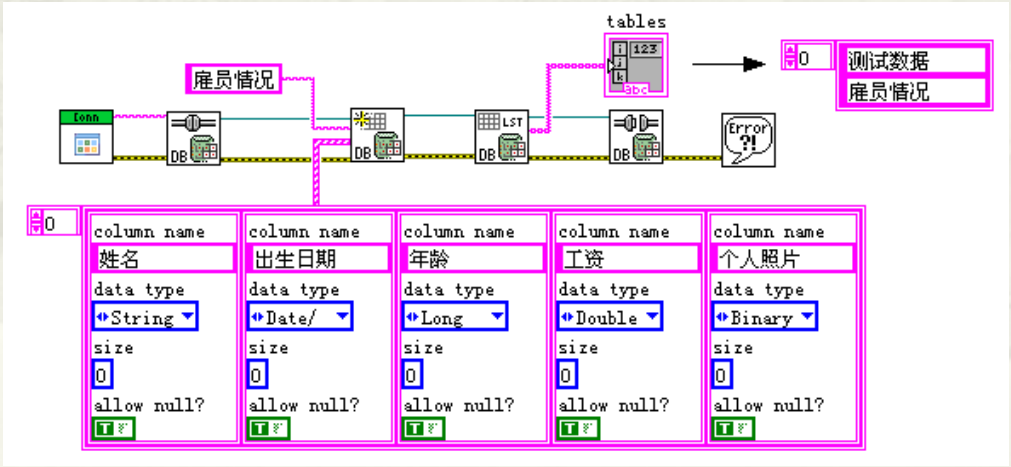




14.2 数据库基本操作

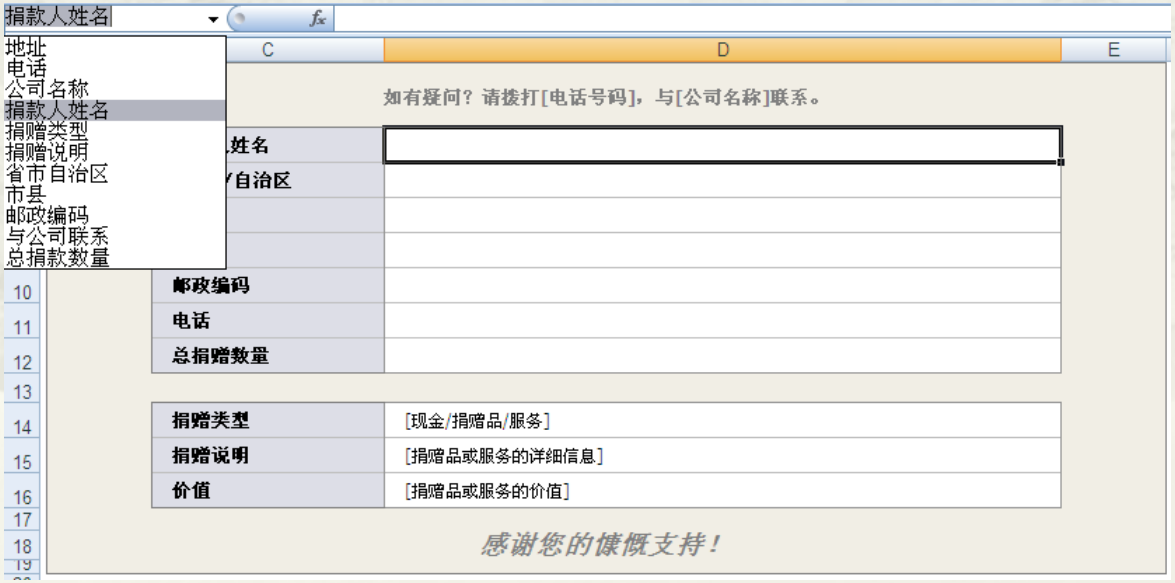
- 14.2.1 建立连接
- 14.2.2 表操作
- 14.2.3 插入数据
- 14.2.4 读取数据
- 14.2.5 记录集与数据浏览
- 14.2.6 事务与提交
- 14.2.7 使用命令对象和SQL语句





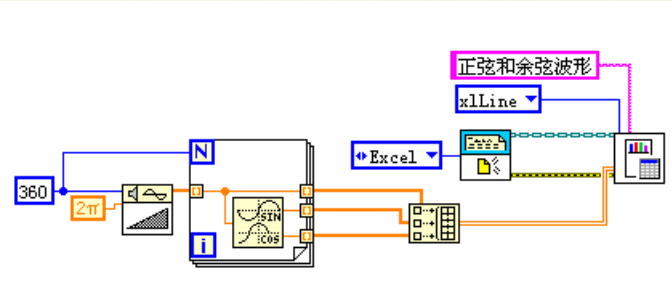
14.3 报表与报表生成工具包

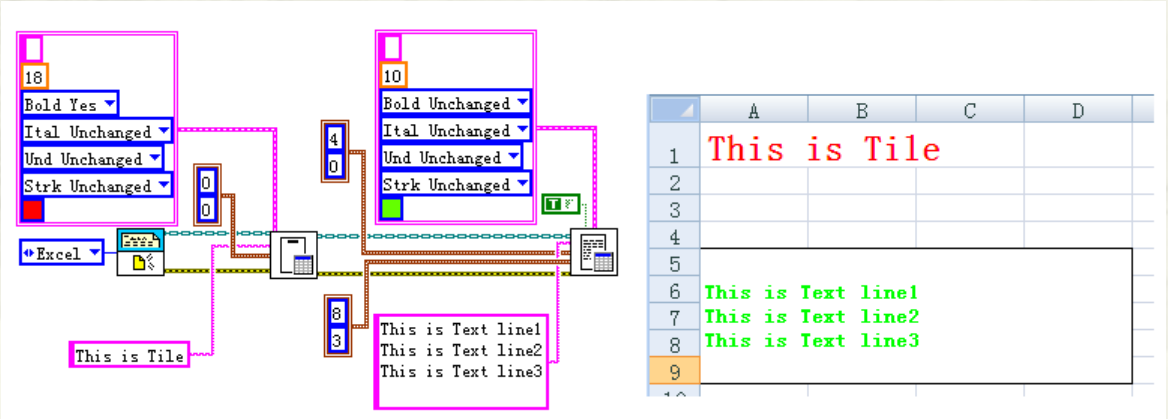
- 14.3.1 LabVIEW中的报表VI
- 14.3.2 VI说明信息与HTML报表
- 14.3.3 报表布局与高级报表VI
- 14.3.4 利用Word和Excel模板创建报表



14.4 利用报表工具操作Excel

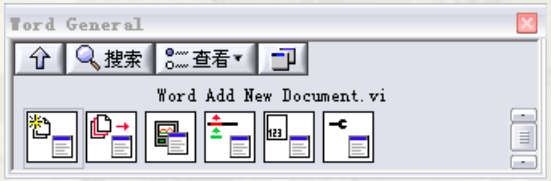
- 14.4.1 常用简单Excel函数
- 14.4.2 单元格操作VI
- 14.4.3 图表与图片VI
- 14.4.4 Excel通用函数和高级函数

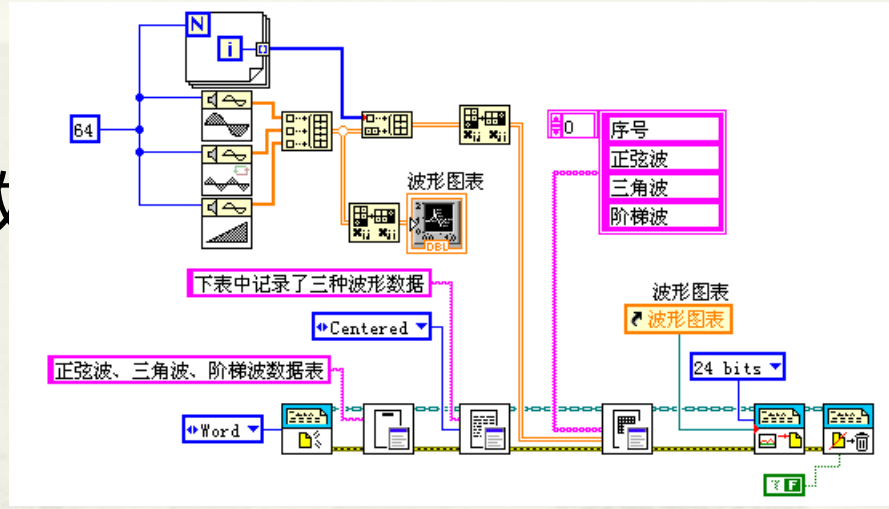


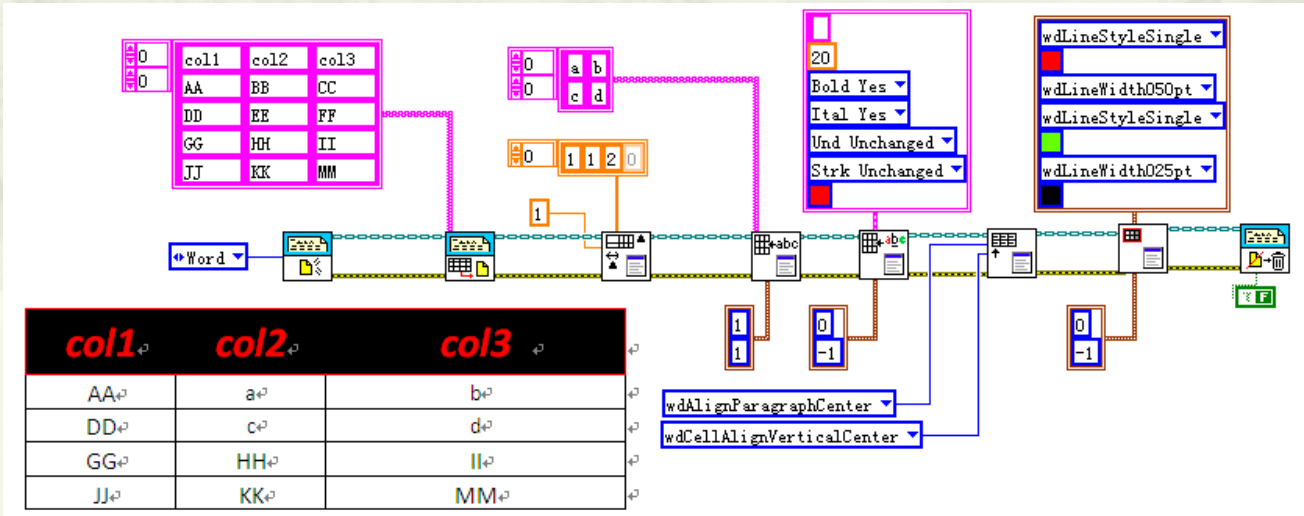


14.5 利用报表工具操作Word

- 14.5.1 Word易用函数
- 14.5.2 Word通用函数
- 14.5.3 Word表格与图表函数



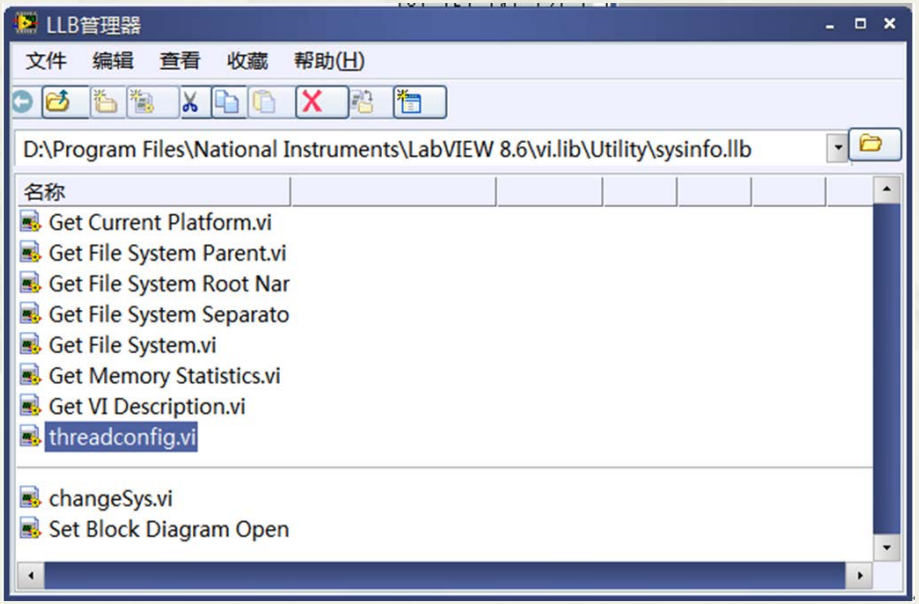




col1	col2	col3
AA	a	b
DD	c	d
GG	HH	II
JJ	KK	MM

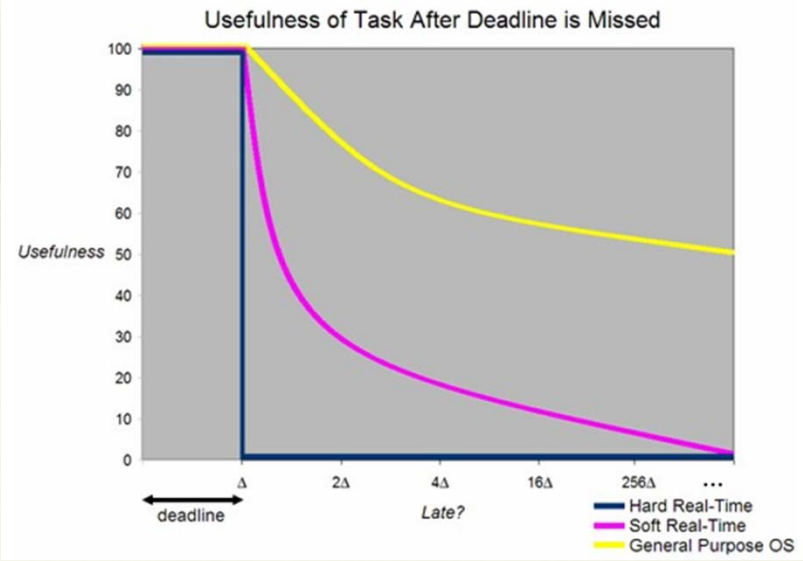
第15章 LabVIEW与实时系统

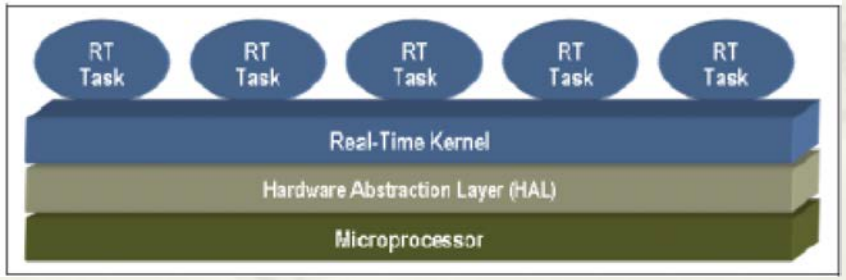
- 15.1 实时系统
- 15.2 Real-Time软件安装及其配置
- 15.3 Real-time高级编程及技巧



15.1 实时系统

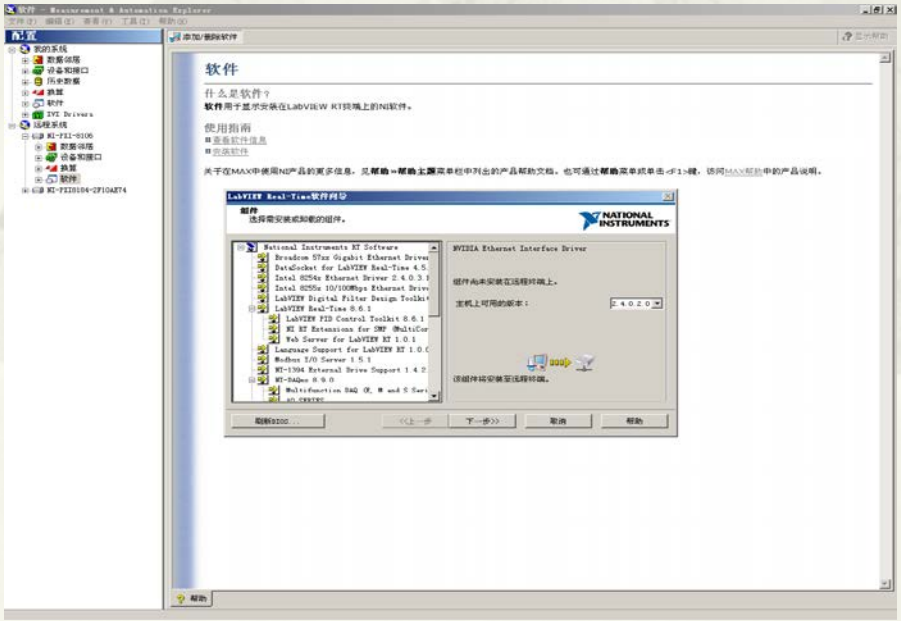
* 15.1.1 操作系统的概念
* 15.1.2 实时操作系统的概念
* 15.1.3 操作系统的有关名词解释
* 15.1.4 LabVIEW中的实时开发软件
* 15.1.5 LabVIEW支持的实时操作系统
* 15.1.6 LabVIEW Real-Time平台概述
* 15.1.7 LabVIEW Real-Time硬件平台的比较

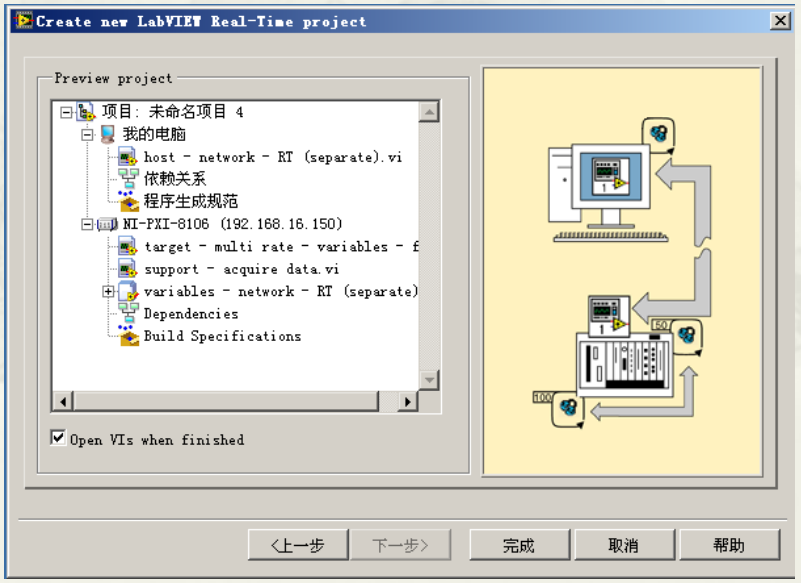




15.2 Real-Time软件安装及其配置

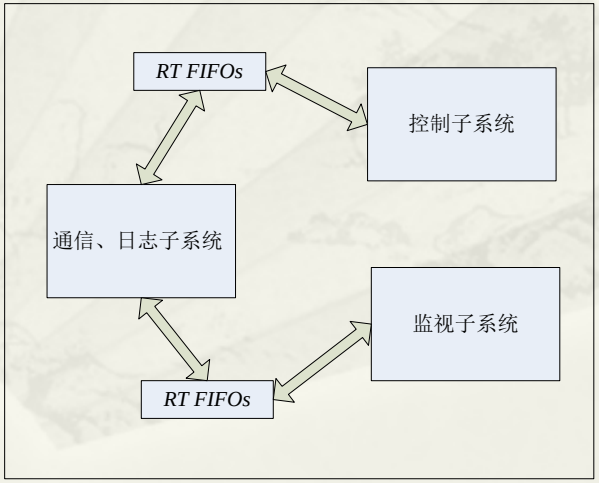
* 15.2.1 MAX下设置远程系统IP
* 15.2.2 给远程设备安装软件
* 15.2.3 识别远程设备
* 15.2.4 建立RT工程





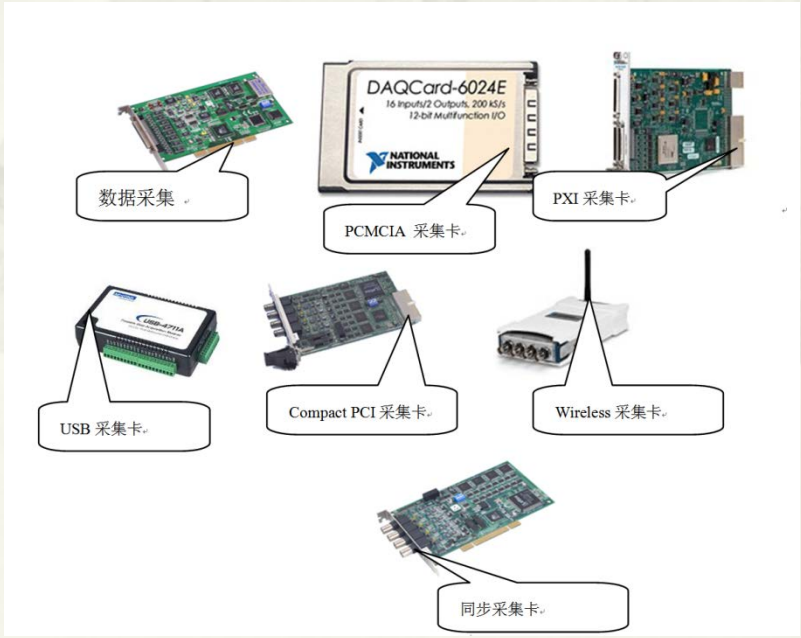
15.3 Real-time高级编程及技巧

- 15.3.1 实时操作系统下的LabVIEW不支持的特性
- 15.3.2 实时操作系统下的多线程
- 15.3.3 实时系统中的时间确定性实现
- 15.3.4 实时系统中线程间通信
- 15.3.5 实时控制系统的软件架构及其评测



第16章 LabVIEW与数据采集

- 16.1 数据采集的一些基本概念
- 16.2 数据采集卡
- 16.3 采样定理
- 16.4 如何降低系统噪声和提高精度
- 16.5 如何选购采集卡
- 16.6 软件采集基础
- 16.7 基于NI-DAQmx的高级编程

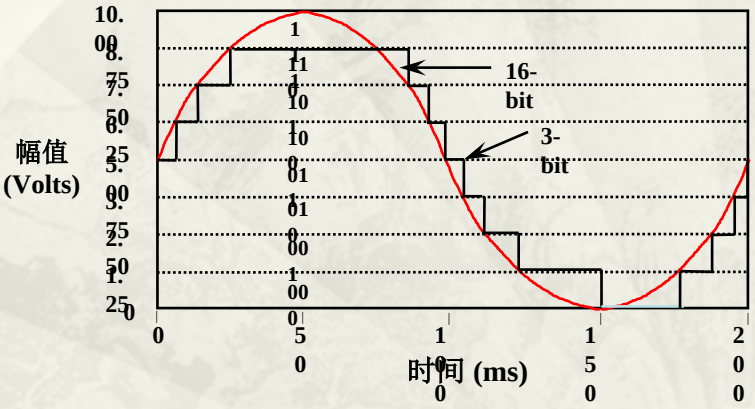


16.1 数据采集的一些基本概念

* 16.1.1 信号
* 16.1.2 传感器
* 16.1.3 信号处理

16.2 数据采集卡

- 16.2.1 数据采集卡定义及其分类
- 16.2.2 多功能数据采集卡原理图
- 16.2.3 数据采集卡的关键参数
- 16.2.4 数据采集卡与信号接地



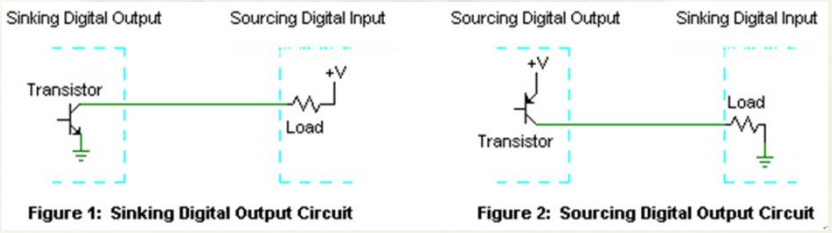
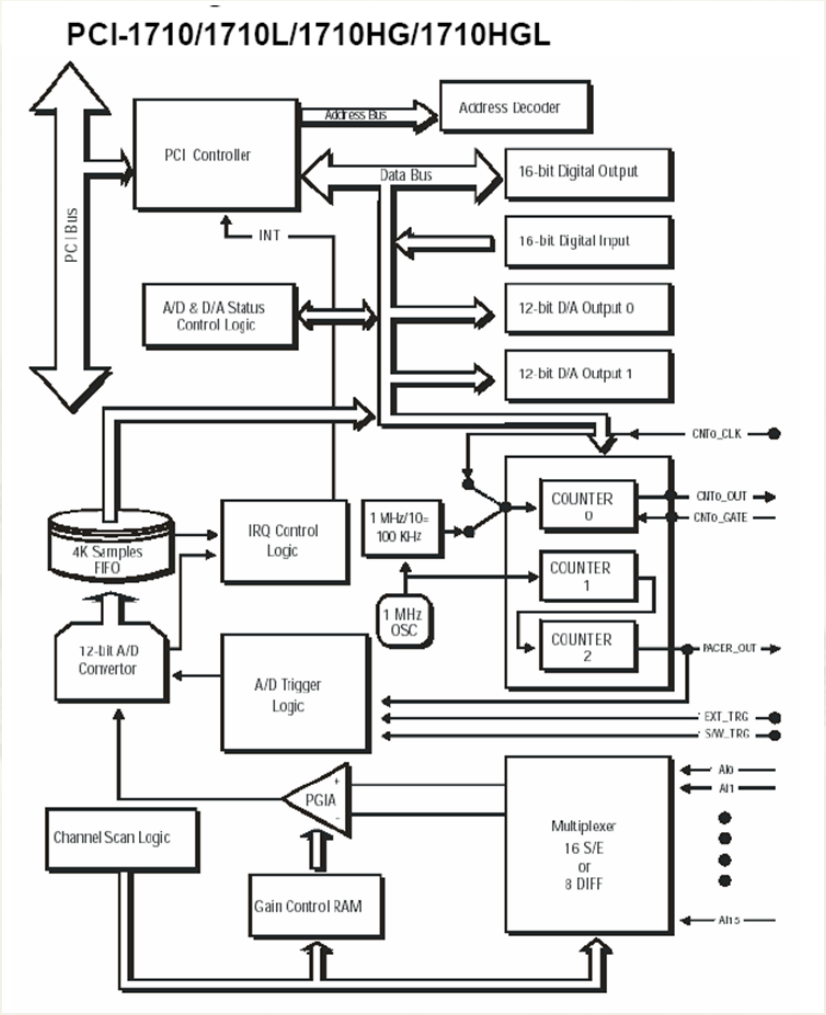


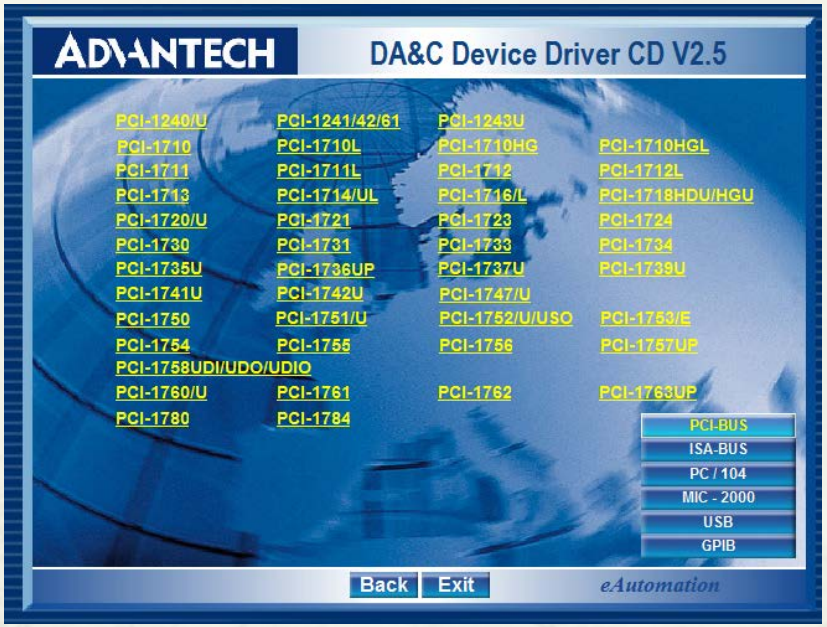
Figure 1: Sinking Digital Output Circuit

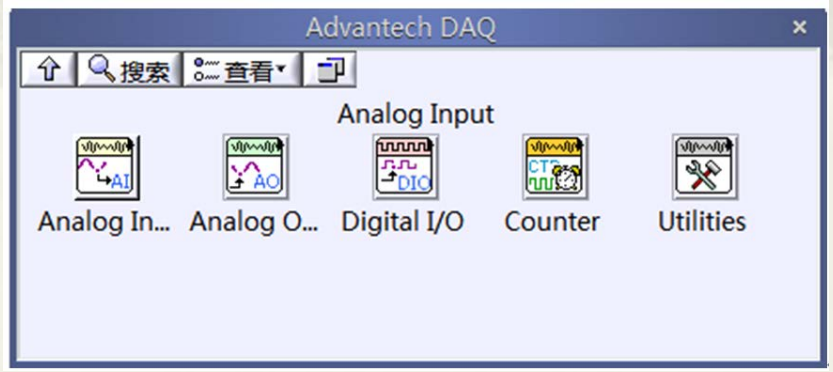
Figure 2: Sourcing Digital Output Circuit



16.6 软件采集基础

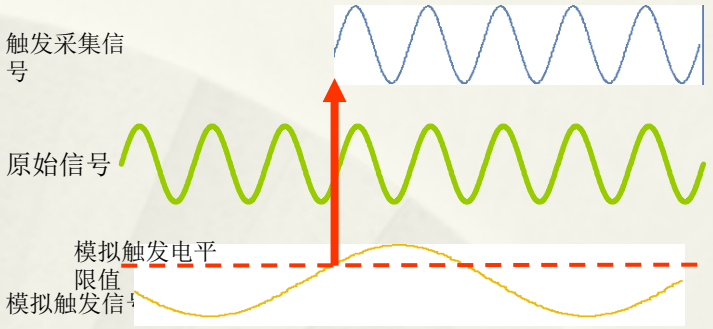
- 16.6.1 采集系统的安装
- 16.6.2 NI采集卡的常用函数
- 16.6.3 研华采集卡的常用函数

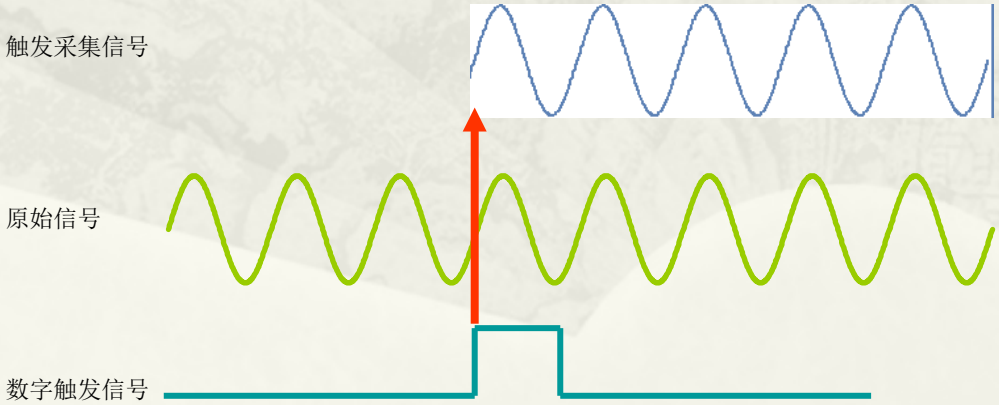


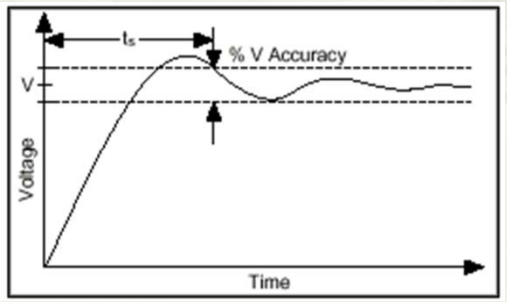


16.7 基于NI-DAQmx的高级编程

- 16.7.1 触发信号
- 16.7.2 采集系统时钟
- 16.7.3 多板卡之间的同步采集
- 16.7.4 完整波形输出
- 16.7.5 并行结构进行采集
- 16.7.6 硬件通过TIME LOOP触发运行
- 16.7.7 依靠DAQmx的事件编写面向事件驱动的程
- 16.7.8 选择合适的读取策略
- 16.7.9 使用DAQmx控制任务安全中止采集
- 16.7.10计数器/定时器及其应用

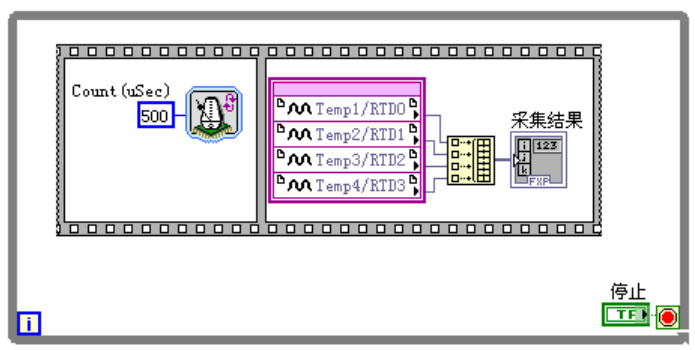






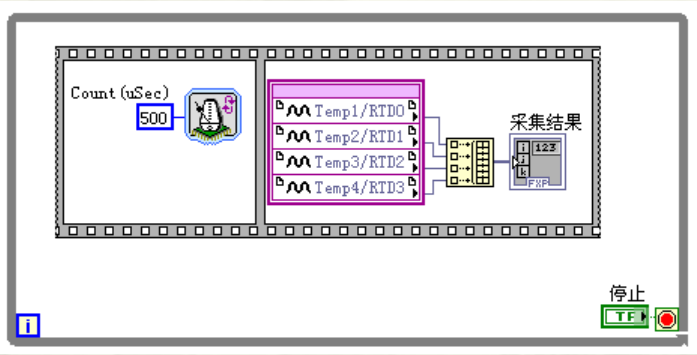
第17章 FPGA工具包

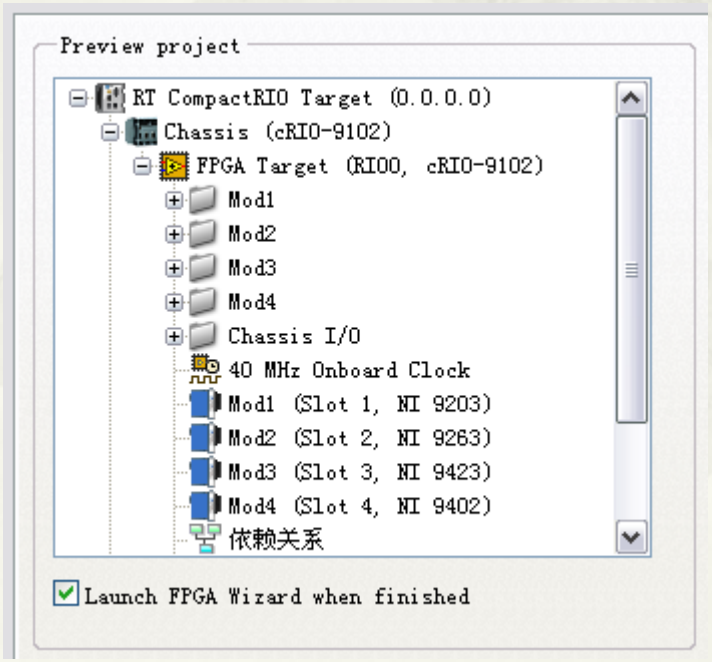
- 17.1 FPGA的基本概念与CRIO的组成
- 17.2 FPGA编程
- 17.3 FPGA与RT之间的数据交换
- 17.4 SPARTAN-3E开发板

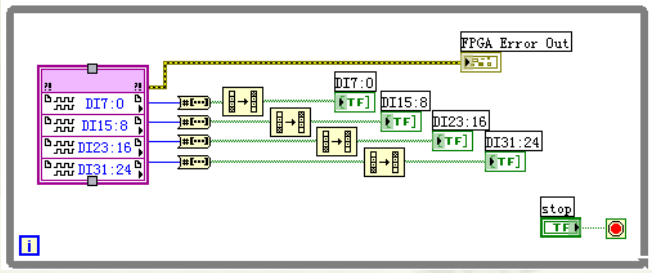


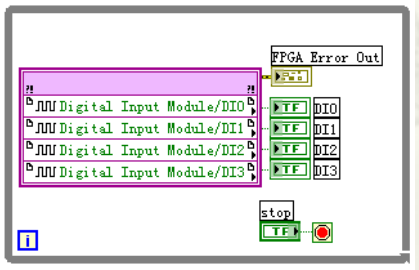
17.1 FPGA的基本概念与CRIO的组成

* 17.1.1 FPGA的基本概念
* 17.1.2 CRIO的构成
* 17.1.3 构建FPGA项目



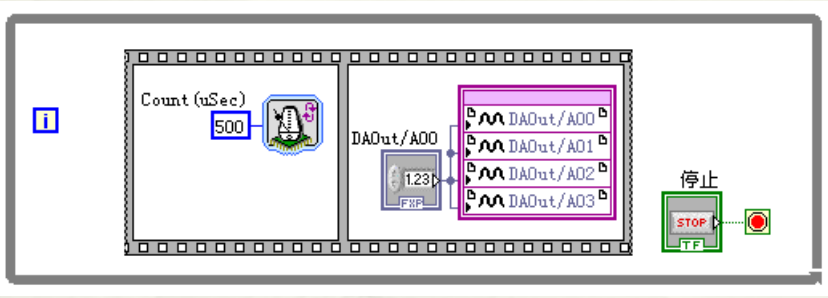


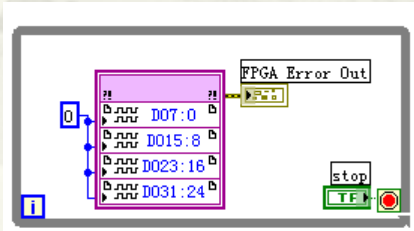


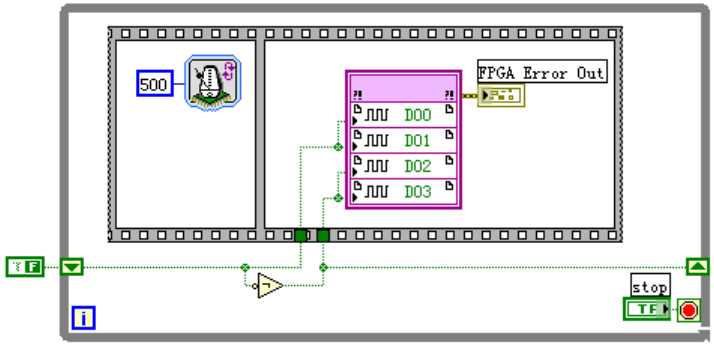


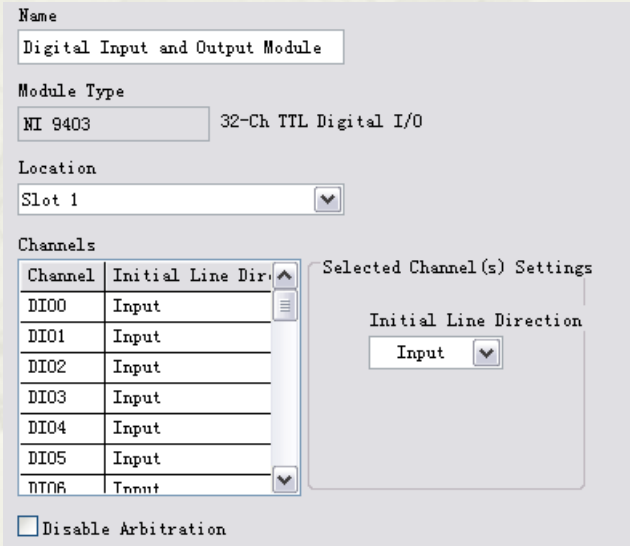
17.2 FPGA编程

- 17.2.1 FPGA基本IO之模拟量输入输出
- 17.2.2 FPGA基本IO之数字量输入输出
- 17.2.3 FPAG定时、时钟与分频
- 17.2.4 FPGA计数器应用
- 17.2.5 触发与外部时钟循环
- 17.2.6 FPGA常用函数
- 17.2.7 FPGA多线程与线程间数据交换
- 17.2.8 FPGA IP Core



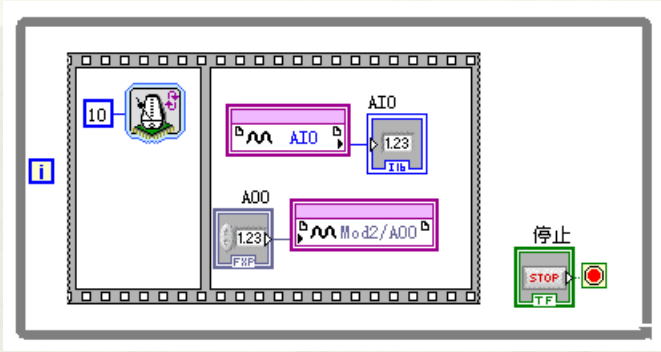


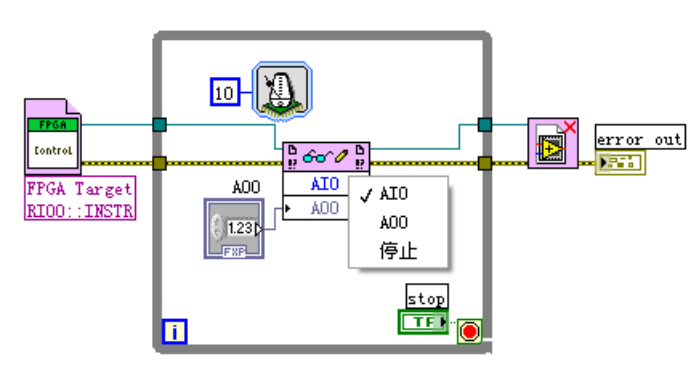


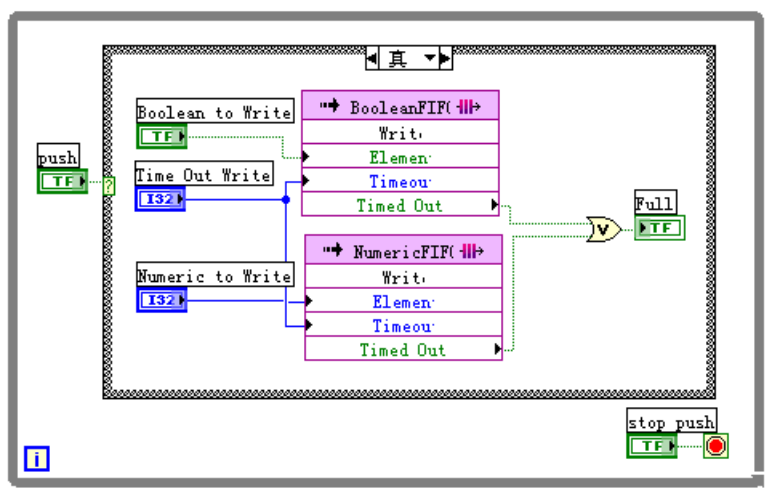


17.3 FPGA与RT之间的数据交换

- 17.3.1 读写控件方式
- 17.3.2 中断
- 17.3.3 FIFO
- 17.3.4 扫描方式
- 17.3.5专用C模块
- 17.3.6 FPGA程序的优化







17.4 SPARTAN-3E开发板

- 17.4.1 SPARTAN-3E简介
- 17.4.2 建立SPARTAN-3E FPGA项目
- 17.4.3 编译FPGA程序