

DRVI/DRLink

可重组虚拟仪器平台

用户使用手册



前言

现代科技的发展，使得改革传统教学方式迫在眉睫！通过增加实验和培训课程，重点培养学生的创造能力和实际操作能力是教学改革的重要内容之一。

作为教学改革新浪潮中的一名成员，德普施科技凭借自己多年来在远程教育、网络化测控、自动控制和机电一体化等领域进行科研和教学实践所取得的丰硕成果，结合国内外多家名校的成功技术及经验，始终致力于开放式实验教学设备的研究与开发工作，并取得了骄人的成绩。

针对教育部提出的进一步提高高等学校实验室建设和管理水平，推进实验教学改革，保证教学质量的思想，德普施科技最新推出面向高校的 DRLab (Dynamic Reconfigurable Lab) 创新实验室系统，DRLab 系统包括：自主研发的 DRVI 可重组虚拟仪器平台、DRLink 可重组计算机控制平台、DRScene 机电设备运动控制仿真平台、各种传感器及实验设备，提供从实验对象、信号获取、信号调理、数据分析及处理的一整套网络化创新实验室解决方案，可广泛适用于机械、自动控制、控制工程、机电一体化、电机等专业，使学校能根据应用需求快速组建一个高水平的实验室，提升学科的建设水平。

目录

前言	1
第一章、基本功能介绍	3
1.1 DRVI/DRLink 简介	3
1.2 系统配置要求	3
1.3 安装 DRVI/DRLink 软件	4
1.4 运行 DRVI/DRLink	7
1.5 软件卸载	8
第二章、DRVI/DRLink 操作方法	9
2.1 前面板	9
2.2 软件芯片表	9
2.3 IC 资源脚本文件	10
2.4 软件平台菜单	19
2.5 快捷工具条	25
第三章、DRVI/DRLink 虚拟仪器设计	27
3.1 软件总线的概念	27
3.2 插接软件芯片	29
3.3 软件芯片连线	30
3.4 软件芯片屏幕布局	31
3.5 软件芯片一览表	32
第四章、DRVI/DRLink 功能扩展	41
4.1 Signal VBScript 语言	41
4.2 用 Signal VBScript 编制软件芯片插件	53
4.3 用 Visual C++编制动态链接库型插件	61
第五章、相关工具使用说明	63
5.1 DRVI/DRLink 传感器定标曲线拟合工具使用说明	63
5.2 DRVI/DRLink 数字滤波器设计工具使用说明	64
5.3 DRVI/DRLink 帮组信息浏览器使用说明	68
5.4 AVI 教学短片制作工具使用说明	69

第一章、基本功能介绍

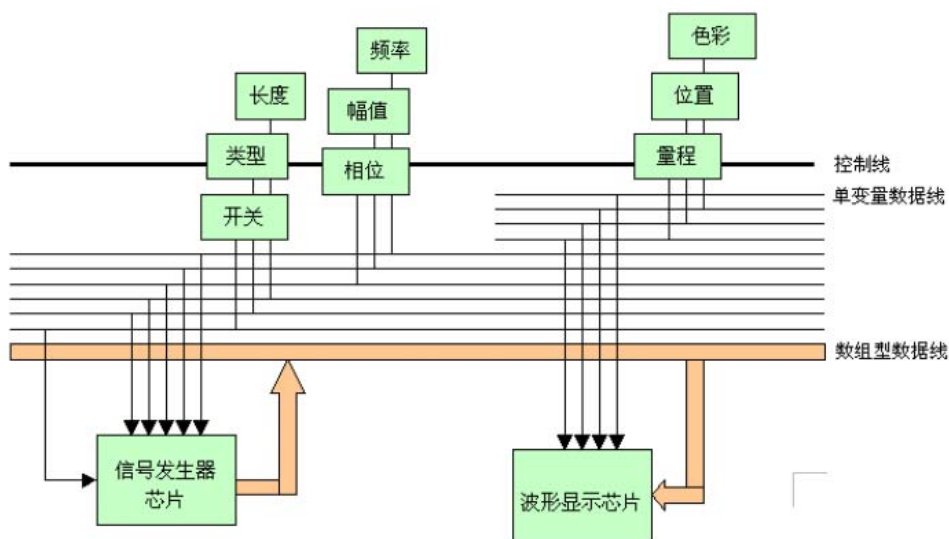
1.1 DRVI/DRLink 简介

DRVI/DRLink 是基于软件总线和芯片结构的快速可重组实验开发平台。DRVI/DRLink 是采用软件总线开放结构和 COM/DCOM 组件的即插即用特性来设计的具有计算机硬件模块化组装特点的面向用户的可在线编程、调试和重组的新型计算机自动控制实验系统，可广泛应用于科学实验、远程教育和实验教学等诸多领域。

DRVI/DRLink 是按照制造业模式设计的新型计算机自动控制应用软件开发平台，其设计思想是按照 PC 机、小轿车等的装配思想，用单独的部件去实现每一项功能，然后再用一个底盘、或总线，把完成各项功能的部件组装起来，快速形成一个产品。DRVI 集成了大多数的信号分析算法，DRVI/DRLink 集成了大多数的控制算法。

用户应用软件不是用程序代码编制出来的，而是用**软件模块**配置出来的。

DRVI/DRLink 的主体为一个带软件控制线和数据线的软主板，其上可插接数字 PID、一阶系统环节、二阶系统环节和、软仪表盘、软信号发生器、软波形显示芯片等软件芯片组，并能与 A/D 卡、D/A 卡、I/O 卡等信号采集硬件进行组合与连接，构成一个能根据应用需求快速重组的计算机自动控制系统。



1.2 系统配置要求

DRVI/DRLink 能够运行在任何支持 Windows9x 的系统上。系统至少应具有 32MB 的 RAM；要是系统能够有效的运行，系统设备应具有 64MB 以上的 RAM。

DRVI/DRLink 能够运行在支持 WindowsNT4.0 或更新的 80x86 计算机系统上(支持 Windows7)。系统至少应具有 32MB 的 RAM 和一个 486/DX 以上的处理器；要使系统能够有效的运行设备应具有 64MB 以上的 RAM 和 Pentium 处理器。

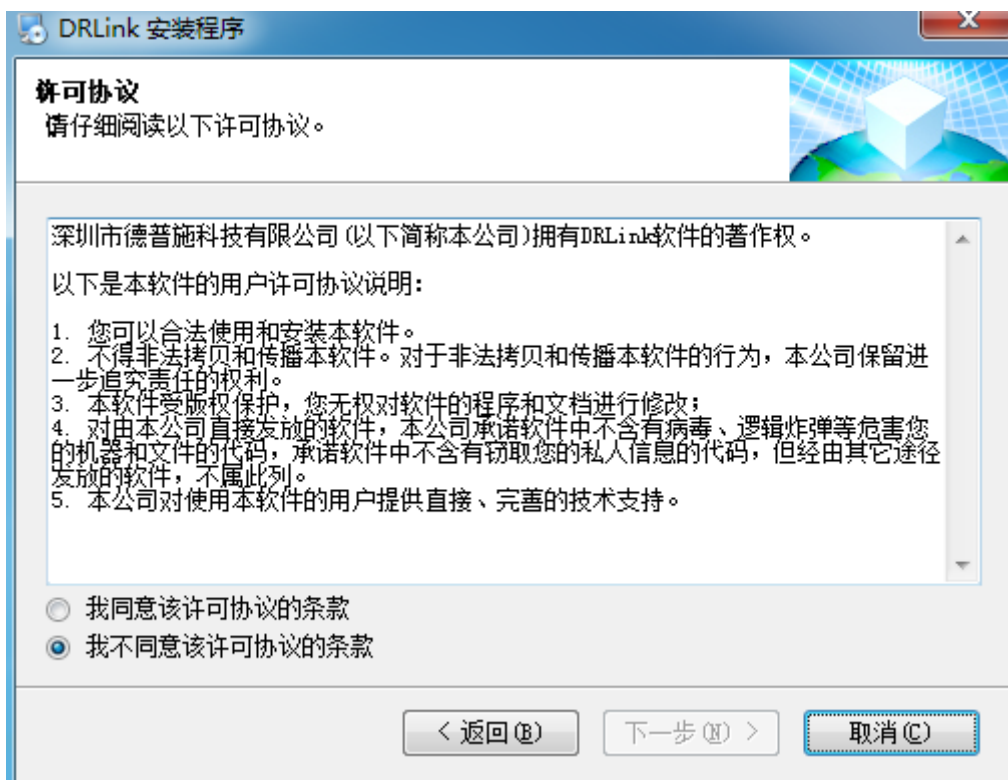
DRVI/DRLink 能够运行在支持 Windows2000 的 80x86 计算机系统上。系统至少应具有 32MB 的 RAM 和一个 Pentium 处理器；要使系统能够有效的运行，系统设备应具有 64MB 以上的 RAM。

1.3 安装 DRVI/DRLink 软件

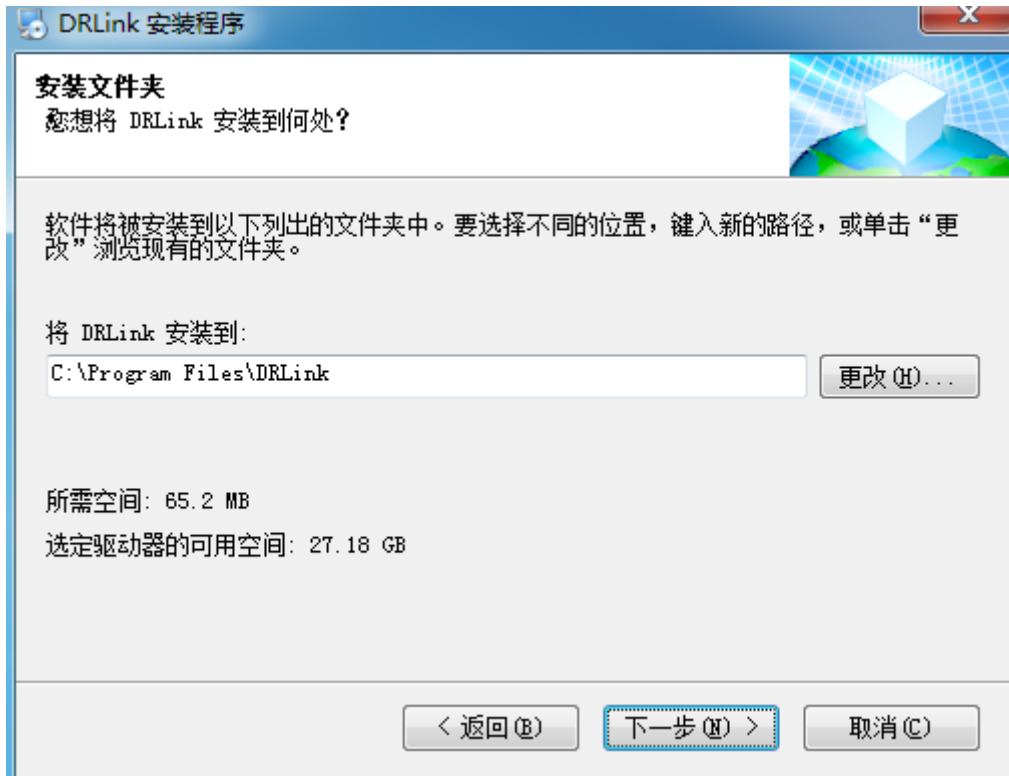
- 1、将 DRVI/DRLink 安装光盘放入光盘驱动器。
- 2、双击打开光盘目录中的 DRVISetup.exe/DRLinkSetup.exe 文件，出现下面的安装界面：



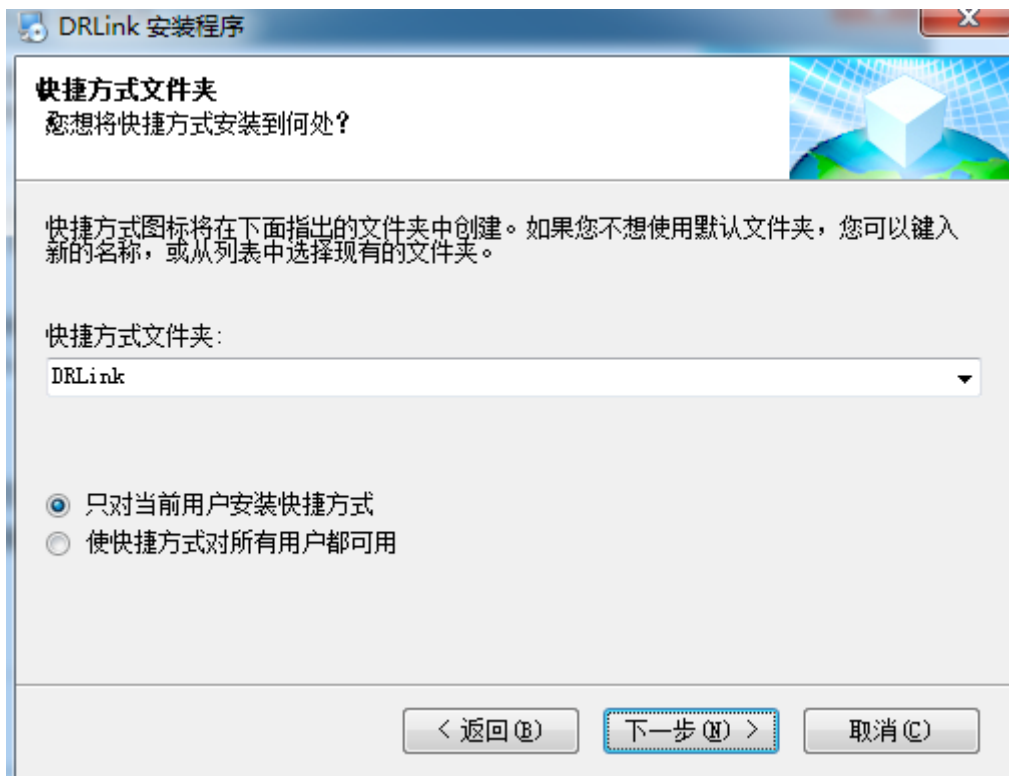
- 3、点击“下一步”，出现下面的“软件许可协议”界面



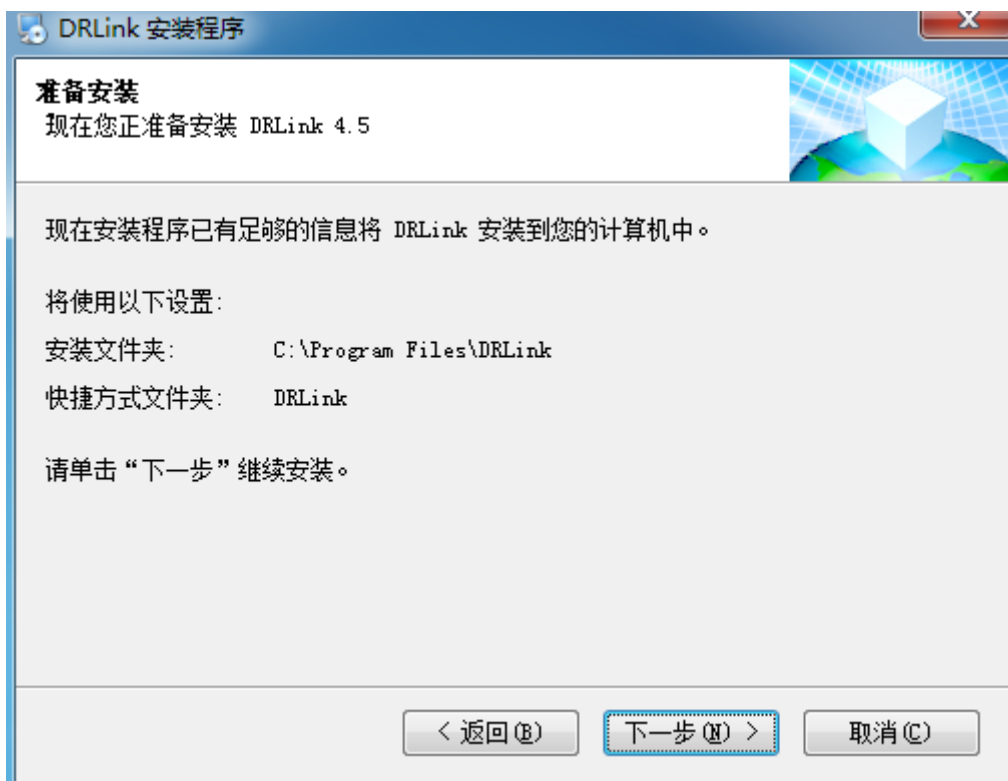
- 4、如果不接受许可协议，则点击“取消”，退出安装程序；如果接受许可协议，则首先点击：“我接受许可协议”，然后点击“下一步”，出现下面的安装界面。



5、软件缺省的安装目录出现在栏位中，如果您要将软件安装到其它位置，可以直接在栏中输入安装路径，或者点击“更改”进行选择。点击“下一步”，出现如下安装界面。



6、软件缺省的快捷程序组出现在输入栏位中，如果您要改变快捷程序组的名称，请在输入栏中填写您所希望的名称。选择是否创建对所有用户，就会对所有用户的开始菜单中添加快捷方式，直接点击“下一步”。



7、点击“下一步”，进行安装，出现安装进度界面。



8、安装完成后，出现“DRVI/DRLink 安装完成”界面，点击完成按钮完成整个 DRVI/DRLink4.5 程序安装。



1.4 运行 DRVI/DRLink

打开 DRVI/DRLink 有以下三种方式启动：

- 1) 在 Windows 桌面上点击图标 / 
- 2) 在开始菜单程序选项中选择 DRLink /DRVI  运行
- 3) 在 DRVI/DRLink 的软件安装目录下双击应用程序 DRLink/ DRVI 运行



1.5 软件卸载

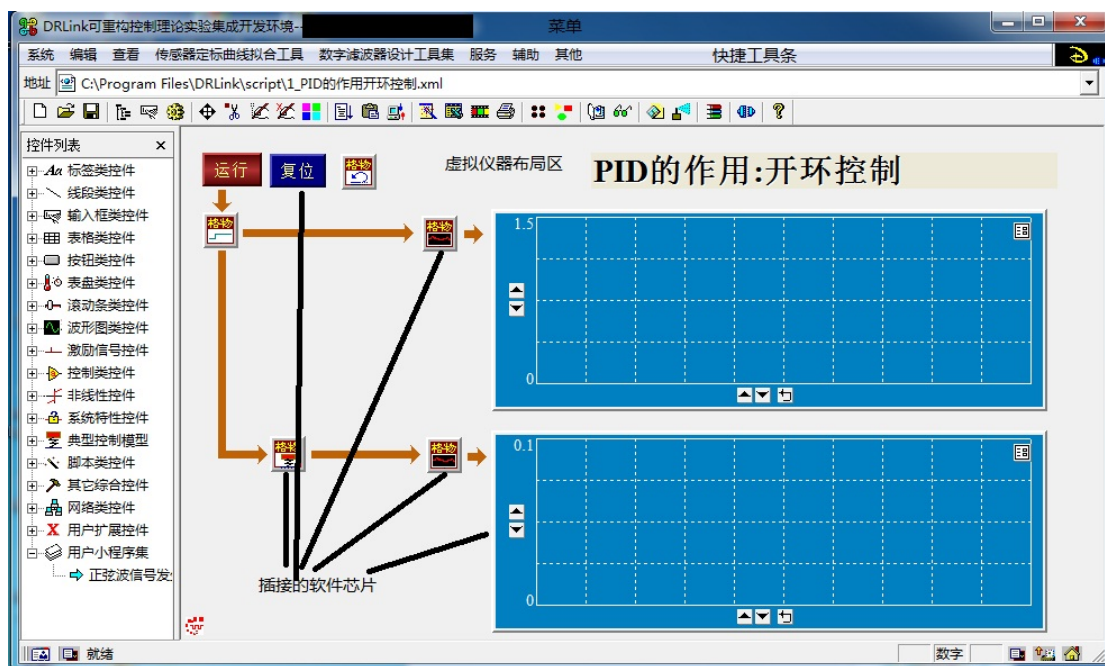
如果想卸载 DRLink/DRVI 应用软件，只需在 Windows 的程序文件夹中，找到 DRLink/DRVI 文件夹，点击“卸载 DRLink/DRVI”图标即可卸载本软件。

第二章、DRVI/DRLink 操作方法


2.1 前面板

DRVI/DRLink 前面板是一个采用 COM 组件容器程序设计的软件总线结构的可重构虚拟仪器可视化开发/运行一体化平台。该操作平台提供了虚拟仪器软件总线底板、软件芯片插件组、嵌入式 Web 服务器、VBScript 脚本语言等功能支持。

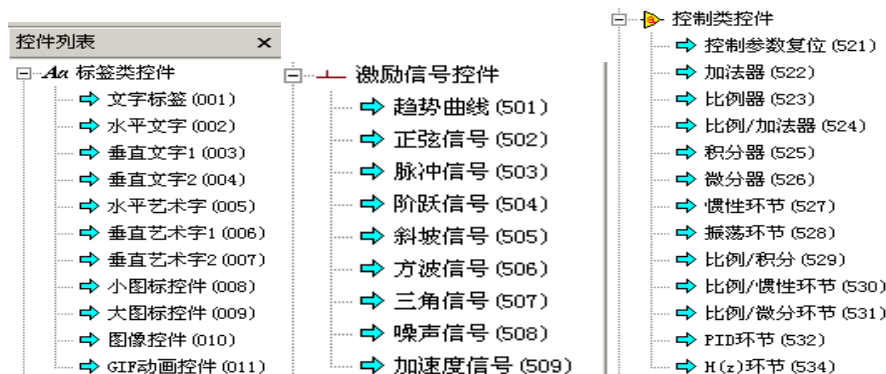
用户可以利用前面板提供的菜单、工具条、软件芯片表，在前面板上可视化插接虚拟仪器软件芯片，快速进行虚拟仪器设计。



2.2 软件芯片表

从 DRVI/DRLink 快捷工具条中选择 ，都可以弹出**软件芯片表**，如下图所示。软件芯片表中的虚拟仪器软件芯片采用 COM 组件技术设计，支持二进制代码直接复用，可直接插接在 DRVI/DRLink 前面板代表的软件总线上使用。

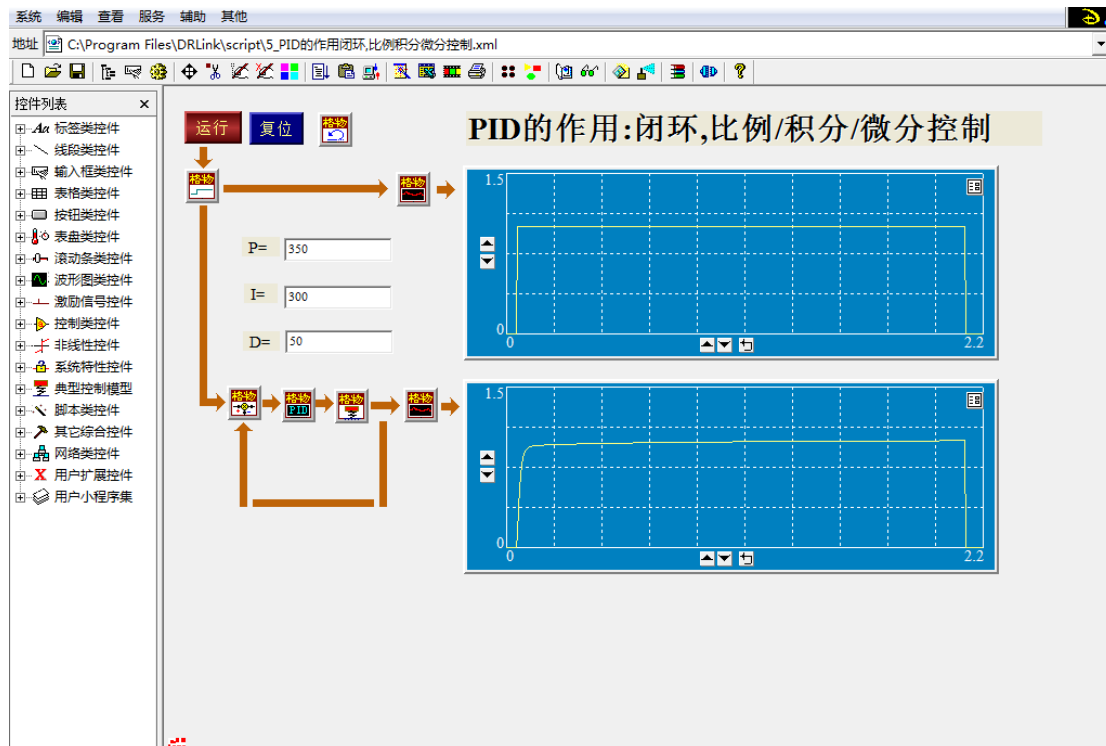
点击下图中的软件芯片就可以再 DRVI/DRLink 前面板达标的软件总线上插接一片该软件芯片。



2.3 IC 资源脚本文件

DRVI/DRLink 采用我们提出的基于 XML 的虚拟仪器标记语言来描述、记录和保存虚拟仪器设计结果。在 DRVI/DRLink 中读入一个 IC 资源脚本文件就生成一个虚拟仪器。脚本文件在用户可视化插接软件芯片和修改芯片属性窗参数时自动生成，一般不直接对脚本文件进行编辑。

例如，下面是可视化设计的“PID 的作用:闭环,比例/积分/微分控制”实验的屏幕拷贝



下面是一个虚拟仪器脚本文件样例：

```
<?xml version="1.0" encoding="GB2312" ?>
<DRVI4.1>
  <FontIC  >
    <ICName value="1280991417"> </ICName>
    <X0 value="279"> </X0>
    <Y0 value="23"> </Y0>
    <Width value="532"> </Width>
    <High value="32"> </High>
    <Caption value="PID 的作用:闭环,比例/积分/微分控制"> </Caption>
    <FGColor value="0"> </FGColor>
    <ShadowColor value="16777215"> </ShadowColor>
    <BGColor value="14215660"> </BGColor>
    <FontWidthHigh value="0.700000"> </FontWidthHigh>
    <Direction value="0"> </Direction>
    <FontProperty value="0"> </FontProperty>
    <ShadowEffect value="1"> </ShadowEffect>
    <BusDataLine value="-1"> </BusDataLine>
  </FontIC>
  <OnOffButtonIC  >
    <ICName value="1280991482"> </ICName>
```

```

<X0 value="19"> </X0>
<Y0 value="23"> </Y0>
<Width value="53"> </Width>
<High value="30"> </High>
<Caption value="运行/停止"> </Caption>
<BGColor value="128"> </BGColor>
<FGColor value="8454143"> </FGColor>
<Value value="0"> </Value>
<BusDataLine value="2"> </BusDataLine>
<Font value="11, 1, 0, 1"> </Font>
</OnOffButtonIC>
<ButtonIC >
  <ICName value="1280991535"> </ICName>
  <X0 value="79"> </X0>
  <Y0 value="24"> </Y0>
  <Width value="50"> </Width>
  <High value="30"> </High>
  <Caption value="复位"> </Caption>
  <BGColor value="8454143"> </BGColor>
  <FGColor value="8388608"> </FGColor>
  <DrivenIC value=""> </DrivenIC>
  <BusDataLine value="1"> </BusDataLine>
  <Font value="11, 1, 0, 0"> </Font>
</ButtonIC>
<CzGenerator >
  <P0 value="1280991573" > </P0>
  <P1 value="19, 76, 187, 220, 32, 32" > </P1>
  <P2 value="2, 10, 3, -1, -1, -1" > </P2>
  <P3 value="-1, -1, -1, -1, 2" > </P3>
  <P4 value="200. 000000, 1. 000000, 10. 000000, 0. 000000, 0. 000000" > </P4>
</CzGenerator>
<ArrowIC >
  <ICName value="1280991623"> </ICName>
  <X0 value="55"> </X0>
  <Y0 value="86"> </Y0>
  <ArrowLength value="155"> </ArrowLength>
  <ArrowWidth value="17"> </ArrowWidth>
  <Color value="484286"> </Color>
  <Direction value="0"> </Direction>
  <StartType value="0"> </StartType>
  <EndType value="1"> </EndType>
</ArrowIC>
<TrendIC >
  <P0 value="1280991675" > </P0>

```

```

<P1 value="214, 79, 172, 293, 32, 32" > </P1>
<P2 value="440, 4, 3, -1" > </P2>
<P3 value="200.000000" > </P3>
</TrendIC>
<ArrowIC >
  <ICName value="1280991701"> </ICName>
  <X0 value="252"> </X0>
  <Y0 value="87"> </Y0>
  <ArrowLength value="20"> </ArrowLength>
  <ArrowWidth value="17"> </ArrowWidth>
  <Color value="484286"> </Color>
  <Direction value="0"> </Direction>
  <StartType value="0"> </StartType>
  <EndType value="1"> </EndType>
</ArrowIC>
<WaveIC >
  <ICName value="1280991741"> </ICName>
  <X0 value="276"> </X0>
  <Y0 value="73"> </Y0>
  <LengthOfShow value="495"> </LengthOfShow>
  <High value="180"> </High>
  <BgColor value="12615680"> </BgColor>
  <FGColor value="16777215"> </FGColor>
  <CurveColor value="8454143"> </CurveColor>
  <Maximum value="1.500000"> </Maximum>
  <Minimum value="0.000000"> </Minimum>
  <Left value="40"> </Left>
  <Type value="0"> </Type>
  <BusArrayLine value="4"> </BusArrayLine>
</WaveIC>
<WaveIC >
  <ICName value="1280991921"> </ICName>
  <X0 value="276"> </X0>
  <Y0 value="270"> </Y0>
  <LengthOfShow value="495"> </LengthOfShow>
  <High value="180"> </High>
  <BgColor value="12615680"> </BgColor>
  <FGColor value="16777215"> </FGColor>
  <CurveColor value="8454143"> </CurveColor>
  <Maximum value="1.500000"> </Maximum>
  <Minimum value="0.000000"> </Minimum>
  <Left value="40"> </Left>
  <Type value="0"> </Type>
  <BusArrayLine value="11"> </BusArrayLine>

```

```

</WaveIC>
<TrendIC >
    <P0 value="1280991954" > </P0>
    <P1 value="221, 278, 203, 306, 32, 32" > </P1>
    <P2 value="440, 11, 9, -1" > </P2>
    <P3 value="200.000000" > </P3>
</TrendIC>
<ArrowIC >
    <ICName value="1280991967"> </ICName>
    <X0 value="91"> </X0>
    <Y0 value="284"> </Y0>
    <ArrowLength value="20"> </ArrowLength>
    <ArrowWidth value="17"> </ArrowWidth>
    <Color value="484286"> </Color>
    <Direction value="0"> </Direction>
    <StartType value="0"> </StartType>
    <EndType value="1"> </EndType>
</ArrowIC>
<ArrowIC >
    <ICName value="1280992075"> </ICName>
    <X0 value="40"> </X0>
    <Y0 value="283"> </Y0>
    <ArrowLength value="20"> </ArrowLength>
    <ArrowWidth value="17"> </ArrowWidth>
    <Color value="484286"> </Color>
    <Direction value="0"> </Direction>
    <StartType value="0"> </StartType>
    <EndType value="1"> </EndType>
</ArrowIC>
<ArrowIC >
    <ICName value="1280992079"> </ICName>
    <X0 value="28"> </X0>
    <Y0 value="110"> </Y0>
    <ArrowLength value="186"> </ArrowLength>
    <ArrowWidth value="16"> </ArrowWidth>
    <Color value="484286"> </Color>
    <Direction value="1"> </Direction>
    <StartType value="0"> </StartType>
    <EndType value="0"> </EndType>
</ArrowIC>
<ADDIC >
    <P0 value="1280992195" > </P0>
    <P1 value="58, 276, 35, 368, 32, 32, 0, 8" > </P1>
    <P2 value="3, 9, -1, -1, -1, -1, -1, -1" > </P2>

```

```

    <P3 value="1.000000,1.000000,1.000000,1.000000" > </P3>
</ADDIC>
<Czpid >
    <P0 value="1280992256" > </P0>
    <P1 value="108,278,118,481,32,32" > </P1>
    <P2 value="1,10,8,5,6,7,-1" > </P2>
    <P3 value="350.000000,300.000000,50.000000,200.000000" > </P3>
</Czpid>
<CzVibrate >
    <P0 value="1280992302" > </P0>
    <P1 value="157,279,112,460,32,32,1,1" > </P1>
    <P2 value="-1,10,9,-1,-1,-1,-1,-1" > </P2>
    <P3 value="200.000000" > </P3>
    <P4 value="1," > </P4>
    <P5 value="10,0," > </P5>
    <P6 value="20,0," > </P6>
</CzVibrate>
<ArrowIC >
    <ICName value="1280992479"> </ICName>
    <X0 value="140"> </X0>
    <Y0 value="284"> </Y0>
    <ArrowLength value="20"> </ArrowLength>
    <ArrowWidth value="17"> </ArrowWidth>
    <Color value="484286"> </Color>
    <Direction value="0"> </Direction>
    <StartType value="0"> </StartType>
    <EndType value="1"> </EndType>
</ArrowIC>
<ArrowIC >
    <ICName value="1280992569"> </ICName>
    <X0 value="256"> </X0>
    <Y0 value="287"> </Y0>
    <ArrowLength value="20"> </ArrowLength>
    <ArrowWidth value="17"> </ArrowWidth>
    <Color value="484286"> </Color>
    <Direction value="0"> </Direction>
    <StartType value="0"> </StartType>
    <EndType value="1"> </EndType>
</ArrowIC>
<ArrowIC >
    <ICName value="1280992611"> </ICName>
    <X0 value="191"> </X0>
    <Y0 value="286"> </Y0>
    <ArrowLength value="30"> </ArrowLength>

```



```

    <ArrowWidth value="17"> </ArrowWidth>
    <Color value="484286"> </Color>
    <Direction value="0"> </Direction>
    <StartType value="0"> </StartType>
    <EndType value="1"> </EndType>
  </ArrowIC>
  <ArrowIC >
    <ICName value="1280992670"> </ICName>
    <X0 value="65"> </X0>
    <Y0 value="309"> </Y0>
    <ArrowLength value="80"> </ArrowLength>
    <ArrowWidth value="17"> </ArrowWidth>
    <Color value="484286"> </Color>
    <Direction value="1"> </Direction>
    <StartType value="1"> </StartType>
    <EndType value="0"> </EndType>
  </ArrowIC>
  <ArrowIC >
    <ICName value="1280992673"> </ICName>
    <X0 value="194"> </X0>
    <Y0 value="309"> </Y0>
    <ArrowLength value="80"> </ArrowLength>
    <ArrowWidth value="17"> </ArrowWidth>
    <Color value="484286"> </Color>
    <Direction value="1"> </Direction>
    <StartType value="0"> </StartType>
    <EndType value="0"> </EndType>
  </ArrowIC>
  <ArrowIC >
    <ICName value="1280992675"> </ICName>
    <X0 value="77"> </X0>
    <Y0 value="376"> </Y0>
    <ArrowLength value="120"> </ArrowLength>
    <ArrowWidth value="17"> </ArrowWidth>
    <Color value="484286"> </Color>
    <Direction value="0"> </Direction>
    <StartType value="0"> </StartType>
    <EndType value="0"> </EndType>
  </ArrowIC>
  <LabelIC >
    <ICName value="1280992823"> </ICName>
    <X0 value="72"> </X0>
    <Y0 value="139"> </Y0>
    <Width value="35"> </Width>

```

```

    <High value="18"> </High>
    <Caption value="P="> </Caption>
    <BGColor value="14215660"> </BGColor>
    <FGColor value="0"> </FGColor>
    <BusDataLine value="-1"> </BusDataLine>
    <Font value="11, 0, 0, 0"> </Font>
    <Direction value="0"> </Direction>
</LabelIC>
<LabelIC >
    <ICName value="1280992826"> </ICName>
    <X0 value="74"> </X0>
    <Y0 value="182"> </Y0>
    <Width value="31"> </Width>
    <High value="18"> </High>
    <Caption value="I="> </Caption>
    <BGColor value="14215660"> </BGColor>
    <FGColor value="0"> </FGColor>
    <BusDataLine value="-1"> </BusDataLine>
    <Font value="11, 0, 0, 0"> </Font>
    <Direction value="0"> </Direction>
</LabelIC>
<LabelIC >
    <ICName value="1280992830"> </ICName>
    <X0 value="73"> </X0>
    <Y0 value="226"> </Y0>
    <Width value="34"> </Width>
    <High value="18"> </High>
    <Caption value="D="> </Caption>
    <BGColor value="14215660"> </BGColor>
    <FGColor value="0"> </FGColor>
    <BusDataLine value="-1"> </BusDataLine>
    <Font value="11, 0, 0, 0"> </Font>
    <Direction value="0"> </Direction>
</LabelIC>
<EDITIC >
    <ICName value="1280992892"> </ICName>
    <X0 value="111"> </X0>
    <Y0 value="140"> </Y0>
    <Width value="100"> </Width>
    <High value="22"> </High>
    <BusDataLine value="5"> </BusDataLine>
    <ShowValue value="350"> </ShowValue>
    <Font value="11, 0, 0"> </Font>
</EDITIC>

```

```

<EDITIC >
    <ICName value="1280992900"> </ICName>
    <X0 value="111"> </X0>
    <Y0 value="184"> </Y0>
    <Width value="100"> </Width>
    <High value="22"> </High>
    <BusDataLine value="6"> </BusDataLine>
    <ShowValue value="300"> </ShowValue>
    <Font value="11, 0, 0"> </Font>
</EDITIC>
<EDITIC >
    <ICName value="1280992906"> </ICName>
    <X0 value="112"> </X0>
    <Y0 value="225"> </Y0>
    <Width value="100"> </Width>
    <High value="22"> </High>
    <BusDataLine value="7"> </BusDataLine>
    <ShowValue value="50"> </ShowValue>
    <Font value="11, 0, 0"> </Font>
</EDITIC>
<ArrowIC >
    <ICName value="1280993054"> </ICName>
    <X0 value="28"> </X0>
    <Y0 value="56"> </Y0>
    <ArrowLength value="20"> </ArrowLength>
    <ArrowWidth value="17"> </ArrowWidth>
    <Color value="484286"> </Color>
    <Direction value="1"> </Direction>
    <StartType value="0"> </StartType>
    <EndType value="1"> </EndType>
</ArrowIC>
<ZeroIC >
    <P0 value="1280993334" > </P0>
    <P1 value="142, 24, 152, 512, 32, 32, 1" > </P1>
</ZeroIC>
<ZLineIC >
    <ICName value="1280993342"> </ICName>
    <Parameters value="1, 1280991535, 1, 1280993334, 1, 1337293"> </Parameters>
</ZLineIC>
<ZLineIC >
    <ICName value="1280993354"> </ICName>
    <Parameters value="2, 1280991482, 1, 1280991573, 9, 1337293"> </Parameters>
</ZLineIC>
<ZLineIC >

```

```

    <ICName value="1280993363"> </ICName>
    <Parameters value="3,1280991573,1,1280991675,2,1337293"> </Parameters>
</ZLineIC>
<ZLineIC >
    <ICName value="1280993367"> </ICName>
    <Parameters value="4,1280991675,1,1280991741,1,1337293"> </Parameters>
</ZLineIC>
<ZLineIC >
    <ICName value="1280993510"> </ICName>
    <Parameters value="5,1280992892,1,1280992256,3,1337293"> </Parameters>
</ZLineIC>
<ZLineIC >
    <ICName value="1280993517"> </ICName>
    <Parameters value="6,1280992900,1,1280992256,4,1337293"> </Parameters>
</ZLineIC>
<ZLineIC >
    <ICName value="1280993522"> </ICName>
    <Parameters value="7,1280992906,1,1280992256,5,1337293"> </Parameters>
</ZLineIC>
<ZLineIC >
    <ICName value="1280993574"> </ICName>
    <Parameters value="3,1280991573,1,1280992195,2,1337293"> </Parameters>
</ZLineIC>
<ZLineIC >
    <ICName value="1280993583"> </ICName>
    <Parameters value="8,1280992195,1,1280992256,2,1337293"> </Parameters>
</ZLineIC>
<ZLineIC >
    <ICName value="1280993713"> </ICName>
    <Parameters value="10,1280992256,1,1280992302,2,1337293"> </Parameters>
</ZLineIC>
<ZLineIC >
    <ICName value="1280994098"> </ICName>
    <Parameters value="9,1280992302,3,1280991954,2,1337293"> </Parameters>
</ZLineIC>
<ZLineIC >
    <ICName value="1280994101"> </ICName>
    <Parameters value="11,1280991954,1,1280991921,1,1337293"> </Parameters>
</ZLineIC>
<ZLineIC >
    <ICName value="1280994111"> </ICName>
    <Parameters value="9,1280992302,3,1280992195,3,1337293"> </Parameters>
</ZLineIC>
</DRVI4.1>

```

2.4 软件平台菜单

2.4.1 系统菜单

使用系统功能菜单可以完成新建、打开、保存 VI 文件，建立符合 VI 文件，读写服务器资源文件及用户管理等功能



新建 VI 文件

创建一个空的脚本文件，用以容纳插接在软件总线上的软芯片参数。

打开 VI 程序文件

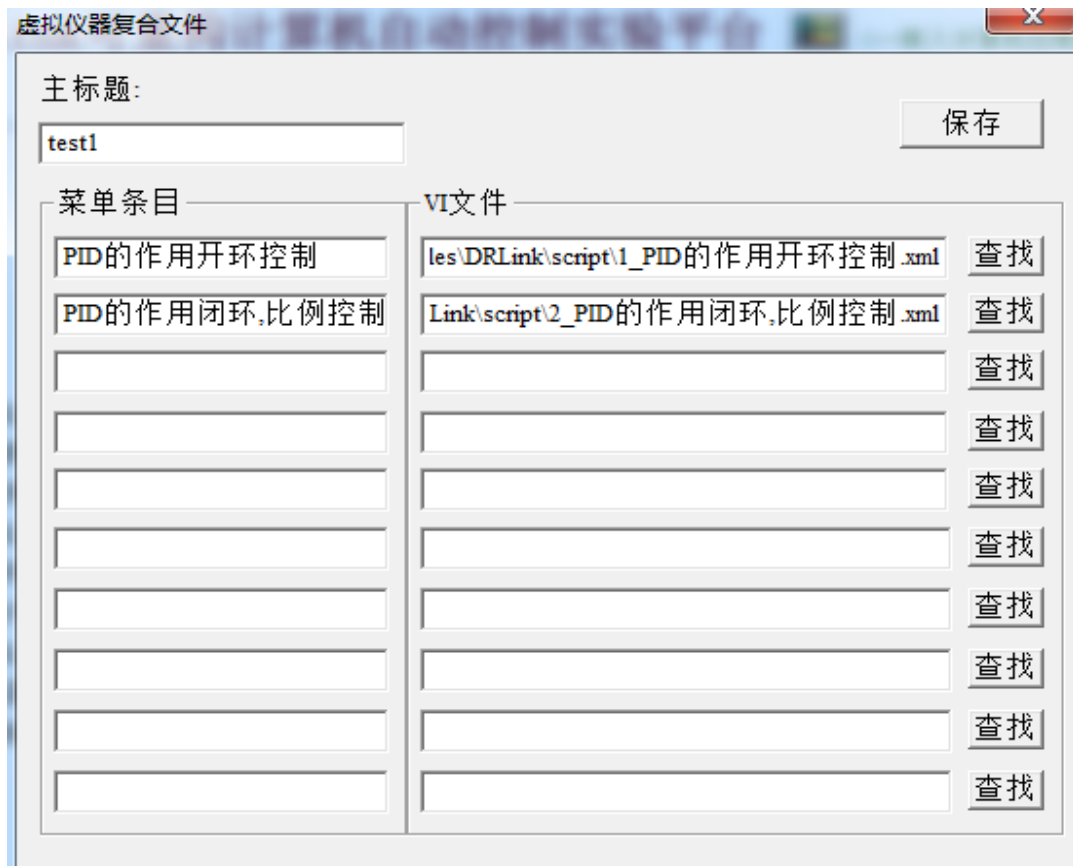
打开已保存子 VI 程序文件以供修改使用。

存子 VI 程序文件

将插接在软件总线上的软芯片参数保存到磁盘上的子 VI 程序文件。将保存的子 VI 程序文件复制到 DRVI/DRLink 程序的相同目录下，DRVI/DRLink 可以将这个子 VI 程序文件自动添加到总线芯片中，以供用户重复调用。

建立虚拟仪器符合 VI 文件

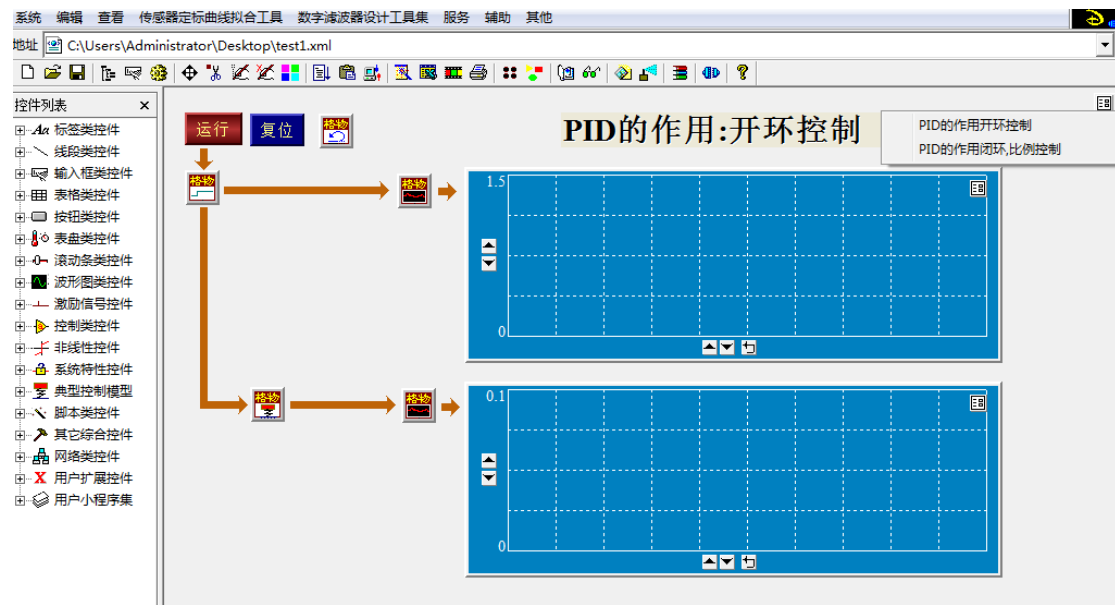
使用建立虚拟仪器复合文件，可以将几个脚本同时保存一个复合虚拟仪器脚本文件中。DRVI/DRLink 再读入虚拟仪器复合脚本文件到虚拟仪器面板中来，可以同时读入几个脚本文件，并且在 DRVI/DRLink 虚拟仪器环境中使用菜单、标题栏、旋钮来进行脚本之间的切换。下面介绍怎么建立虚拟仪器复合文件。点击建立虚拟仪器复合文件菜单，然后弹出如下窗口：



先填入一个主标题，上例中选择的是 test1，然后点击右侧的查找按钮添加相应的脚本。点击保存按钮建立符合脚本文件。

要读入符合脚本文件，使用读入脚本文件的功能就可以正常读入复合脚本文件。读入上例的复

合脚本文件如下图所示：



读服务器资源文件

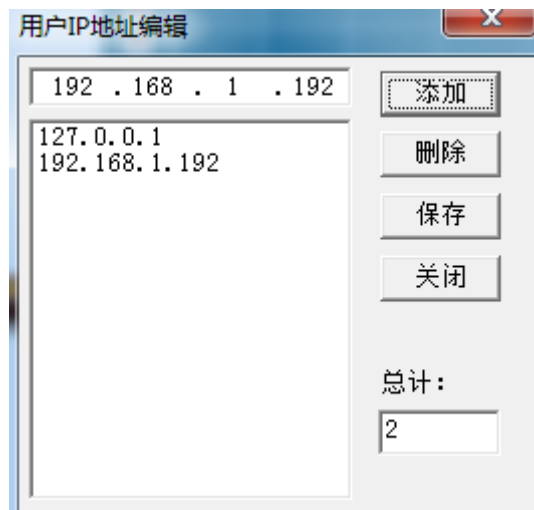
通过网络读取另一台计算机上运行的 DRVI/DRLink 软件总线上插接的 IC 芯片信息，并按读取软件芯片在本地 DRVIDRLink 软件总线上插接和复制。弹出对话框，然后输入对方 IP 地址，最后点击设定。（注意：对方嵌入式 Web 服务器必须在开启状态）

写服务器资源文件

通过网络将本机 DRVI/DRLink 上插接的软件芯片信息发送到另一台计算机上运行的 DRVI/DRLink 中，并在其软件总线上插接和复制。弹出对话框，然后输入对方 IP 地址，最后点击设定。（注：对方的嵌入式 Web 服务器必须在开启状态）

用户管理

对局域网中 DRVI/DRLink 用户进行管理，设置合法用户的 IP 地址。选该条目弹出合法用户地址编辑输入对话框，如下图所示：



将合法用户的 IP 地址添加到用户栏中，点保存，关闭 DRVI/DRLink，重新启动 DRVI/DRLink，开启 DRVI/DRLink 中的 Web 服务器功能，然后合法用户就可以通过网络注册使用了。

2.4.2 编辑菜单

编辑菜单提供编辑状态选择、删除控件、连接控件、删除 连接色彩选择、VI 源文件、粘贴 VI 脚本、图形拷贝、数据拷贝和屏幕打印等功能。

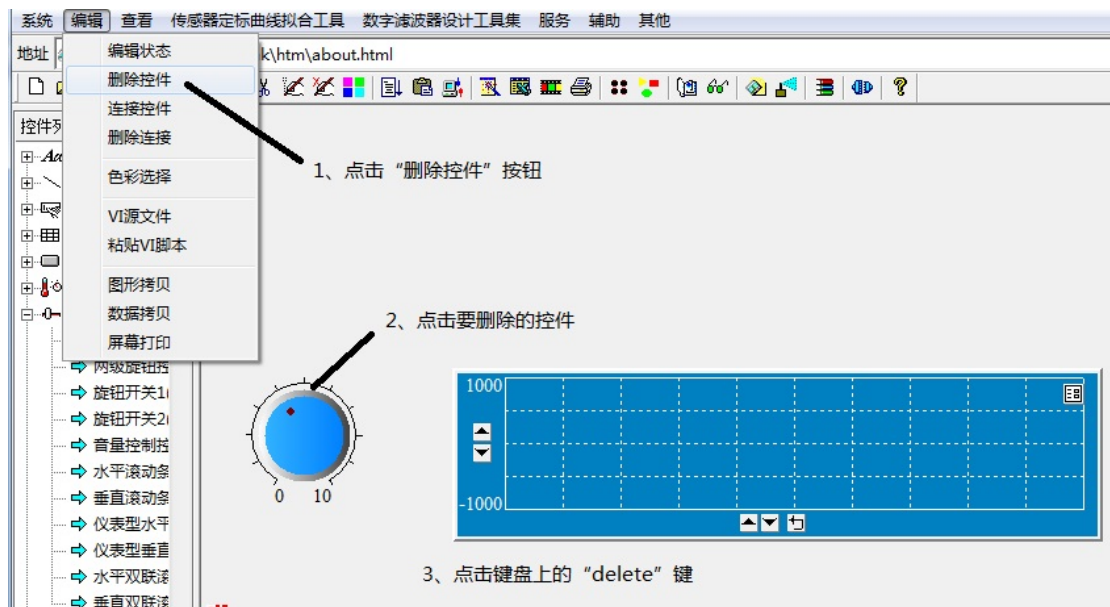


编辑状态

进入编辑状态，对虚拟仪器布局区的控件大小、位置进行 改变。方法是点击编辑中的编辑状态即可。退出编辑状态即在 虚拟仪器布局区的空白处双击。

删除控件

从软件总线上删除一片芯片。方法是选择该条目，然后用鼠标点击要删除的芯片，最后按下键盘上的“Delete”键。



连接控件

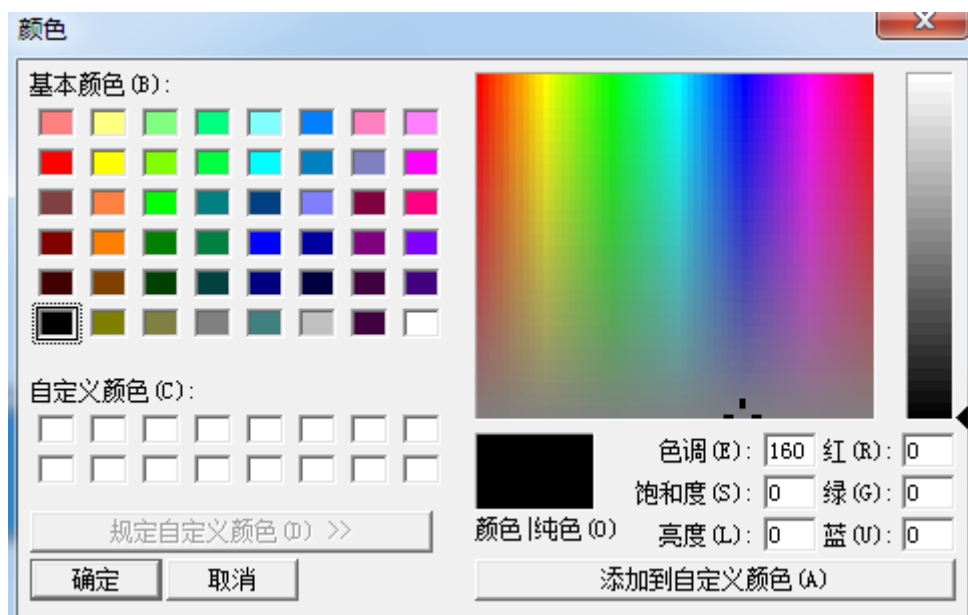
将控件连接到软件总线上面。方法是选择该条目，然后左键点击要连接的芯片，然后选择要连接芯片的端口，然后选择与该芯片连接的另外芯片的端口即可完成连接。DRVI/DRLink 系统会自动创建一条软件总线去完成 2 个芯片的数据交换工作。

删除连接

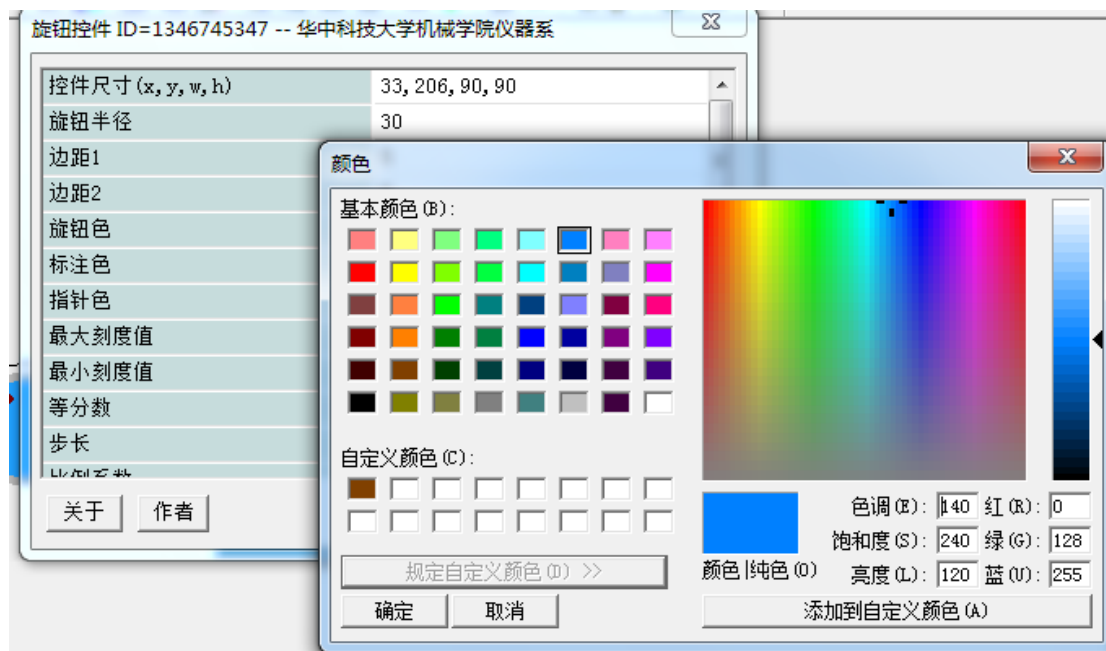
删除一条数据总线。方法是选择该条目，然后

色彩选择

选择软件芯片中使用的 RGB 色彩码。方法是选择该条目，弹出色彩选择对话框如下图所示：



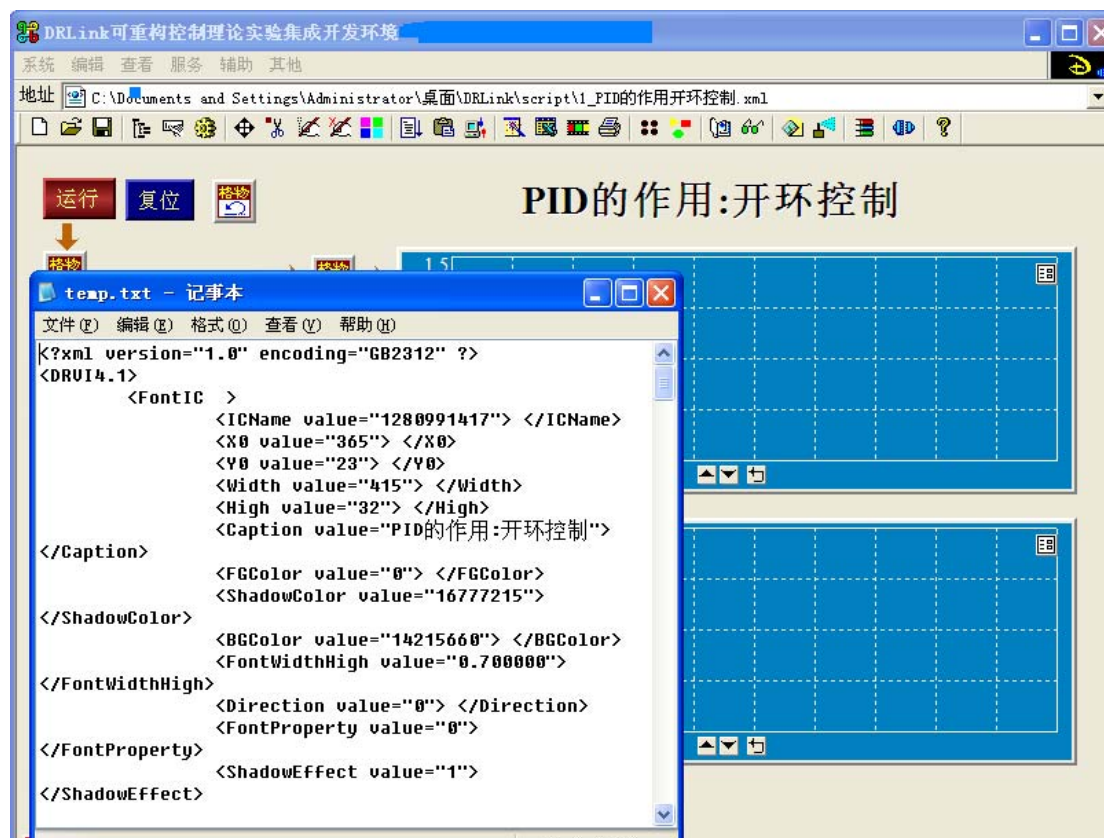
点击设定将选择的画笔色彩码放在剪切板上，然后在软件芯片上点击右键弹出属性窗，最后进入色彩参数编辑框，点击右键弹出编辑窗，将色彩 RGB 代码添入色彩编辑栏上去就可以了。



VI 源文件

显示当前插接在软件总线上的软件芯片布局及设置的脚本文件。

DRVI/DRLink 采用 VI 脚本文件来对构造的虚拟仪器惊醒描述，脚本文件的作用类似于浏览器中使用的 HTML 网页。用户可用脚本文件来交换虚拟仪器设计情况。



粘贴 VI 脚本

将存放在剪贴板上的 VI 脚本文件插入到软件总线上。

图形拷贝

将当前 DRVI/DRLink 显示窗口中的内容以位图形式拷贝到剪贴板上，然后用户可以用“Word”，“画笔”等软件中的粘贴功能将图片插入到这些软件的编辑区内。

数据拷贝

将存储在软件内存芯片中的波形、频谱等分析数据以 EXCEL 电子表格数据格式拷贝到剪贴板上，然后用户可以用“EXCEL”中的粘贴功能将数据插入“EXCEL”的编辑区内。

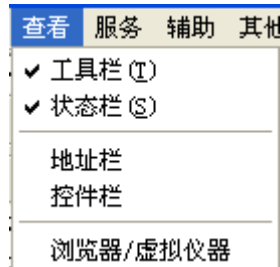
选择该功能后弹出对话框，输入待拷贝的波形、频谱数据的软件内存条芯片编号和拷贝的数据长度（点数），然后点击拷贝即可。



屏幕打印

将当前 DRVI/DRLink 显示窗口中的内容以位图形式拷贝到剪贴板上，然后用户可以用“Word”，“画笔”等软件中的粘贴功能将图片插入到这些软件的编辑区内，进行打印等操作。

2.4.3 查看菜单



查看菜单提供显示/隐藏工具栏、状态栏、地址栏、控件栏和 切换浏览器/虚拟仪器界面的功能。

工具栏 状态栏 地址栏 控件栏

在用户选择点击工具栏、状态栏、地址栏、控件栏激活前面的小勾时，DRVI/DRLink 系统会将工具栏、状态栏、地址栏、控件栏显示出来而当用户再次点击取消工具栏、状态栏、地址栏、控件栏的小勾时，工具栏、状态栏、地址栏、控件栏则会隐藏。

浏览器/虚拟仪器

点击浏览器/虚拟仪器按钮会将 DRVI/DRLink 主界面由浏览器与虚拟仪器界面之间相互切换。

2.4.4 服务菜单

服务菜单提供将本地计算机打开 DRVI/DRLink 服务器功能的方法，在线局域网中的计算机扫描的功能，本机网络参数获取的功能。

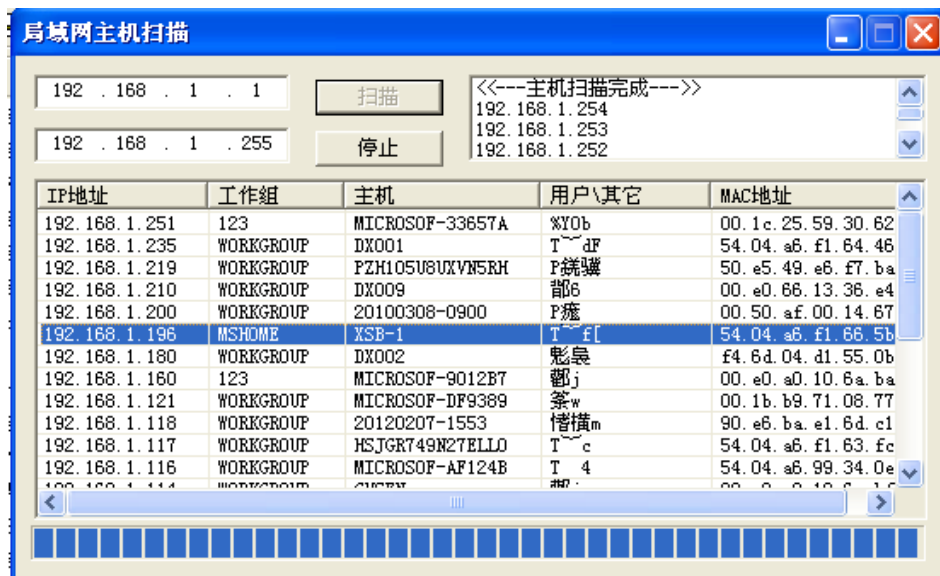


DRVI/DRLink 服务器

启动 DRVI/DRLink 内置的 Web 服务器和同端口的数据服务器，支持浏览器访问、控制和网络数据交换，服务器端口为 8500。服务器启动后 DRVI/DRLink 标题栏中将出现“服务器 开”的字样。DRVI/DRLink 内置 Web 服务器开启后就可以用浏览器进行访问。

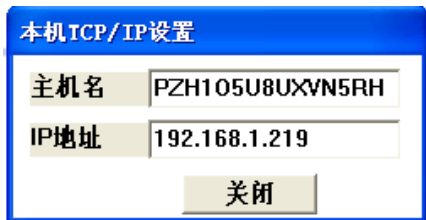
在线计算机扫描

使用鼠标左键点击“在线计算机扫描”后，弹出对话框。可以扫描出指定网段中的计算机数量及 IP 等网络参数。



本机网络参数

查看本机网络参数。



2.5 快捷工具条

DRVI/DRLink 工具条提供了常用的资源文件读/写、软件布局、控件连接、屏幕拷贝、数据拷贝等功能。



新建

新建一个 IC 资源文件。

打开

打开一个 IC 资源文件。

保存

保存一个 IC 资源文件。

控件表

单击控件表，显示或者隐藏控件树。

地址栏

单击地址栏，显示或者隐藏地址栏目。

编辑状态

单击编辑状态，进入编辑状态可以对控件尺寸、大小、布局进行编辑。

删除控件

单击删除控件选择删除所有控件会将布局到前面板上的所有控件进行删除。选择删除当前选中控件会进入编辑状态，选中要删除的控件按下“Delete”按键完成对控件的删除。

连接控件

点击连接控件进入连接控件的编辑状态，点击选择要连接的控件，点击左键选择要连接的端口，点击选择另外与之相连的控件选择连接的端口，即可完成连接。

删除连接

点击删除连接，进入删除连接的编辑状态。选择要删除的线，左键点击控件选择要删除的线即可完成连线的删除。

色彩选择

点击色彩选择，进入系统调色板。



源文件

点击源文件按钮，获取 VI 源文件代码。

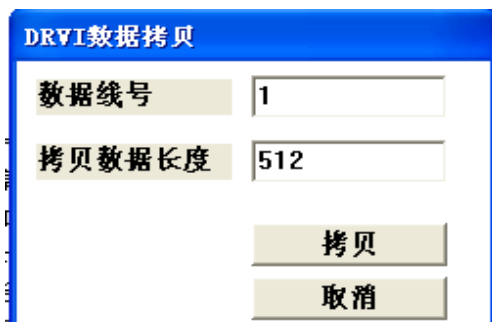


图形拷贝

点击图形拷贝按钮，会将前面板进行截屏。

数据拷贝

点击数据拷贝按钮，弹出如下对话框。点击拷贝将拷贝指定线号中的数据到剪切板中。



GIF 动画

点击 GIF 动画，将在前面板中添加一个 GIF 动画控件。将控件连接到数据总线中可以将前面版中的动作制作成为 GIF 动画插入到多媒体教学中。

虚拟仪器样例

点击虚拟仪器样例，直接在下拉列表中选择相应制作好的虚拟仪器进行工作。

DRVI/DRLink 使用 说明

点击 DRVI/DRLink 使用说明将弹出 DRVI/DRLink CHM 使用说明。

关于

点击关于，将弹出 DRVI/DRLink 版本情况。

第三章、DRVI/DRLink 虚拟仪器设计

3.1 软件总线的概念

计算机硬件总线技术的发展和插件式硬件模块结构，使计算机的生产、组装和升级变得十分容易，普通用户也能选用和安装新的硬件老重构个人计算机的功能。计算机硬件总线技术这种积木化结构设计成功，促使了计算机软件总线概念的诞生和应用。

与传统的软件开发系统相比，基于软件总线的开发系统利用 COM/DCOM 组件支持二进制代码模块直接引用的特点，可取消传统软件开发中的编译、链接环节，实现系统开发环境和应用环境的统一，实现软件模块的即插即用，实现系统功能的在线调整。用户不需要了解复杂的函数调用，因为虚拟仪器软件总线和软件芯片的所有设置都可以通过 COM 控件的属性页来设置，这样可以大大减少用户软件开发时间。

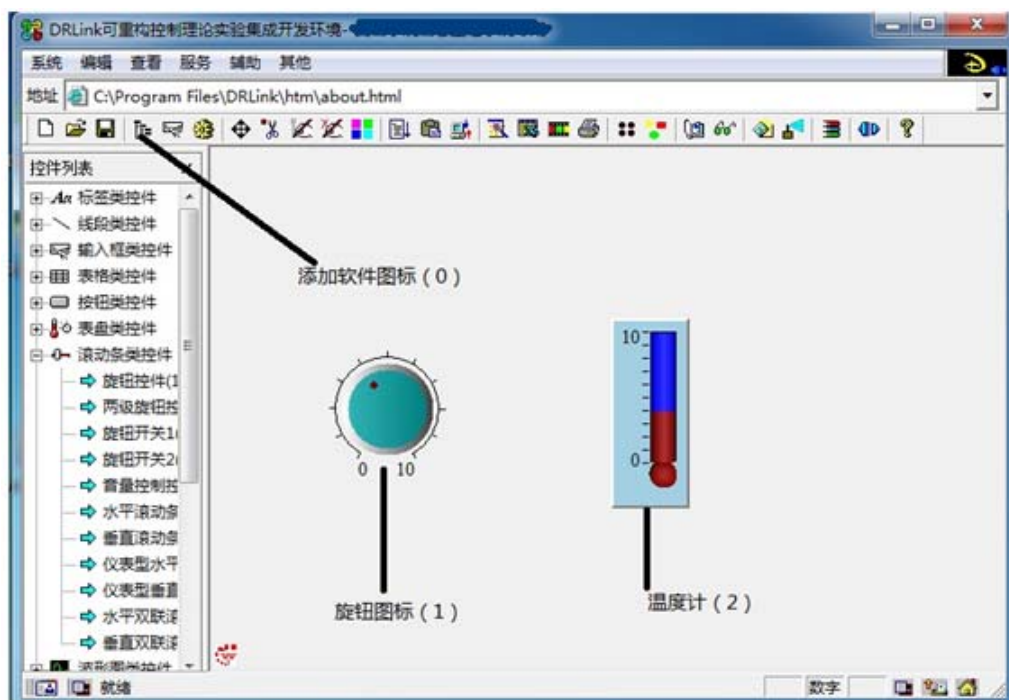
深圳德普施科技有限公司采用软件总线结构和思想，成功的开发了基于软件总线和软件芯片的快速可重构虚拟仪器开发平台，起最大的优点是支持虚拟仪器软件芯片的热插拔和即插即用，支持在运行阶段虚拟仪器功能的调整，能适应测试任务变更和调整的需要。

DRVI/DRLink 的核心是一个 COM/DCOM 容器程序，表头、按钮、信号处理模块、硬件驱动模块等虚拟仪器软件模块则设计为 COM/DCOM 组件。利用 COM/DCOM 容器程序对 COM/DCOM 组件的热插拔支持，可以实现对虚拟仪器软件芯片的热插拔。

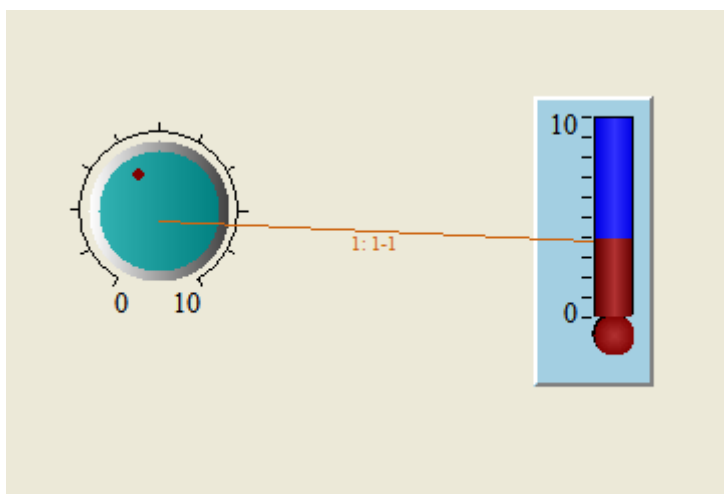
为实现虚拟仪器软件芯片之间的数据交换，DRVI/DRLink 中设置了 256 条数据线组成的软件总线，软件总线可传输有效值等单变量数据，也可传输波形、频谱等数据。虚拟仪器软件芯片可以通过这组透明的数据总线进行数据传输和命令数据交换。任何两个虚拟通讯线路连接在一起的节点间可以彼此交换数据一样。

下面通过例子来说明 DRVI/DRLink 中软件总线的概念和虚拟仪器软件芯片与软件总线的连接，以及系统的工作原理。

先启动 DRVI/DRLink，显示虚拟仪器/运行面板，点击添加软件图标 (0)，弹出软件芯片选择窗，点击旋钮图标 (1) 和温度计图标 (2)，将这两个软件芯片添加到布局区内。

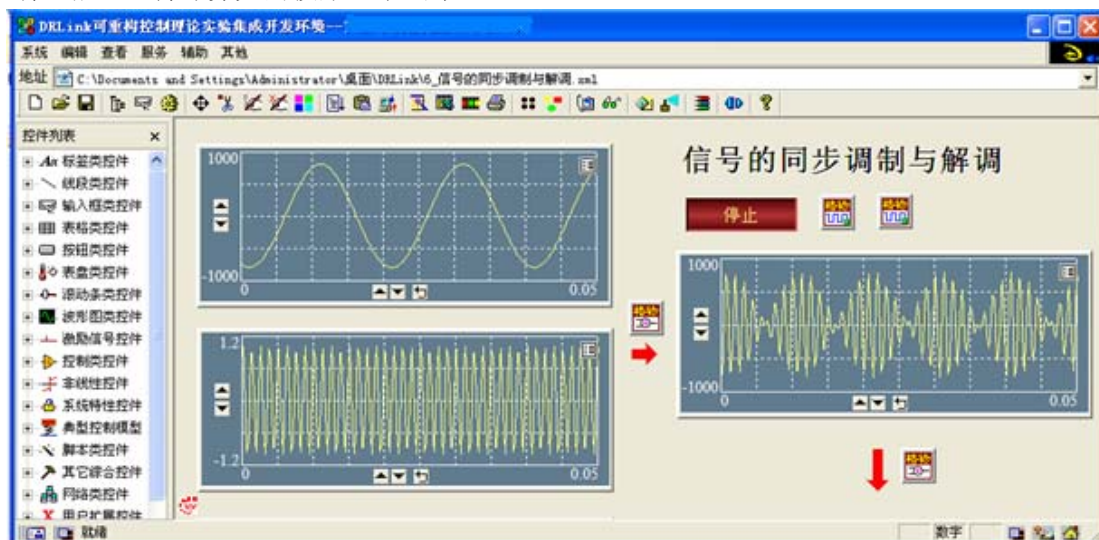


在两个芯片上分别点击右键鼠标键可以弹出属性参数设置对话框，修改其中的属性值可以对控件的样式显示文字等条目进行相应的改变。然后使用连接控件的工具将两个控件连接起来。这时两个虚拟仪器软件芯片就通过软件总线连接在一起了，用鼠标旋动按钮，则温度计的指示值也在同步变动。



DRVI/DRLink 软件总线带数据驱动功能，任何一个虚拟仪器软件芯片改动数据线上的值后就会在总线上形成一个数据流，驱动连接在该数据线上的其它虚拟一起软件芯片。

为进一步说明问题，下面给出通过一个“信号的同步调整与解调”实验的例子。先插入一个开关芯片，然后插入 2 个信号发生器芯片，将开关芯片的开关数据线与两个信号发生器端口连接起来，实现开关按钮对于信号发生器的启停的控制。接着插入两个波形图显示控件，将两个信号发生器的信号输出端口分别于两个波形图显示控件的幅值输入端口相连，将信号发生器产生的信号显示在波形图上面。接着插入一个波形合成控件、一个波形图显示控件，分别将两个信号发生器输出端口与波形合成控件的通道 1 信号与通道 2 信号相连，然后将波形合成芯片的合成信号输出端口与波形图显示端口相连，将计算完的波形显示出来。



虚拟仪器软件芯片与软件总线连线过程很简单，首先在快捷工具栏中选择连线工具，然后在软件芯片上点击左键选择要连接的端口，然后到目标控件上点击左键选择与之连接的端口便完成与软件总线的连接。连好线后，各软件芯片就在数据流的驱动下联合工作，形成一个特定功能的虚拟仪器。软件总线是一个并行的结构，所实现系统功能与软件芯片的插接次序无关，同时各软件芯片间也没有直接联系。程序工作的任何时段都可以通过热插/拔软件芯片来调整系统功能，这一点就像计算机硬件中的 USB 设备一样。

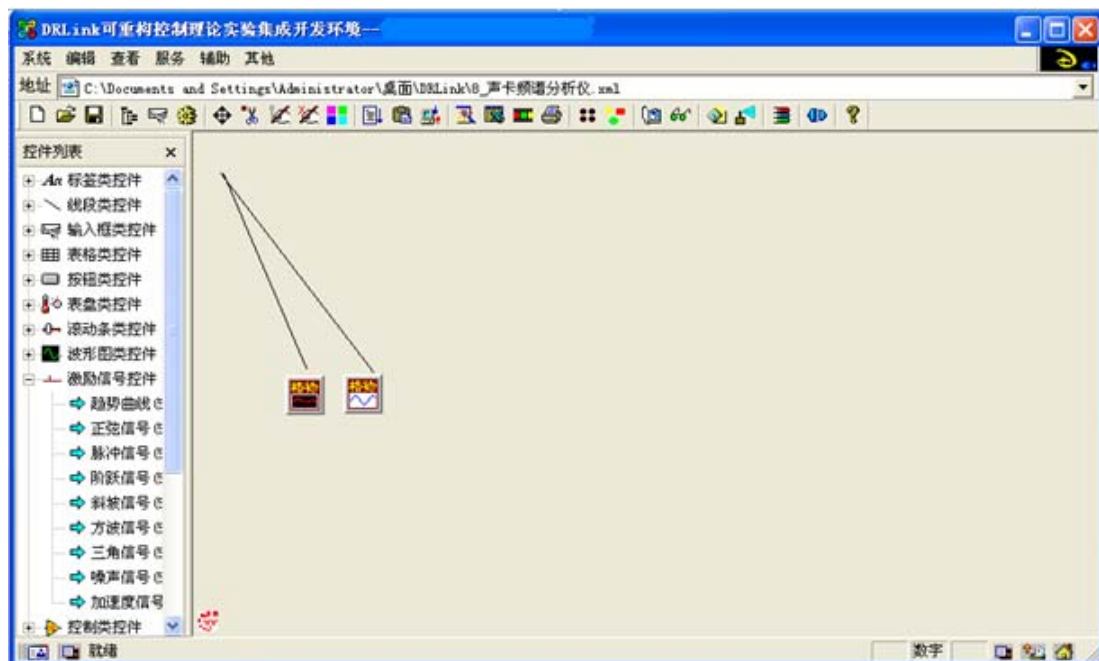
3.2 插接软件芯片

DRVI/DRLink 通过在前面板上可视化插接虚拟仪器软件芯片来构建虚拟仪器或测量实验。插接软件芯片的过程很简单。

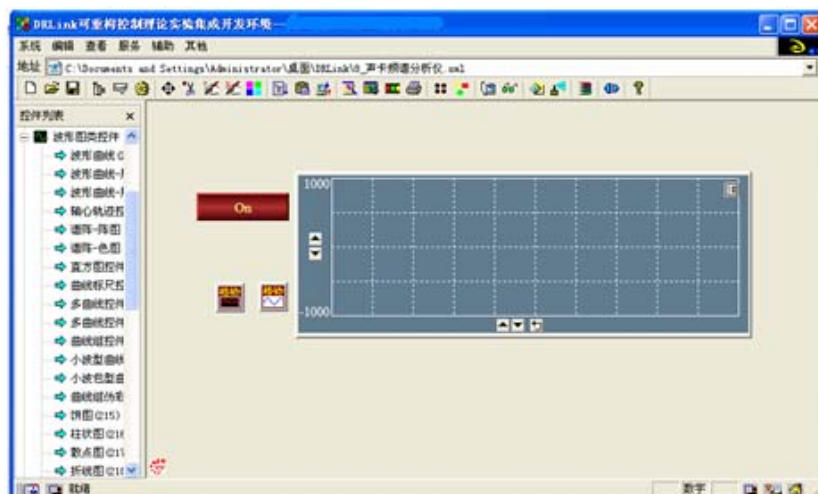
- 1、**点击软件芯片表中的图标选择要插接的控件。**
- 2、将鼠标移动到前面板在需要插接的地方点击鼠标左键完成插接。
- 3、在 DRVI/DRLink 前面板上新插接的软件芯片上点下鼠标不放，将其拖动到合适的位置。
- 4、重复 1-3 步骤，插入其他软件芯片。

DRVI/DRLink 采用软件总线的并行结构，所设计的虚拟仪器功能与芯片的插接顺序无关。先插接信号采集，然后是分析、计算芯片，最后插接按钮类、显示类芯片。

例如，设计一个简单的通过产生正弦波信号并显示的虚拟仪器过程如下。首先在控件芯片表中点击选择激励信号控件中的正弦信号，插入一片正弦信号发生控件，用于产生正弦信号芯片插接后用鼠标将其拖动到合适的位置。再从激励信号控件中找到趋势曲线控件将其插入到前面板中并拖动到合适的位置。



在控件列表中选择**波形图类控件**，点击波形图-展缩控件，用于显示采样信号波形，芯片插接后使用鼠标将其拖动到合适位置；最后从控件列表中选择按钮类控件点击“启停按钮”控件使用以上方法插接到前面板中并放到合适的位置。



3.3 软件芯片连线

DRVI/DRLink 通过采用数据进行软件芯片与软件总线的连线，修改软件芯片属性表中的数据就可以更改软件芯片与软件总线的连接关系，实现基于数据的焊接。软件芯片属性如下有三种类型。

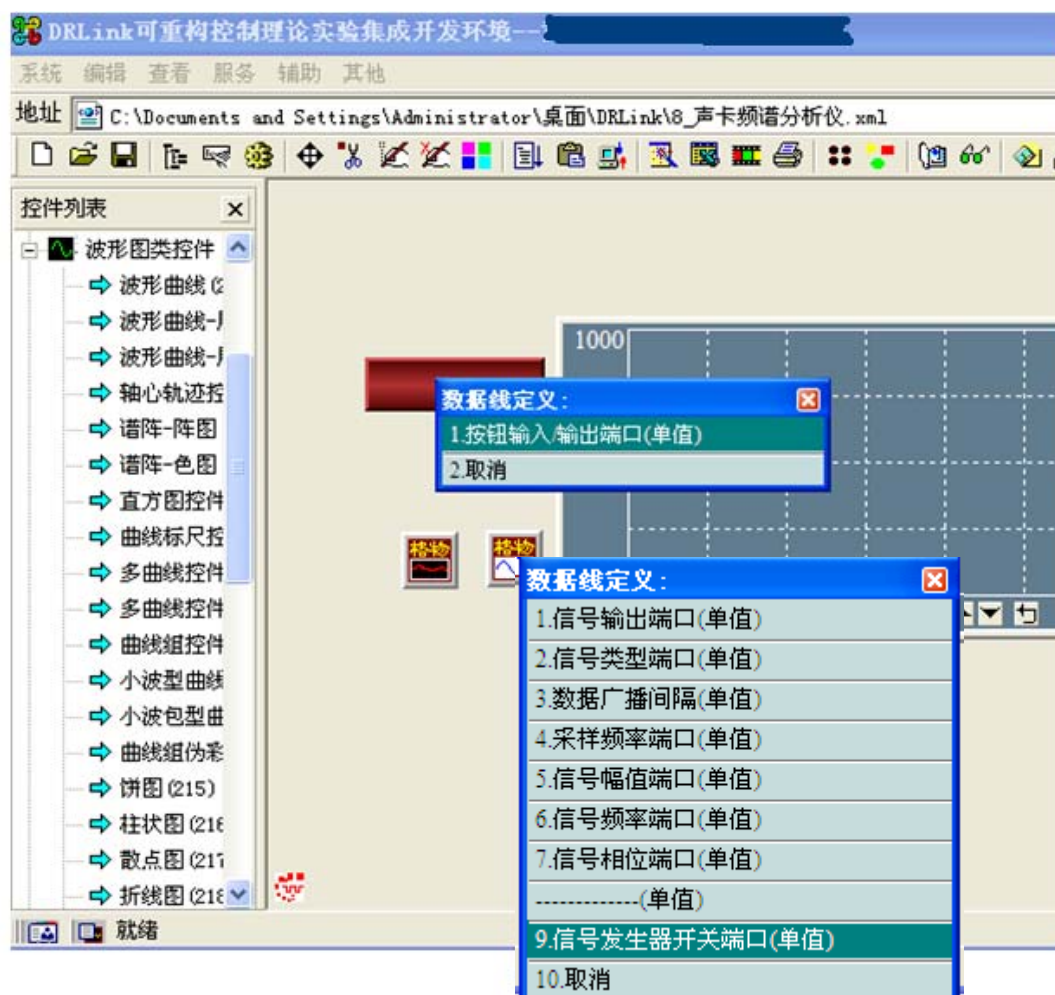
I 类参数：软件芯片内部属性参数，只影响软件芯片自身的显示方式，计算方法等。该参数与外界无关，设定后其值不会变化。

II 类参数：软件芯片的外部属性参数，通常是软件总线上某条数据线上的值，该类参数与外界相关，其值能够被其他软件芯片通过数据线从外部修改。

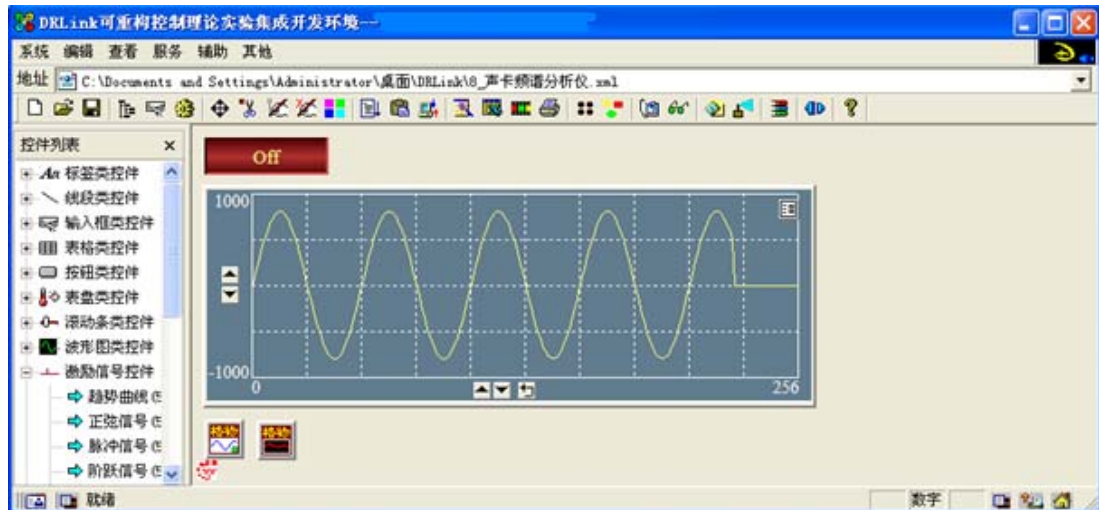
III 类参数：软件芯片与软件总线的接口参数，定义了软件芯片与软件总线数据线和数组型数据线的连接关系。该类参数与外界孤寡设定后其值不会变化。

DRVI/DRLink 中内部设置了 256 条数据线组成的软件总线，任何两个虚拟仪器软件芯片只要连接在一条数据线上就可以在彼此之间交换数据和进行控制。

例如，上节介绍通过正弦信号发生芯片产生正弦波的例子。插入完了所有芯片后就可以对芯片进行连线。首先通过连接按钮输入输出端口和信号发生器开关端口。在控件上点击左键选择相应的端口，软件自动完成连接。然后在使用相同的方法连接趋势曲线与现实波形图的控件。



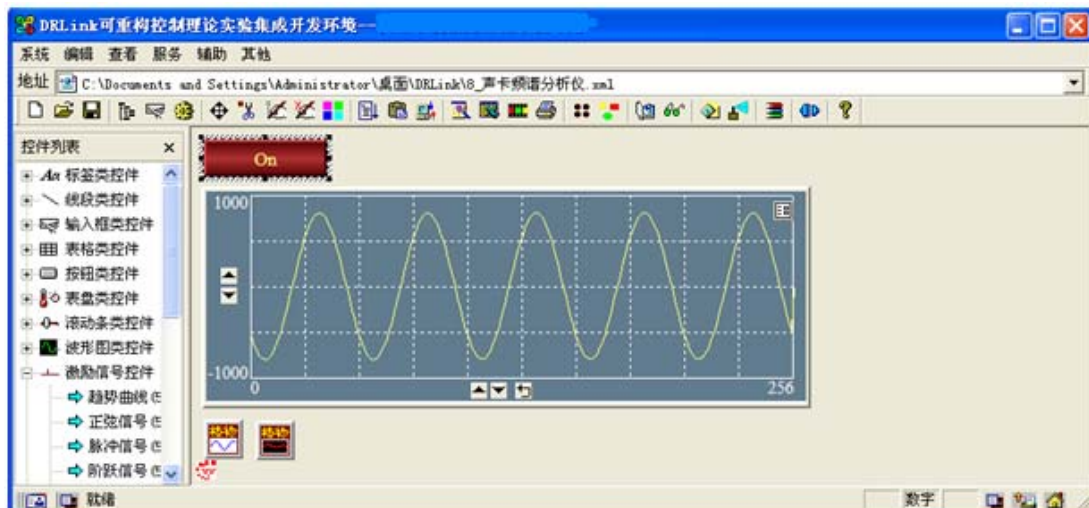
这样 4 个虚拟仪器软件模块就逻辑上连接到了仪器，用鼠标按下开关就会在波形显示芯片上产生正弦波形了，再次点击“开关”就停止了产生过程，波形就停止了。



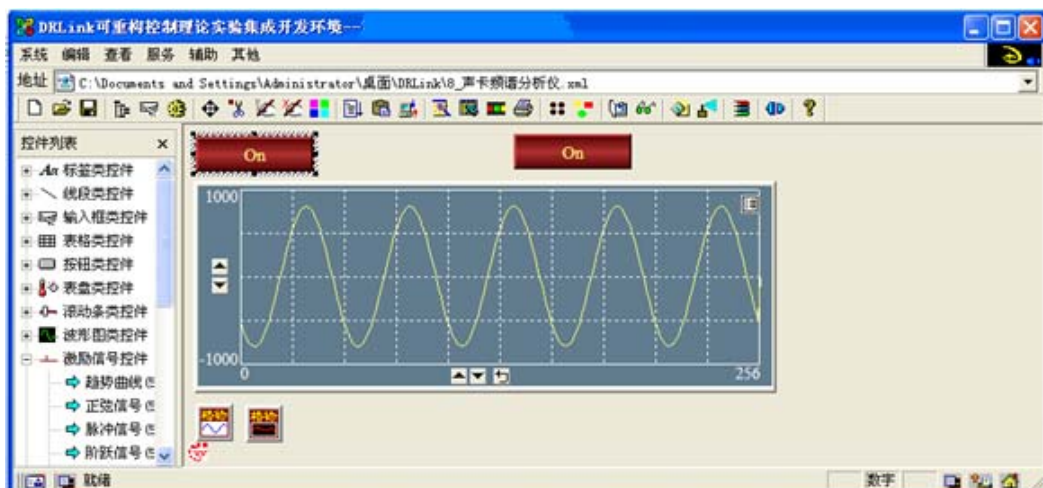
整个连线过程就这样简单，只要按虚拟仪器和测试实验任务要求选择好虚拟仪器软件和设计出它们的连接关系，就可以快速设计出一个符合测试任务要求的虚拟仪器。

3.4 软件芯片屏幕布局

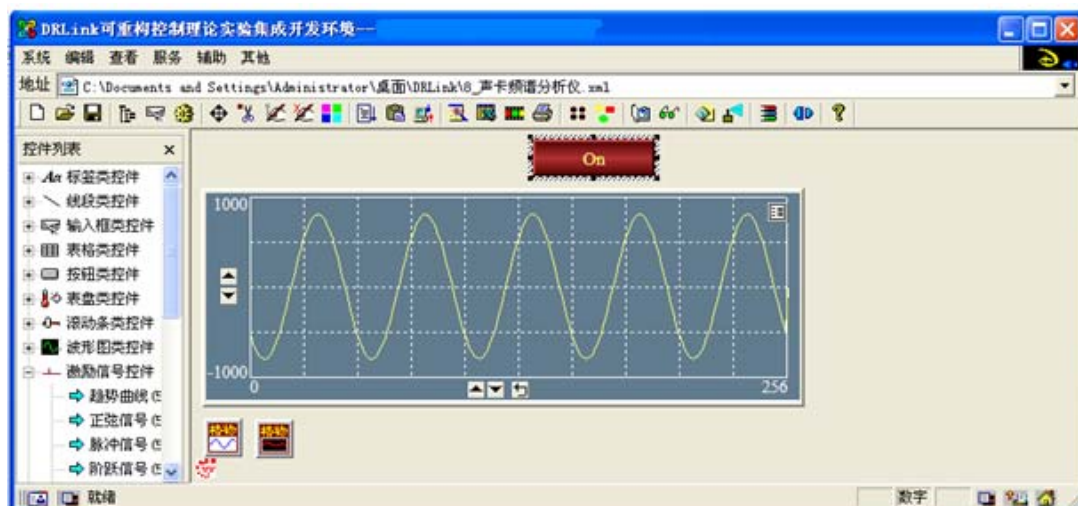
插接在 DRVI/DRlink 前面板上的虚拟仪器软件芯片的屏幕位置是可以移动和调整的，点击快捷工具条中的“编辑状态”。进入编辑状态后，点击要移动的芯片。



然后在待移动的软件芯片上按下鼠标不放就可以将其拖动到新位置。



松开鼠标左键，芯片移动到指定位置，从而实现屏幕的调整。



3.5 软件芯片一览表

DRVI/DRLink 虚拟仪器实验平台提供了搭建各种虚拟仪器实验所需要的各种芯片，其中包括显示类控件，输入类控件，运算类控件和用户自定义控件。

标签芯片

标签芯片（文字标签、水平文字、垂直文字、艺术字）的作用是：在 DRVI 前面板中插入一条文字信息显示标签，以显示一些说明性文字。另外它也可以与软件总线上的一条数据线相连，动态显示数据线上的数据值。

图标芯片

图标芯片的作用是：插入一个图形显示控件，用于显示 JPG 或 GIF 格式的图形文件，或动态显示内存数组中的图形内容。

GIF 动画芯片

GIF 动画芯片的作用是：插入一个 GIF 动画图标，以显示一些需要动画的装饰。另外它也可以与软件面板板上的数据线相连，控制动画图片的运行。

边框线芯片

边框线芯片的作用是：在屏幕上绘制一条矩形边框线，用来对版面进行装饰。

流程线芯片

流程线芯片的作用是：在屏幕上绘制一条流程线，用来描述程序数据流向。

箭头线芯片

箭头线芯片的作用是：在屏幕上绘制一条箭头线，用来指示芯片的工作顺序，或信号的流向。

3D 效果装饰线芯片

3D 效果装饰线芯片的作用是：在屏幕上绘制一条 3D 效果的装饰线。

数字调节按钮芯片

数字调节按钮芯片的作用生成一个可调节的数字输入控件。数字调节按钮芯片与软件总线中的一条数据线相连，用户可以通过鼠标点击来调节和改变该数据线上的值，从而达到改变连接在该数据线上其他软件芯片工作参数的目的。连接在该数据线上其他软件芯片对数据的修改也会改变数字调节按钮芯片的显示值。

输入框芯片

输入框芯片的作用是：为系统提供一个数值输入文本框，用户可以在其中直接输入数值。输入框芯片与软件总线中的一条数据线相连，修改输入框中的值就可以改变该数据线上的值，从而达到

改变连接在该数据线上其他软件芯片工作参数的目的。连接在该数据线上其他软件芯片对数据的修改也会改变输入框芯片的显示值。

多项选择芯片

多项选择芯片的作用是：提供多选一功能。多项选择芯片与软件总线中的一条数据线相连，用户可以通过鼠标点击多项选择芯片来改变所连接的输出数据线的状态值（0, 1, 2, ...），从而达到控制连接在该数据线上其他软件芯片工作状态的目的。

下拉选择芯片

下拉选择芯片的作用是：提供多选一功能。下拉选择芯片与软件总线中的一条数据线相连，用户可以通过鼠标点击下拉选择芯片来改变所连接的输出数据线的状态值（0, 1, 2, ...），从而达到控制连接在该数据线上其他软件芯片工作状态的目的。

HTML 报表芯片

HTML 报表芯片的作用是：插入一个 HTML 文档显示框，以态显示一些 HTML 格式的报表和文字。它也可以与软件面包板上的一条数据线相连，动态显示数据线上的 HTML 内容。

GIRD 数字/文字报表芯片

GIRD 数字/文字报表芯片的作用是：插入一个 GIRD 报表显示框，以态显示一些数字和文字。它也可以与软件面包板上的一条数据线相连，动态显示数据线上的内容。

数据库表格 (Browse) 芯片

数据库表格 (Browse) 芯片的作用是：插入一个 GIRD 报表显示框，以 Browse 方式显示数据库中的内容。它也可以与软件面包板上的一条数据线相连，将表格中的数据输出。这是一个扩展插件，需要 DBFArray.dll 支持。

数据库表格 (Edit) 芯片

数据库表格 (Edit) 芯片的作用是：插入一个 GIRD 报表显示框，以 Edit 方式显示数据库中的内容。它也可以与软件面包板上的一条数据线相连，将表格中的数据输出。这是一个扩展插件，需要 DBFArray.dll 支持。

开关芯片

开关芯片的作用是生成一个可控制状态的开关量控件：通过鼠标对此芯片的点击来向其他芯片发出一个单次运行/停止控制命令。

几何图形开关芯片

几何图形开关芯片的作用是生成一个几何形状的可控制状态的开关量控件：通过鼠标对此芯片的点击来向其他芯片发出一个单次运行/停止控制命令。

操作图标按钮

操作图标按钮的作用是生成一个带图标的可控制状态的开关量控件：通过鼠标对此芯片的点击来向其他芯片发出一个单次运行/停止控制命令。

多联开关芯片

多联开关芯片的作用是：提供一个多按键组合开关。多联开关芯片与软件总线中的一条数据线相连，用户可以通过鼠标点击多多联开关芯片来改变所连接的输出数据线的状态值（0, 1, 2, ...），从而达到控制连接在该数据线上其他软件芯片工作状态的目的。

图标芯片

图标芯片的作用是生成一个带图标的可控制状态的开关量控件：通过鼠标对此芯片的点击来向其他芯片发出一个单次运行/停止控制命令。这是一个扩展插件，需要 ICONLib.dll 支持。

信号操作工具条

信号操作工具条的作用是：提供一个曲线操作按键组，可用鼠标对所连接的软件总线数组型数据线上的波形或频谱数据进行移动、扩展、幅值放大等操作，处理后的数据从另一条数组型数据线输出。

温度计型仪表芯片

温度计型仪表芯片的作用是：提供类似物理设备温度计、容器液位计的显示功能。温度计型仪表芯片与软件总线中的一条数据线相连，数据线上数据的变化会使温度计型仪表芯片的液位示值也同步变化。

数码表头芯片

数码表头芯片的作用是：提供类似物理设备数码表头的显示功能。数码表头芯片与软件总线中的一条数据线相连，数据线上数据的变化会使数码表头芯片的显示值也同步变化。

圆型仪表芯片

圆型仪表芯片的作用是：提供一个和物理指针式圆型仪表盘功能相似的芯片。圆型仪表芯片与软件总线中的一条数据线相连，数据线上数据的变化会使圆型仪表芯片的指针示值也同步变化。

方型仪表芯片

方型仪表芯片的作用是：提供一个和物理指针式圆型仪表盘功能相似的芯片。方型仪表芯片与软件总线中的一条数据线相连，数据线上数据的变化会使圆型仪表芯片的指针示值也同步变化。

数码管芯片

数码管芯片的作用是：提供类似于物理设备 LED 数码显示屏的功能，用 LED 方式显示数字。LED 显示芯片与软件总线中的一条数据线相连，可以动态显示数据线上数据的变化。

进程条芯片

进程条芯片的作用是：用于动态显示系统中一个任务的执行进度或完成的比例。进程条芯片与软件总线中的一条数据线相连，根据数据线上的值显示任务执行程度。

警示灯芯片

警示灯芯片的作用是：模拟物理设备信号灯的功能。警示灯芯片与软件总线中的一条数据线相连，可以根据数据线上不同的“灯状态”值显示不同的颜色。

旋钮芯片

旋钮芯片的作用是：为用户提供一个和物理旋钮功能类似的芯片。旋钮芯片与软件总线中的一条数据线相连，用户可以通过鼠标的点击和拖动来旋转芯片图标上的指针，调整数据线上的值，从而达到改变连接在该数据线上其他软件芯片工作参数的目的。

旋钮开关芯片

旋钮开关芯片的作用是：为用户提供一个和物理旋钮开关功能类似的芯片。旋钮开关芯片与软件总线中的一条数据线相连，用户可以通过鼠标的点击和拖动来旋转开关芯片图标上的指针，调整数据线上的值，从而达到改变连接在该数据线上其他软件芯片工作参数的目的。

推杆芯片

推杆芯片的作用是：模拟物理设备推杆的功能。推杆芯片与软件总线中的一条数据线相连，用户可以通过鼠标拖动推杆上的滑块来改变数据线上的值，从而达到改变连接在该数据线上其他软件芯片工作参数的目的。推杆芯片也可以用作示值器，连接在该数据线上其他软件芯片对数据的修改也会改变推杆上滑块位置。

波形/频谱显示芯片

波形/频谱显示芯片的作用是：在屏幕上用二维曲线方式显示所连接的软件总线数组型数据线上的波形或频谱数据。曲线图分展缩和非展缩两种，非展缩曲线图显示的数据长度由图形宽度决定，也就是一个屏幕像素对应曲线上一个数据；展缩曲线图显示的数据长度则由设定的显示数据量确定，数据量大时可能几个数据压缩在一个像素点上显示。

标尺曲线图显示芯片

标尺曲线图显示芯片的作用是：在屏幕上用二维曲线方式显示所连接的软件总线数组型数据线上的波形或频谱数据，同时在 X、Y 方向加标尺。曲线采用展缩方式显示，数据量大时可能几个数据压缩在一个像素点上显示。

条形图显示芯片

条形图显示芯片的作用是：在屏幕上用直方图方式显示所连接的软件总线数组型数据线上的数据，如 1/1 和 1/3 倍频程谱。

X-Y 曲线显示芯片

X-Y 曲线显示芯片的作用是：显示以 X、Y 方式同步输入的两通道信号组成的信号波形，如轴心轨迹、李沙育图形等。X-Y 曲线显示芯片与软件总线中的两条数组型数据线相连，一条输入 X 方向数据，另一条输入 Y 方向数据，并动态显示由这两条数据线上数据组成的曲线。

多通道波形/频谱显示芯片

多通道波形/频谱显示芯片的作用是：在屏幕上用二维曲线方式显示所连接的多条软件总线数组型数据线上的波形或频谱数据。曲线图分展缩和非展缩两种，非展缩曲线图显示的数据长度由图形宽度决定，也就是一个屏幕像素对应曲线上一个数据；展缩曲线图显示的数据长度则由设定的显示数据量确定，数据量大时可能几个数据压缩在一个像素点上显示。

谱阵芯片

谱阵芯片的作用是：对一个长时间段的信号进行连续观测，分段显示信号的波形/频谱，并以三维谱阵或色图的方式显示，从而在一个长的观测时间段内了解信号波形/频率成分随时间的变化情况。

曲线组显示芯片

曲线组显示芯片的作用是：在屏幕上显示一组二维曲线。曲线组分普通曲线组、小波系数图、小波包系数图 3 种。

曲线组伪彩色图显示芯片

曲线组伪彩色图显示芯片的作用是：在屏幕上用伪彩色图的方式显示一组二维曲线。

VBScript 脚本芯片

VBScript 脚本芯片的作用是：在 GWVI 前面板上用 Signal VBScript 中的函数进行编程，设计用户自定义的 VBScript 小程序。

VB DLL 芯片

VB DLL 芯片的作用是：在 GWVI 前面板上用 Signal VBScript 中的 DLL 函数加载 DLL，调用其函数进行编程。

VB COM 芯片

VB COM 芯片的作用是：在 GWVI 前面板上用 Signal VBScript 中的 COM 函数加载 COM 组件，调用其函数进行编程。

VB 绘图芯片

VB 绘图芯片的作用是：在 GWVI 前面板上用 Signal VBScript 中的图形函数进行编程，在整个程序窗口范围内对界面进行绘制。

VB 虚拟仪器芯片

VB 虚拟仪器芯片的作用是：在 GWVI 前面板上用 Signal VBScript 中的图形函数进行编程，在控件窗口范围内对界面进行绘制。

公式控件

公式控件的作用是提供一个小的可编程计算器，对数据线变量进行运算。

条件执行控件

条件执行控件的作用是提供一个可编程条件语句，对数据线输入变量状态进行判断，按状态进行输出。

正弦波信号发生器芯片

正弦波信号发生器芯片的作用是按照正弦函数的定义产生一个典型信号，其数学公式如下：

$$x(t) = A \cdot \sin(2\pi \cdot f \cdot t + p)$$

脉冲信号发生器芯片

脉冲信号发生器芯片的作用是按照脉冲函数的定义产生一个典型信号，其数学公式如下：

$$x(t) = K\delta(t - t_0)$$

阶跃信号发生器芯片

阶跃信号发生器芯片的作用是按照阶跃函数的定义产生一个典型信号，其数学公式如下：

$$x(t) = \begin{cases} K, t \geq t_0 \\ 0, t < t_0 \end{cases}$$

斜坡信号发生器芯片

斜坡信号发生器芯片的作用是按照斜坡函数的定义产生一个典型信号，其数学公式如下：

$$x(t) = K * t$$

加速度信号发生器芯片

加速度信号发生器芯片的作用是按照加速度函数的定义产生一个典型信号，其数学公式如下：

$$x(t) = (K \cdot t^2) / 2$$

方波信号发生器芯片

方波信号发生器芯片的作用是按照方波函数的定义产生一个典型信号。

三角波信号发生器芯片

三角波信号发生器芯片的作用是按照三角波函数的定义产生一个典型信号。

白噪声信号发生器芯片

白噪声信号发生器芯片的作用是按照白噪声的定义产生一个典型信号。

系统参数复位芯片

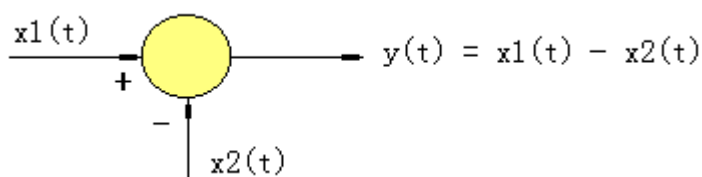
系统参数复位芯片的作用是对系统中的数组型数据线、单值数据线和各典型控制环节芯片的初始值进行复位和清零。点击该芯片就可以对系统参数进行复位，也可以用所连接的开关线控制其运行。

趋势曲线控件芯片

趋势曲线控件芯片用于将各典型控制环节的输出以先进先出的方式存储到一个数组型数据线上，以便于对其输出进行显示和分析。

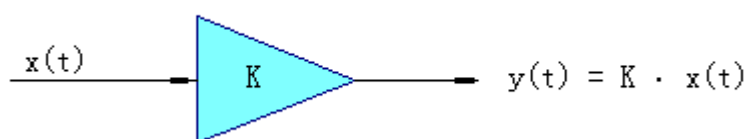
加法器环节芯片

加法器环节芯片的作用是对两路输入信号进行叠加，合成为一路输出信号，如下图所示：



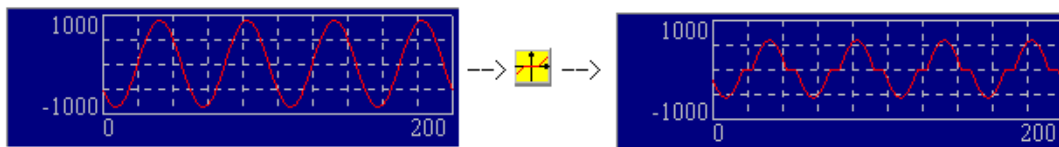
比例环节芯片

比例环节芯片的作用是对输入信号进行放大，然后输出，如下图所示：



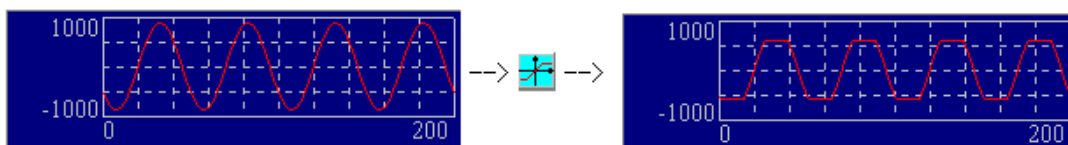
饱和环节芯片

饱和环节芯片的作用是在系统中插入一个非线性饱和环节，对输入信号进行饱和处理，然后输出。如下图所示：



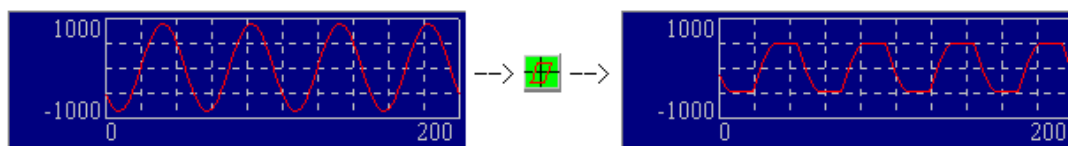
死区环节芯片

死区环节芯片的作用是在系统中插入一个非线性死区环节，对输入信号进行死区处理，然后输出。如下图所示：



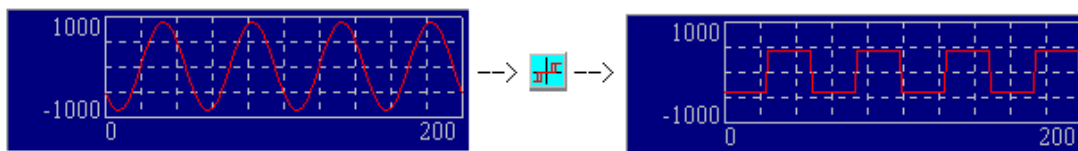
间隙环节芯片

间隙环节芯片的作用是在系统中插入一个非线性间隙环节，对输入信号进行间隙处理，然后输出。如下图所示：



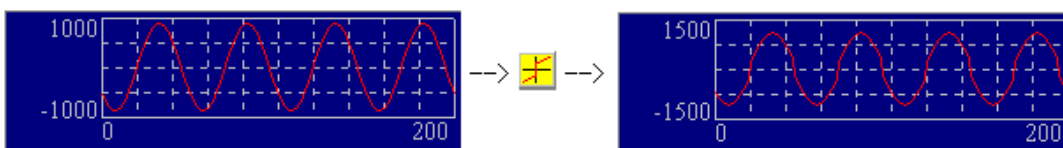
继电环节芯片

继电环节芯片的作用是在系统中插入一个非线性继电环节，对输入信号进行继电处理，然后输出。如下图所示：



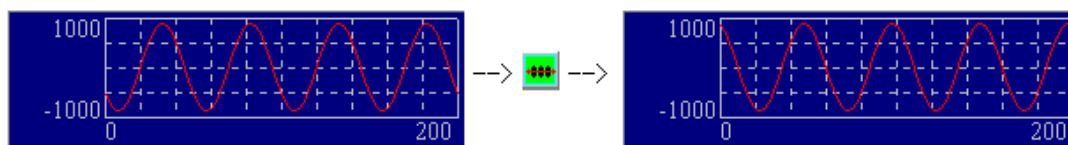
摩擦环节芯片

摩擦环节芯片的作用是在系统中插入一个非线性摩擦环节，对输入信号进行摩擦处理，然后输出。如下图所示：



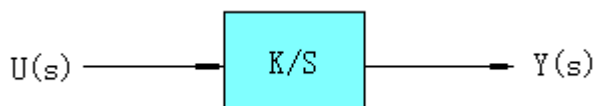
延时环节芯片

延时环节芯片的作用是在系统中插入一个非线性延时环节，对输入信号进行延时处理，然后输出。如下图所示：



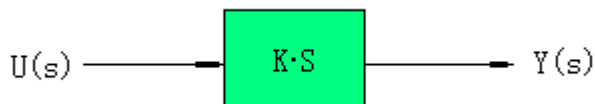
积分环节芯片

积分环节芯片的作用是对输入信号进行积分，然后输出，如下图所示：



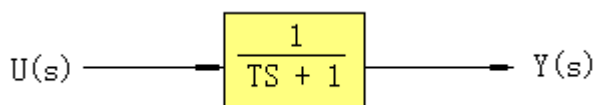
微分环节芯片

微分环节芯片的作用是对输入信号进行微分，然后输出，如下图所示：



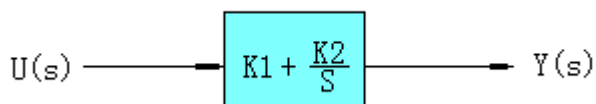
一阶惯性环节芯片

一阶惯性环节芯片的作用是在系统中引入一个惯性单元，其系统传递函数如下：



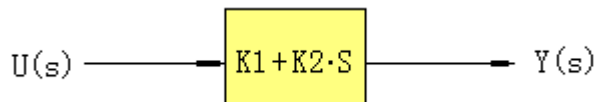
比例积分环节芯片

比例积分环节芯片的作用是在系统中引入一个比例+积分单元，其系统传递函数如下：



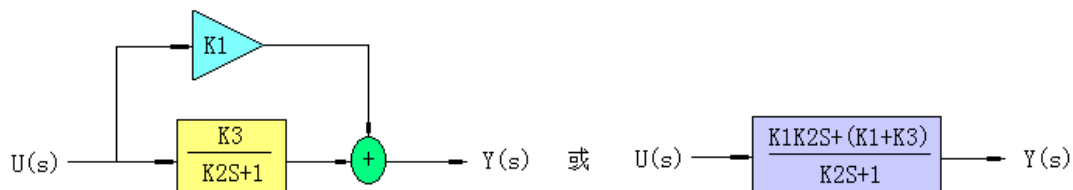
比例微分环节芯片

比例微分环节芯片的作用是在系统中引入一个比例+微分单元，其系统传递函数如下：



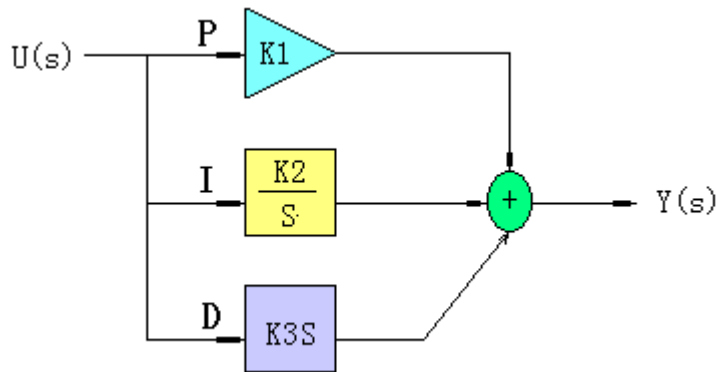
比例+惯性环节芯片

比例+惯性环节芯片的作用和传递函数如下图所示：



PID 环节芯片

PID 环节芯片的作用是对输入信号进行比例+积分+微分处理，然后输出，如下图所示：



时域波形基本参数计算芯片

时域波形基本参数计算芯片的作用是：计算输入到此芯片的波形数组的时域波形基本参数。并通过数据线输出。

频谱运算芯片

频谱运算芯片的作用是：对所连接的数组型输入数据线上的波形数据进行 FFT 变换，计算信号的实频/虚频、功率谱、幅频/相频，并将计算结果输出到两条数组型数据线上。

频率计芯片

频率计芯片的作用是：识别信号中谐波成份的频率、幅值和相位。频率计芯片采用三级 FFT 算法对所连接的数组型输入数据线(软件芯片)上的波形数据谐波分析，识别信号谐波谱峰的频率、幅值和相位，计算结果通过三条数据线进行输出，只要在这些数据线上连接表头等数据显示芯片就可以读出识别结果。

谱窗函数芯片

谱窗函数芯片的作用是：在时域用窗函数对信号进行截断，通过选择不同的窗函数可以减小后续 FFT 频谱计算中的能量泄漏。

FFT 频谱校正芯片

FFT 频谱校正芯片的作用是：对所连接的数组型输入数据线(软件芯片)上的波形数据进行 FFT 变换和频谱校正，计算信号的实频/虚频、功率谱、幅频/相频，并将计算结果输出到两条数组型数据线上(软件芯片)上。

由于信号截断和能量泄漏，FFT 算法计算出的频谱有很大的误差（幅值误差最大可达 36%），为解决此问题，本芯片在 FFT 变换数据的基础上采用三点校正法对谱峰数据进行校正。该芯片对谱峰稀疏的信号有较好的处理效果。

信号相关系数计算芯片

信号相关系数计算芯片的作用是：对所连接的两个数组型数据线上的波形数据进行相关分析，计算信号的自相关系数或互相关系数，并将计算结果输出到另一条数组型数据线上。

概率密度/分布函数芯片

概率密度/分布函数芯片的作用是：对所连接的数组型输入数据线上的波形数据进行概率密度或概率分布计算，并将计算结果输出到一条数组型数据线上。

信号微分/积分运算芯片

信号微分/积分运算芯片的作用是：所连接的数组型输入数据线上的波形数据进行微分或者积分的运算，并将计算结果输出到另一条数组型输出数据线上。

倍频程分析芯片

倍频程分析芯片的作用是：采用 FFT 算法计算声音输入的声音信号/噪声信号的倍频程谱，显示倍频程谱。

频域滤波芯片

频域包络检波芯片的作用是：用频域滤波器对信号进行带通滤波，提取信号中的有用成分。

频域包络检波芯片

频域包络检波芯片的作用是：用频域垂直滤波器对信号进行带通滤波和 Hilbert 变换方法，提取信号中的包络成分。

FIR 数字滤波器芯片

FIR 数字滤波器芯片的作用是：按照设定的 FIR 滤波器系数对信号进行数字滤波，去除信号中的干扰频率成分。

IIR 数字滤波器芯片

IIR 数字滤波器芯片的作用是：按照设定的 IIR 滤波器系数对信号进行数字滤波，去除信号中的干扰频率成分。

FIR/IIR 数字滤波器设计芯片

FIR/IIR 数字滤波器设计芯片的作用是：按照设定的滤波器阶次和上下截止频率对滤波器进行设计，并将滤波器系数和特性通过数据线输出。

连续傅立叶展宽谱分析芯片

连续傅立叶展宽谱芯片的作用是：对选定的频率段进行局部放大，以更高的频率分辨率显示频谱的细节。

ZOOM-FFT 频谱细化分析芯片

ZOOM-FFT 频谱细化分析芯片的作用是：对选定的频率段进行局部放大，以更高的频率分辨率显示频谱的细节。

谱阵芯片

谱阵芯片的作用是：对一个长时间段的信号进行连续观测，分段显示信号的波形/频谱，并以三维谱阵或色图的方式显示，从而在一个长的观测时间段内了解信号波形/频率成分随时间的变化情况。

功率倒频谱运算芯片

功率倒频谱运算芯片的作用是：对所连接的数组型输入数据线上的波形数据进行 FFT 变换，计算信号的对数功率谱，然后再对其进行 FFT 变换，计算信号频谱的功率谱，分析信号频谱中的周期成分。

AR 模型分析芯片

AR 模型分析芯片的作用是：对所连接的数组型输入数据线(软件芯片)上的波形数据进行 AR 模型分析，计算 AR 模型系数和 AR 功率谱。

数组运算芯片

数据运算芯片的作用是：对所连接的两条数组型输入数据线上的波形数据进行四则运算（加、减、乘、除）或者取对数运算，并将计算结果输出到另一条数组型数据线上。可用于复杂波形合成等应用场合。

第四章、DRVI/DRLink 功能扩展

4.1 Signal VBScript 语言

4.1.1 简介

Signal VBScript 是在网页设计中 VBScript 编程语言的基础上针对 DRVI/DRLink 功能扩展需要而扩展的一个内嵌在 DRVI/DRLink 中的在线编程语言，教师和学生可以象设计网页中的 VBScript、JavaScript 小程序那样用 Signal VBScript 设计小程序来扩展 DRVI/DRLink 功能。

如果您已了解 VBScript 或 Visual Basic，就会很快熟悉 Signal VBScript。即使您没有学过这两种语言也可以通过下面的介绍快速学会简单小程序设计。

4.1.2 Signal VBScript 变量和数据类型

与其它编程语言不同，VBScript 只有一种数据类型，称为 Variant。Variant 是一种特殊的数据类型，根据使用的方式，它可以包含不同类别的信息。Variant 用于数字上下文中时作为数字处理，用于字符串上下文中时作为字符串处理。

用户在编程时不需要定义变量类型，变量类型在第一次对该变量赋值时由初始值确定。例如下面是一段 VBScript 程序代码：

```
Dim a,b,c
a=1
b=2.5
c="Hi"
```

其中 Dim 为变量申明语句，变量 a、b 初始化为数字量，c 初始化为字符串。

不同类型的变量不能在一起直接运算，需用 CStr 函数将数字量转换为字符串，或用 CDb1 函数将字符串转换为数字量。

4.1.3 数组变量

数组变量和普通变量是以相同的方式用 Dim 声明的，唯一的区别是声明数组变量时变量名后面带有括号 ()。下例声明了一个包含 5 个元素的一维数组：

```
Dim A(5)
```

虽然括号中显示的数字是 5，但由于在 VBScript 中所有数组都是基于 0 的，所以这个数组实际上包含 6 个元素。在基于 0 的数组中，数组元素数目总是括号中显示的数目加 1。

在数组中使用索引为数组的每个元素赋值。从 0 到 5，将数据赋给数组的元素，如下所示：

```
A(0) = 1
A(1) = 2
A(2) = 3
. . .
A(5) = 6
```

与此类似，使用索引可以检索到所需的数组元素的数据。例如：

```
. . .
x = A(3)
```

数组并不仅限于一维，声明多维数组时用逗号分隔括号中每个表示数组大小的数字。在下例中，Table 变量是一个有 6 行和 11 列的二维数组：

```
Dim MyTable(5, 10)
```

在二维数组中，括号中第一个数字表示行的数目，第二个数字表示列的数目。

4.1.4 VBScript 运算符

VBScript 有一套完整的运算符，包括算术运算符、比较运算符、连接运算符和逻辑运算符。

算术运算符		比较运算符		逻辑运算符	
描述	符号	描述	符号	描述	符号
求幂	^	等于	=	逻辑非	Not
负号	-	不等于	<>	逻辑与	And
乘	*	小于	<	逻辑或	Or
除	/	大于	>	逻辑异或	Xor
整除	\	小于等于	<=	逻辑等价	Eqv
求余	Mod	大于等于	>=	逻辑隐含	Imp
加	+	对象引用比较	Is		
减	-				
字符串连接	&				

4.1.5 使用条件语句

使用条件语句和循环语句可以控制 Script 的流程。使用条件语句可以编写进行判断和重复操作的 VBScript 代码。在 VBScript 中可使用以下条件语句：

If...Then...Else 语句

Select Case 语句

使用 If...Then...Else 进行判断例程：

```
.....
If value = 0 Then
    b=1
End If
.....
If b = 0 Then
    c=1
Else
    c=2
End If
.....
```

使用 Select Case 进行判断例程：

```
Select Case value
    Case 0
        value=1
    Case 1
        value=2
    Case 2
        value=3
    Case Else
        value=4
End Select
```

4.1.6 使用循环语句

循环用于重复执行一组语句。循环可分为三类：一类在条件变为 False 之前重复执行语句，一类在条件变为 True 之前重复执行语句，另一类按照指定的次数重复执行语句。

在 VBScript 中可使用下列循环语句：

Do...Loop: 当（或直到）条件为 True 时循环。

For...Next: 指定循环次数，使用计数器重复运行语句。

使用 Do 循环例程：

```
myNum = 20
```

```
Do While myNum > 10
    myNum = myNum - 1
    counter = counter + 1
```

```
Loop
```

使用 For...Next 例程：

```
Dim j, total
```

```
For j = 2 To 10 Step 2
    total = total + j
```

```
Next
```

4.1.7 使用过程

VBScript 中，过程被分为两类：**Sub 过程**和**Function 过程**。

Sub 过程

Sub 过程是包含在 **Sub** 和 **End Sub** 语句之间的一组 VBScript 语句，执行操作但不返回值。

Sub 过程可以使用参数（由调用过程传递的常数、变量或表达式）。如果 Sub 过程无任何参数，则 Sub 语句必须包含空括号（**()**）。

例如：

```
Sub ConvertTemp(data)
```

```
temp = data/128
```

```
End Sub
```

Function 过程

Function 过程是包含在 Function 和 End Function 语句之间的一组 VBScript 语句。

Function 过程与 Sub 过程类似，但是 **Function 过程可以返回值**。Function 过程通过**函数名**返回一个值，这个值是在过程的语句中赋给函数名的。Function 返回值的数据类型总是 **Variant**。

```
Function Celsius(fDegrees)
```

```
Celsius = (fDegrees - 32) * 5 / 9
```

```
End Function
```

4.1.8 常用 Signal VBScript 标准函数

1. 数学函数

Abs 函数：返回一个数字的绝对值。

调用方法：a=Abs(-100)

Asc 函数：返回对应字符串中**第一个字母的 ANSI 字节码**。

调用方法：a=Asc("ABCD")

Atn 函数：返回一个数字的弧正切值(arctangent)。

调用方法：a=Atn(1)*180/3.14

Cdbl 函数：返回已转换成 Double 型的字符串的值。

调用方法：a=Abs("12.5")

Cos 函数：返回一角度(弧度)的余弦值。

调用方法：a=Cos(60*(3.14/180))

CStr 函数：返回已转换成字符串的数字量的值。

调用方法：a=Cstr(2.56)

Exp 函数：返回 e（自然对数的底数）的某次方。

调用方法：a=Exp(1)

Int 函数：返回数字的整数部分。

调用方法：a=Int(3.25)

Log 函数：返回一个数字的自然对数。

调用方法：a=Log(12)

Randomize：初始化随机数产生器。

调用方法：Randomize Time

Rnd 函数：返回一个随机数(0 到 1)。

调用方法：a=Rnd()

Round 函数：返回已进位到指定小数位的数字。

调用方法：a=Round(2.75678, 2)

Sgn 函数：返回指出数字之正负号的整数。

调用方法：a=Sgn(-11)

Sin 函数：返回一个角度(弧度)的正弦值。

调用方法：a=Sin(60*(3.14/180))

Sqr 函数：返回一个数字的平方根。

调用方法：a=Sqr(9)

Tan 函数：返回一个角度的正切值。

调用方法：a=Tan(1)

Time 函数：返回 Date 子类型的当前系统时间。

调用方法：a=Time()

2. 软件总线接口函数

GWVI.ReadPort：读取与 VB 芯片端口相连的软件总线数据线的值

调用方法：值=GWVI.ReadPort(端口号)

GWVI.WritePort: 设定与 VB 芯片端口相连的软件总线的值

调用方法: GWVI.WritePort 端口号, 设定值

GWVI.ReadInterval: 读取与 VB 芯片端口相连的数组型数据线的的数据点间隔

调用方法: 数据点间隔=GWVI.ReadInterval(端口号)

GWVI.GetArrayStart: 读取与 VB 芯片端口相连的数组型数据线的起始点坐标

调用方法: 数据点间隔=GWVI.GetArrayStart(端口号)

GWVI.ReadPortArray: 读取与 VB 芯片端口相连的数组型数据线上的波形或频谱数据到数组中

调用方法: **GWVI.ReadPortArray 端口号, 读取点数, 数组名**

```
Dim data(2048), data1(2048)
```

```
For K = 0 To 2047
```

```
    data(k)=.....
```

```
Next
```

```
.....
```

```
GWVI.ReadPortArray 1, 2048, data
```

GWVI.WriteInterval: 设定与 VB 芯片端口相连的数组型数据线的的数据点间隔

调用方法: GWVI.WriteInterval 端口号, 数据点间隔

GWVI.SetArrayStart: 设定与 VB 芯片端口相连的数组型数据线的起始点坐标

调用方法: GWVI.SetArrayStart 端口号, 起始点坐标

GWVI.WritePortArray: 用数组值设定与 VB 芯片端口相连的数组型数据线上的波形或频谱数据

调用方法: GWVI.WritePortArray 端口号, 读取点数, 数组名

```
Dim data(2048), data1(2048)
```

```
For K = 0 To 2047
```

```
    data(k)=...
```

```
Next
```

```
.....
```

```
GWVI.WritePortArray 1, 2048, data
```

GWVI.WritePortString: 设定与 VB 芯片端口相连的数据线的字符串数据

调用方法: GWVI.WritePortString 端口号, 字符串

GWVI.ReadPortString: 取与 VB 芯片端口相连的数据线的数据

调用方法: s=GWVI.ReadPortString(端口号)

返回值: 字符串

GWVI.ReadPortArrayData: 取与 VB 芯片端口相连的数组数据线上的一个数据

调用方法: v=GWVI.ReadPortArrayData(端口号, 数据位置)

返回值: 数字

GWVI.WritePortArrayData: 写与 VB 芯片端口相连的数组数据线上的一个数据

调用方法: GWVI.WritePortArrayData 端口号, 数据位置, 值

GWVI.GetMatrixLine: 读与 VB 芯片端口相连的矩阵数据线上的数据

调用方法: GWVI.GetMatrixLine 端口号, 列数, 行数, 存储数组

GWVI.GetMatrixX: 读与 VB 芯片端口相连的矩阵数据线上的数据列数

调用方法: x=GWVI.GetMatrixX(端口号)

返回值: 数字

GWVI.GetMatrixY: 读与 VB 芯片端口相连的矩阵数据线上的数据行数

调用方法: y=GWVI.GetMatrixY(端口号)

返回值: 数字

GWVI.SetMatrixLine: 写与 VB 芯片端口相连的矩阵数据线上的数据

调用方法: GWVI.SetMatrixLine 端口号, 列数, 行数, 存储数组

GWVI.RefreshPort: 刷新与 VB 芯片端口相连的数据线

调用方法: GWVI.RefreshPort 端口号

3. 图形函数

GWVI.WriteText: 在(x, y)用色彩 c 写字符串 Str。

调用方法: GWVI.WriteText x, y, c, "Hi..."

GWVI.WriteTextEx: 在(x, y)用色彩 c 写 h 点高度的字符串 Str。

调用方法: GWVI.WriteTextEx x, y, c, h, "Hi..."

GWVI.DrawLine: 在(x1, y1)到(x2, y2)间用色彩 c 画一条直线。

调用方法: GWVI.DrawLine x1, y1, x2, y2, c

GWVI.FillRectangle: 用色彩 c 填充(x, y)到(x+w, y+h)的矩形区域。

调用方法: GWVI.FillRectangle x, y, w, h, c

GWVI.Fillcircle: 用色彩 c 填充(x, y)为圆心, r 为半径的圆形区域。

调用方法: GWVI.Fillcircle x, y, r, c

GWVI.DrawCircle: 用色彩 c 以(x, y)为圆心, r 为半径画圆。

调用方法: GWVI.DrawCircle x, y, r, c

GWVI.DrawRectangle: 用色彩 c 以(x, y)和(x+w, y+h)为端点画矩形。

调用方法: GWVI.DrawRectangle x, y, w, h, c

GWVI.DrawArc: 用色彩 c 以(x, y)为圆心, r 为半径画圆, a1、a2 为起始角和终止角画圆弧。

调用方法: GWVI.DrawArc x, y, r, a1, a2, c

GWVI.Setcolor: 用红、绿、蓝三原色合成色彩。

调用方法: GWVI.Setcolor red, green, blue

函数返回值: 图形函数中使用的色彩码。

GWVI.GetRGB: 取色彩的红、绿、蓝三原色。type=0/1/2。

调用方法: GWVI.GetRGB(type, color)

函数返回值: 红 / 绿或蓝分量

GWVI.DrawPixel: 用色彩 c, 在(x, y)画点。

调用方法: GWVI.DrawPixel x, y, c

GWVI.Draw3Dline: 用色彩 c1, c2, 在(x1, y1)和(x2, y2)间以间距 d 画 3 维效果的直线

调用方法: GWVI.Draw3DLine x1, y1, x2, y2, d, c1, c2

GWVI.Draw3DEdge: 用色彩 c1, c2, 以(x, y)和(x+w, y+h)为端点画 3 维效果的矩形。

调用方法: GWVI.Draw3DEdge x, y, w, h, c1, c2

GWVI.DrawPicture: 用二维数组 data 在(x0, y0)-(x0+w, y0+h)区域绘制图片。

调用方法: GWVI.DrawPicture x0, y0, w, h, data

4. 常用图形色彩码表:

Value	Color
0	Black
1	Blue
2	Green
3	Cyan
4	Red
5	Magenta
6	Brown
7	Light Gray
8	Dark Gray
9	Light Blue
10	Light Green
11	Light Cyan
12	Light Red
13	Light Magenta
14	Yellow
15	White

常用图形色彩码表

5. 文件操作

1) 文件选择对话框:

描述: 弹出文件选择对话框, 选择读盘或存盘文件名。

语法: `Value=GWVI.FileDialog(long type, LPCTSTR ext)`

参数: `Type=0`:打开读文件对话框, `Type=1`:打开存文件对话框;`ext`=文件类型

返回值: `-1`=取消, `0`=确定, 选择的文件存放在 `GWVI.Buffer` 中

样例:

```
rc=GWVI.FileDialog(0,"*.txt")
If rc = 0 Then
    name=GWVI.Buffer
End If
```

2) 文本文件读/写:

描述: 打开指定的文件并返回一个 `TextStream` 对象, 可以读取此对象或将其追加到文件

语法: `object.OpenTextFile(filename[, iomode[, create[, format]]])`

部分	描述
object	必选。应为 <code>FileSystemObject</code> 对象的名称。
filename	必选。指明要打开的文件名称。
iomode	可选。输入/输出模式, 是下列两个常数之一: <code>1=ForReading</code> 或 <code>2=ForWriting</code> 。
create	可选。Boolean 值, 指出当指定的 <code>filename</code> 不存在时是否能够创建新文件。允许创建新文件时为 <code>True</code> , 否则为 <code>False</code> 。默认值为 <code>False</code> 。
format	可选。 <code>0</code> =ASCII 格式文件, <code>-1</code> =Unicode 格式文件。

写文件样例:

```
Dim fs, f
Set fs=CreateObject("Scripting.FileSystemObject")
Set f = fs.OpenTextFile("c:\temp\test.txt",2,True)
f.WriteLine "嗨, 你好!"
f.Close
```

读文件样例:

```
Dim fs, f
Set fs=CreateObject("Scripting.FileSystemObject")
Set f = fs.OpenTextFile("c:\temp\test.txt",1,False)
textin=f.ReadLine()
f.Close
```

3) 二进制文件读/写:

`GWVI.openfile`: 打开二进制文件用于文件读写

调用方法: `GWVI.openfile type, name`

参数说明: `type= 0`=打开文件用于读操作; `1`=打开文件用于写操作

参数说明: `name=` 文件名和路径

`GWVI.closefile`: 关闭打开的文件流

调用方法: `GWVI.closefile`

`GWVI.readdata`: 读二进制数。

调用方法: `data=GWVI.readdata(type)`

参数说明: type- 0=读一个 char 字符; 1=读一个 short 量; 2=读一个 long 量; 3=读一个 float 量; 4=读一个 double 量

GWVI.writedata: 写二进制数。

调用方法: Document.writedata type, data

参数说明: type- 0=写一个 char 字符; 1=写一个 short 量; 2=写一个 long 量; 3=写一个 float 量; 4=写一个 double 量

参数说明: data - 写入的数据

注: 用户可以用文本文件读/写、二进制文件读写函数编制自己的特殊格式的数据文件读写 VBScript 扩展芯片插件。

6. 端口操作函数

GWVI.Initport: 初始化端口操作。

调用方法: GWVI.Initport()

GWVI.Inport: 从指定端口地址读数据

调用方法: value=GWVI.Inport(address, size)

参数说明: address-端口地址

参数说明: size-读入的字节数, 1=inportbyte, 2=inportword, 3=inportdword

GWVI.Outport: 从指定端口地址写数据

调用方法: GWVI.Outport address, value, size

参数说明: address-端口地址

参数说明: value-写出的数值

参数说明: size-写的字节数, 1=outportbyte, 2=outportword, 3=outportdword

7. 串行口通讯函数

GWVI.OpenCom: 初始化串行口。

调用方法: Rc=GWVI.OpenCom(port, baud)

参数说明: port: 串行口号, 取 1, 2, 3

参数说明: baud: 波特率, 9800, 19200,

返回值: 0=成功, 1=失败

GWVI.CloseCom: 关闭串行口。

调用方法: GWVI.CloseCom()

GWVI.WriteCom: 从指定串行口发送数据。

调用方法: Rc=GWVI.WriteCom(data, len)

参数说明: data-Ascii 格式的字符数据

参数说明: len-发送的字符数据 Byte 数

返回值: 实际发送的字符长度

GWVI.WriteComByte: 从指定串行口发送 1 个字节的数据。

调用方法: Rc=GWVI.WriteComByte(dByte)

参数说明: dByte, 整型变量, 取值区间限制在 0-255.

返回值: 实际发送的字符长度

GWVI.ReadCom: 从指定串行口接收数据。

调用方法: Rc=GWVI.ReadCom(len, timeout)

参数说明: len-接收的最大字符数据 Byte 数

参数说明: timeout-超时, 单位毫秒

返回值: 实际接收的字符长度, 数据存放在 GWVI.Buffer 中

GWVI.ReadComByte: 从指定串行口接收 1 个字节的数据。

调用方法: Rc=GWVI.ReadComByte

返回值: 接收的字节数据值, 取值区间限制在 0-255.

8. 其他函数

GWVI.Delay: 等待函数。

调用方法: GWVI.Delay(msecond)

参数说明: msecond:函数运行等待的时间, 单位好毫秒

GWVI.HTTPCommand: 执行 HTTP 请求。

调用方法: Rc=GWVI.HTTPCommand(url)

返回值: 执行结果。

GWVI.CallBrowser: 跳转到内部浏览器页面, 执行 HTTP 请求。

调用方法: GWVI.CallBrowser(url)

9. 信号分析函数

GWVI.dofft: FFT 变换函数。

调用方法: GWVI.dofft way, length, array1, array2

参数说明: way: 0=正变换, 1=反变换

参数说明: length: FFT 变换数据长度, 取 2^n , 如 256, 512, 1024

参数说明: array1:代入原始数据的实部, 返回变换后信号的实部

参数说明: array2:代入原始数据的实部, 返回变换后信号的实部

GWVI.doDB: 分贝值计算函数。

调用方法: DB=GWVI.doDB(value)

参数说明: value: 计算值

返回值: 计算出的分贝值

GWVI.doap: 复数幅值、相位计算函数。

调用方法: rc=GWVI.doap(way, re, im)

参数说明: way: 0=计算复数的幅值, 1=计算复数的相位。

参数说明: re:复数的实部

参数说明: im:复数的虚部

返回值: 计算出的幅值或相位

GWVI.dorelation: 相关系数计算函数。

调用方法: GWVI.dorelation len, ch1, ch2, array

参数说明: length: 信号数据长度

参数说明: ch1: 代入 1 通道信号数据

参数说明: ch2: 代入 2 通道信号数据

返回值: 若 ch1 和 ch2 的数据源相同, 则计算出的是信号的自相关系数, 若 ch1 和 ch2 的数据源不同, 则计算出的是信号的互相关系数。

GWVI.FreFilter: 频域信号滤波函数。

调用方法: GWVI.FreFilter len, order, fl, fh, array

参数说明: length: 信号数据长度

参数说明: order: 等效模拟滤波器阶次

参数说明: fl: 滤波器下截止频率

参数说明: fh: 滤波器上截止频率

参数说明: array: 代入原始数据, 返回滤波后数据

GWVI.FreEnvelop: 频域 Hilbert 变换信号包络检波函数。

调用方法: GWVI.FreEnvelop len, order, fl, fh, array

参数说明: length: 信号数据长度

参数说明: order: 等效模拟滤波器阶次

参数说明: fl: 滤波器下截止频率

参数说明: fh: 滤波器上截止频率

参数说明: array: 代入原始数据, 返回信号包络检波数据

GWVI.Spectrum: 信号频谱计算函数。

调用方法: GWVI.Spectrum way, length, data, array1, array2

参数说明: way: 0=实频-虚频谱, 1=幅值-相位谱, 3=功率谱

参数说明: length: FFT 变换数据长度, 取 2^n , 如 256, 512, 1024

参数说明: data: 信号数据

参数说明: array1: 依据 way 的值, 返回实频谱/幅值谱/功率谱

参数说明: array2: 依据 way 的值, 返回虚频谱/相位谱

GWVI.ZoomFFT: 信号采用 Zoom FFT 算法计算信号的细化频谱。

调用方法: GWVI.ZoomFFT type, len, Fs, cF, rate, inarray, array1, array2, pararray

参数说明: type: 0=实频-虚频谱, 1=幅值-相位谱, 3=功率谱, 4=对数幅值-相位谱, 5=对数功率谱

参数说明: len: 数据长度

参数说明: Fs: 信号采样频率

参数说明: cF: 频率细化中心频率

参数说明: rate: 频率细化倍数, 小于/等于 len/1024

参数说明: inarray: 信号数据

参数说明: array1: 依据 way 的值, 返回实频谱/幅值谱/功率谱

参数说明: array2: 依据 way 的值, 返回虚频谱/相位谱

参数说明: pararray: pararray[0] 返回细化频率下位置, pararray[0] 返回细化频谱谱线间隔

GWVI.ZoomSpectrum: 信号采用 Chip-Z 算法计算信号的细化频谱。

调用方法: GWVI.ZoomSpectrum type, len, spelen, fl, fh, dt, inarray, array1, array2

参数说明: type: 0=实频-虚频谱, 1=幅值-相位谱, 3=功率谱, 4=对数幅值-相位谱, 5=对数功率谱

参数说明: len: 数据长度

参数说明: spelen: 细化谱线数

参数说明: fl: 频率细化下限频率

参数说明: fh: 频率细化上限频率

参数说明: dt: 信号采样间隔

参数说明: inarray: 信号数据

参数说明: array1: 依据 way 的值, 返回实频谱/幅值谱/功率谱

参数说明: array2: 依据 way 的值, 返回虚频谱/相位谱

GWVI.SetWin: 时域窗函数系数计算函数。将窗函数系数与信号相乘就可以对信号进行加窗处理。

调用方法: GWVI.SetWin type, len, dataarray

参数说明: type: 0: 矩形窗; 设置为 1: Hanning 窗; 设置为 2: Hamming 窗; 设置为 3: BlackMan 窗; 设置为 4: 平顶窗。

参数说明: len: 窗函数长度, 取 2^n , 如 256, 512, 1024

参数说明: dataarray: 返回计算出的窗函数系数

GWVI.FDT: 信号微分/积分运算函数。

调用方法: GWVI.FDT type, amp, len, data

参数说明: type: 0=微分运算, 1=积分运算

参数说明: amp: 微积分常数

参数说明: len: 信号数据长度

参数说明: data: 代入原始数据, 返回滤波后数据

GWVI.FIR: 时域 FIR 信号滤波函数。

调用方法: GWVI.FIR coeflen, coefarray, datalen, dataarray

参数说明: coeflen: FIR 滤波器系数长度

参数说明: coefarray: 滤波器系数数组

参数说明: datalen: 信号数据长度

参数说明: dataarray: 代入原始数据, 返回滤波后数据

GWVI.FIREnvelop: 时域 FIR 包络信号检波函数。

调用方法: GWVI.FIREnvelop coeflen, coefarray0, coefarray90, datalen, dataarray

参数说明: coeflen: FIR 滤波器系数长度

参数说明: coefarray0: 零相位滤波器系数数组

参数说明: coefarray90: 90 度相位滤波器系数数组

参数说明: datalen: 信号数据长度

参数说明: dataarray: 代入原始数据, 返回滤波后数据

GWVI.IIRLowHigh: 时域 IIR 巴特沃斯低通/高通信号滤波函数。

调用方法: GWVI.IIRLowHigh type, order, a, b, c, datalen, dataarray

参数说明: type: 0=低通, 1=高通

参数说明: order:巴特沃斯滤波器阶次
 参数说明: a:各节滤波器 a 系数数组
 参数说明: b:各节滤波器 b 系数数组
 参数说明: c:各节滤波器 c 系数数组
 参数说明: datalen:信号数据长度
 参数说明: dataarray:代入原始数据, 返回滤波后数据

GWVI. IIRBand: 时域 IIR 巴特沃斯带通信号滤波函数。

调用方法: GWVI. IIRBand order, a, b, c, d, e, datalen, dataarray

参数说明: type:0=低通, 1=高通
 参数说明: order:巴特沃斯滤波器阶次
 参数说明: a:各节滤波器 a 系数数组
 参数说明: b:各节滤波器 b 系数数组
 参数说明: c:各节滤波器 c 系数数组
 参数说明: d:各节滤波器 d 系数数组
 参数说明: e:各节滤波器 e 系数数组
 参数说明: datalen:信号数据长度
 参数说明: dataarray:代入原始数据, 返回滤波后数据

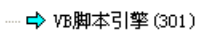
GWVI. dwf: 离散小波变换函数。

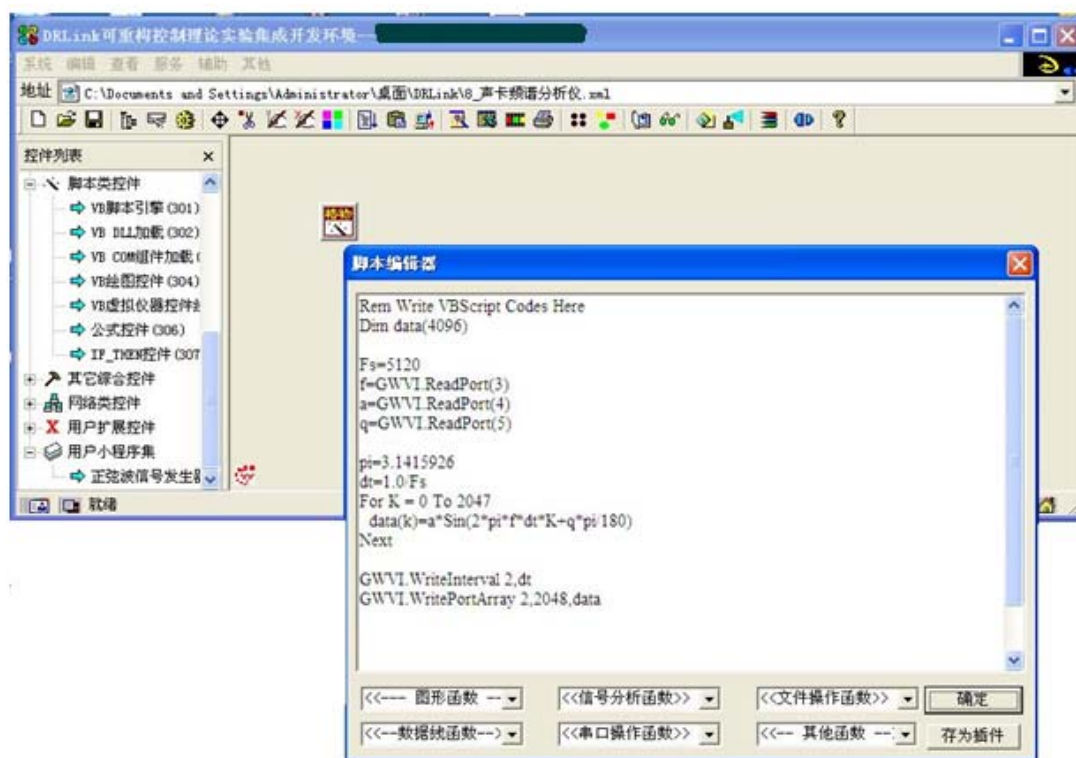
调用方法: GWVI. dwftype, coeflen, c1, c2, scale, datalen, dataarray, w1, w2, w3, w4, w5, w6, w7


参数说明: type:0=离散二进小波变换, 1=离散二进正交小波变换, 2=小波包变换
 参数说明: coeflen:小波滤波器系数长度
 参数说明: c1:低通小波滤波器系数数组
 参数说明: c2:高通小波滤波器系数数组
 参数说明: scale:小波变换尺度数, 0<取值≤6
 参数说明: datalen:信号数据长度
 参数说明: dataarray:原始数据
 参数说明: w1:第一阶小波系数数组
 参数说明: w2:第二阶小波系数数组
 参数说明: w3:第三阶小波系数数组
 参数说明: w4:第四阶小波系数数组
 参数说明: w5:第五阶小波系数数组
 参数说明: w6:第六阶小波系数数组
 参数说明: w7:第七阶小波系数数组

4.2 用 Signal VBScript 编制软件芯片插件

Signal VBScript 通常不作为一个独立的语言来使用, 而是设计含 Signal VBScript 小程序的如那件芯片扩展插件来扩展 DRVI/DRlink 中软件的种类和功能。

DRVI/DRlink 中提供了用于存放 Signal VBScript 小程序的 VBScript 脚本程序的芯片。点击控件栏中的脚本类控件下的“VB 脚本引擎”  图标就可以在软件总线上插入一片 VBScript 脚本程序芯片。在芯片上点击鼠标左键, 弹出小程序编辑窗口, 在其中写入 Signal VBScript 小程序就可以形成一个 Signal VBScript 小程序软件芯片扩展插件。

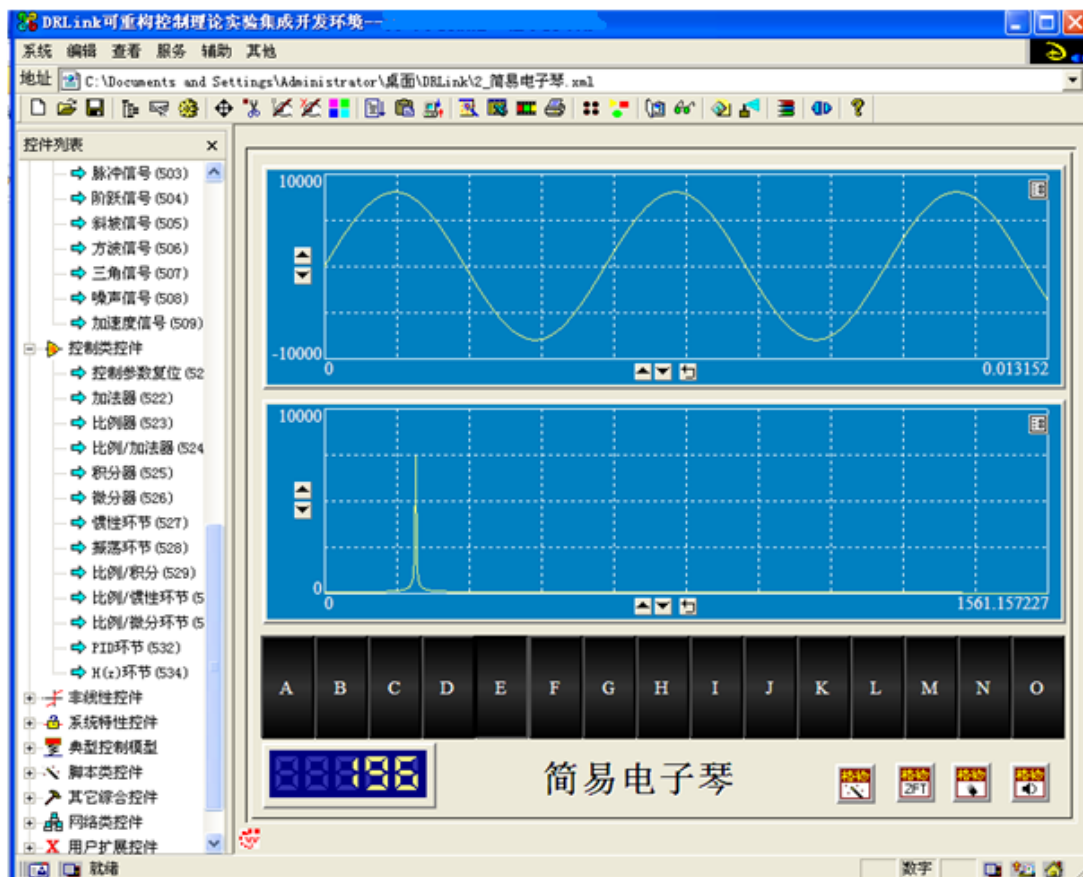


 插件可以接受其它芯片发来的工作命令，也可以与软件总线的一条数据线（或数组型数据线）相连，靠该数据线上的数据流动来驱动。

下面给出两个用 Signal VBScript 设计的软件芯片扩展插件的例子。

A. 简易电子琴设计

下图是 DRVI/DRLink 设计的简单电子琴，它用一片数字信号发生器产生正弦信号来作为标准存音信号、用一片计算机喇叭放音芯片放音、用一片波形/频谱曲线显示芯片显示信号波形、用一片多联开关来作为琴键。问题的难点是压下 A、B、...、O 琴键要分别产生 131、147、...、523Hz 的正弦波信号。多联开关芯片输出的是 0、1、2、...、14，很难通过 DRVI/DRLink 已有芯片将其转换成为 131、147、...、523 并作为数字信号发生器的输出频率。



这个问题用 Signal VBScript 设计一个扩展插件来解决则很简单，插件功能是读取多联开关芯片输出数据线上的数据，然后用条件语句将其转换为 131、147、...、494 频率值并产生波形到数据线上。

程序代码如下所示：

```
<?xml version="1.0" encoding="GB2312" ?>
<DRVI4.1>
  <WaveIC >
    <ICName value="1282634488"> </ICName>
    <X0 value="22"> </X0>
    <Y0 value="32"> </Y0>
    <LengthOfShow value="645"> </LengthOfShow>
    <High value="180"> </High>
    <BgColor value="12615680"> </BgColor>
    <FGColor value="16777215"> </FGColor>
    <CurveColor value="8454143"> </CurveColor>
    <Maximum value="10000.000000"> </Maximum>
    <Minimum value="-10000.000000"> </Minimum>
    <Left value="50"> </Left>
    <Type value="0"> </Type>
    <YArrayLine value="9"> </YArrayLine>
    <XArrayLine value="0"> </XArrayLine>
```



```

</WaveIC>
<WaveIC  >
  <ICName value="1282634494"> </ICName>
  <X0 value="22"> </X0>
  <Y0 value="220"> </Y0>
  <LengthOfShow value="645"> </LengthOfShow>
  <High value="180"> </High>
  <BgColor value="12615680"> </BgColor>
  <FGColor value="16777215"> </FGColor>
  <CurveColor value="8454143"> </CurveColor>
  <Maximum value="10000.000000"> </Maximum>
  <Minimum value="0.000000"> </Minimum>
  <Left value="50"> </Left>
  <Type value="0"> </Type>
  <YArrayLine value="10"> </YArrayLine>
  <XArrayLine value="0"> </XArrayLine>
</WaveIC>
<EdgeIC  >
  <ICName value="1282634839"> </ICName>
  <X0 value="9"> </X0>
  <Y0 value="18"> </Y0>
  <Width value="667"> </Width>
  <High value="542"> </High>
  <LightColor value="16777215"> </LightColor>
  <DarkColor value="0"> </DarkColor>
  <LineWidth value="1"> </LineWidth>
  <WidthOfEdge value="4"> </WidthOfEdge>
  <Type value="3"> </Type>
</EdgeIC>
<FontIC  >
  <ICName value="1282634978"> </ICName>
  <X0 value="245"> </X0>
  <Y0 value="508"> </Y0>
  <Width value="192"> </Width>
  <High value="32"> </High>
  <Caption value="简易电子琴"> </Caption>
  <FGColor value="0"> </FGColor>
  <ShadowColor value="16777215"> </ShadowColor>
  <BGColor value="14215660"> </BGColor>
  <FontWidthHigh value="0.700000"> </FontWidthHigh>
  <Direction value="0"> </Direction>
  <FontProperty value="0"> </FontProperty>
  <ShadowEffect value="1"> </ShadowEffect>
  <BusDataLine value="-1"> </BusDataLine>

```



```

</FontIC>
<RAMIC  >
  <ICName value="1282635063"> </ICName>
  <X0 value="575"> </X0>
  <Y0 value="511"> </Y0>
  <Line1 value="11"> </Line1>
  <Line2 value="-1"> </Line2>
</RAMIC>
<MultButtonIC  >
  <ICName value="1282636360"> </ICName>
  <X0 value="21"> </X0>
  <Y0 value="409"> </Y0>
  <Width value="43"> </Width>
  <High value="83"> </High>
  <Caption value="A#B#C#D#E#F#G#H#I#J#K#L#M#N#O#P#"> </Caption>
  <FGColor value="16777215"> </FGColor>
  <BGColor value="0"> </BGColor>
  <NumofStatus value="15"> </NumofStatus>
  <Direction value="0"> </Direction>
  <Value value="4"> </Value>
  <BusDataLine value="7"> </BusDataLine>
  <Font value="11,1,0,1"> </Font>
</MultButtonIC>
<LEDIC  >
  <ICName value="1282636579"> </ICName>
  <X0 value="22"> </X0>
  <Y0 value="495"> </Y0>
  <Width value="140"> </Width>
  <High value="53"> </High>
  <FGColor value="14"> </FGColor>
  <BGColor value="1"> </BGColor>
  <LengthOfLed value="5"> </LengthOfLed>
  <Align value="1"> </Align>
  <Value value="196.000000"> </Value>
  <BusDataLine value="8"> </BusDataLine>
</LEDIC>
<Script  >
  <ICName value="1282636670"> </ICName>
  <Position value="482,512,385,526,32,32,7,0"> </Position>
  <Parameter1 value="-1,8,9,-1,-1,-1,-1,-1"> </Parameter1>
  <script>

```

<![CDATA[Rem Write VBScript Codes Here

```
a=GWVI.ReadPort(1)
```

```
f=0
If a=0 Then
    f=131
End If
If a=1 Then
    f=147
End If
If a=2 Then
    f=165
End If
If a=3 Then
    f=175
End If
If a=4 Then
    f=196
End If
If a=5 Then
    f=220
End If
If a=6 Then
    f=247
End If
If a=7 Then
    f=262
End If
If a=8 Then
    f=294
End If
If a=9 Then
    f=330
End If
If a=10 Then
    f=349
End If
If a=11 Then
    f=392
End If
If a=12 Then
    f=440
End If
If a=13 Then
    f=494
End If
```

```

If a=14 Then
    f=523
End If
GWVI.WritePort 3,f
Dim x(44000)
L=22000
Fs=44100
dt=1.0/Fs
A=8000.0
Pi=3.14
v=2*pi*dt*f
For K = 0 To L
    x(k)=A*Sin(v*K)
Next
GWVI.WritePortArray1 4,22000,dt,0,x

```

]]></script>

```

</Script>
<ZLineIC  >
    <ICName value="1282636681"> </ICName>
    <Parameters value="7,1282636360,1,1282636670,1,1337293"> </Parameters>
</ZLineIC>
<ZLineIC  >
    <ICName value="1282636785"> </ICName>
    <Parameters value="8,1282636670,3,1282636579,1,1337293"> </Parameters>
</ZLineIC>
<ZLineIC  >
    <ICName value="1282636788"> </ICName>
    <Parameters value="9,1282636670,4,1282634488,1,1337293"> </Parameters>
</ZLineIC>
<Fourier  >
    <ICName value="1282636844"> </ICName>
    <Position value="530,511,407,530,32,32,1"> </Position>
    <Parameter1 value="16384,9,10,11,1,16,-1,-1"> </Parameter1>
    <Parameter2 value="100.000000,4.000000"> </Parameter2>
</Fourier>
<ZLineIC  >
    <ICName value="1282636868"> </ICName>
    <Parameters value="9,1282634488,1,1282636844,1,1337293"> </Parameters>
</ZLineIC>
<ZLineIC  >
    <ICName value="1282636871"> </ICName>
    <Parameters value="10,1282636844,2,1282634494,1,1337293"> </Parameters>
</ZLineIC>
<ZLineIC  >

```

```
<ICName value="1282636877"> </ICName>
<Parameters value="11,1282636844,3,1282635063,1,1337293"> </Parameters>
</ZLineIC>
<SoundPlay >
  <ICName value="1282636960"> </ICName>
  <Position value="622,511,541,517,32,32"> </Position>
  <Parameter1 value="1,22000,9,-1,0"> </Parameter1>
</SoundPlay>
<ZLineIC >
  <ICName value="1282637008"> </ICName>
  <Parameters value="9,1282634488,1,1282636960,1,1337293"> </Parameters>
</ZLineIC>
```

</DRVI4.1>

B.波形合成实验设计

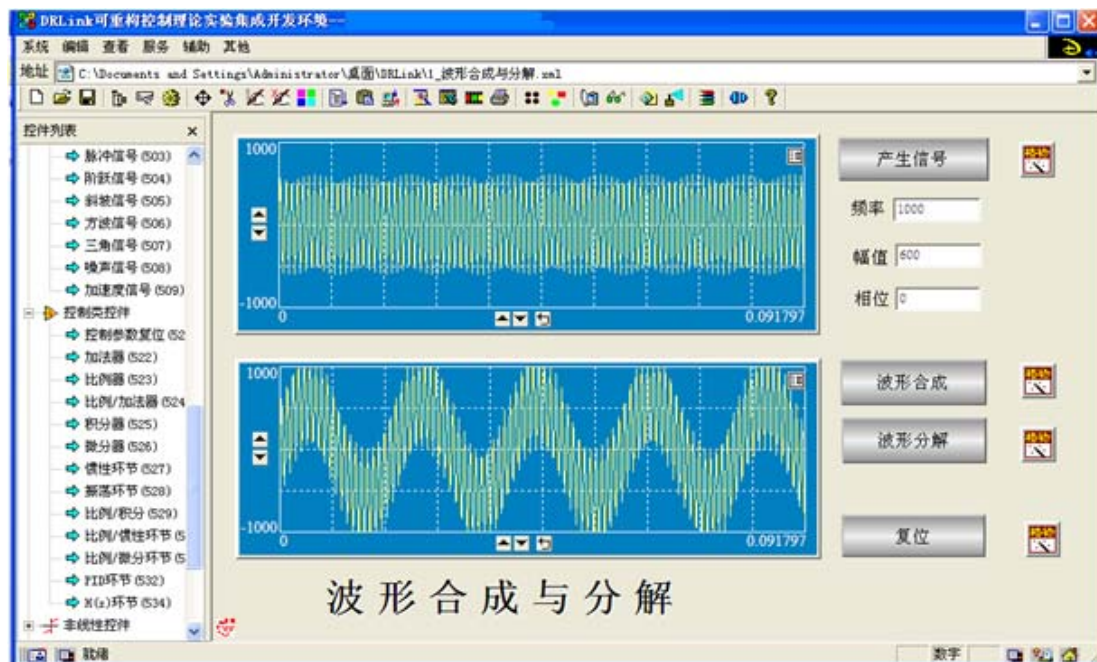
波形合成与分解是教学中常用到的一个实验。

按傅里叶分析原理，任何周期信号都可以用一组三角函数 $\{\sin(2\pi nf_0t), \cos(2\pi nf_0t)\}$ 的组合表示：

$$X(t) = a_0/2 + a_1 \sin(2\pi nf_0t) + b_1 \cos(2\pi nf_0t) + a_2 \sin(2\pi nf_0t) + b_2 \cos(2\pi nf_0t) + \dots$$

也就是说，我们可以利用一组正弦波和余弦波来合成任意形状的周期信号。

该实验需要设计一个波形合成器，让学生动手用标准正弦波信号合成方波、三角波、拍波等信号波形。下面是我们用 DRVI/DRlink 设计的该实验：



其中用到了三片 Signal VBScript 扩展插件，第一片用来产生频率相位标准的正弦波信号，器代码如下：

```

Dim data(4096)
Fs=5120
f=GWVI.ReadPort(2)
a=GWVI.ReadPort(3)
q=GWVI.ReadPort(4)
pi=3.1415926
dt=1.0/Fs
For K = 0 To 2047
    data(k)=a*Sin(2*pi*f*dt*K+q*pi/180)
Next
GWVI.WritePortArray1 5,2048,dt,0,data
第二片用来进行波形累加合成
Dim data(4096),data1(4096)
dt=GWVI. ReadInterval(2)
GWVI.ReadPortArray 2,2048,data
GWVI.ReadPortArray 3,2048,data1
For K = 0 To 2047
    data(k)=data(k)+data1(k)
Next
GWVI.WritePortArray1 3,2048,dt,0,data
第三片用来清除合成结果
Dim data(4096)
dt=GWVI. ReadInterval(2)
For K = 0 To 2047
    data(k)=0
Next
GWVI.WritePortArray1 2,2048,dt,0,data
GWVI.WritePortArray1 3,2048,dt,0,data

```

4.3 用 Visual C++ 编制动态链接库型插件

为了提高系统的开放性和扩展性，DRVI/DRlink 采用浏览器中的功能扩展插件技术，提供了一个动态链接库（DLL）扩展插件接口，用户可以自己设计动态链接库（DLL）型的扩展插件，扩展 DRVI/DRlink 的虚拟仪器芯片集。

为了实现与 DRVI/DRlink 的插接，每个动态链接库（DLL）扩展插件需定义如下四个标准接口函数（VC 形式）：

```

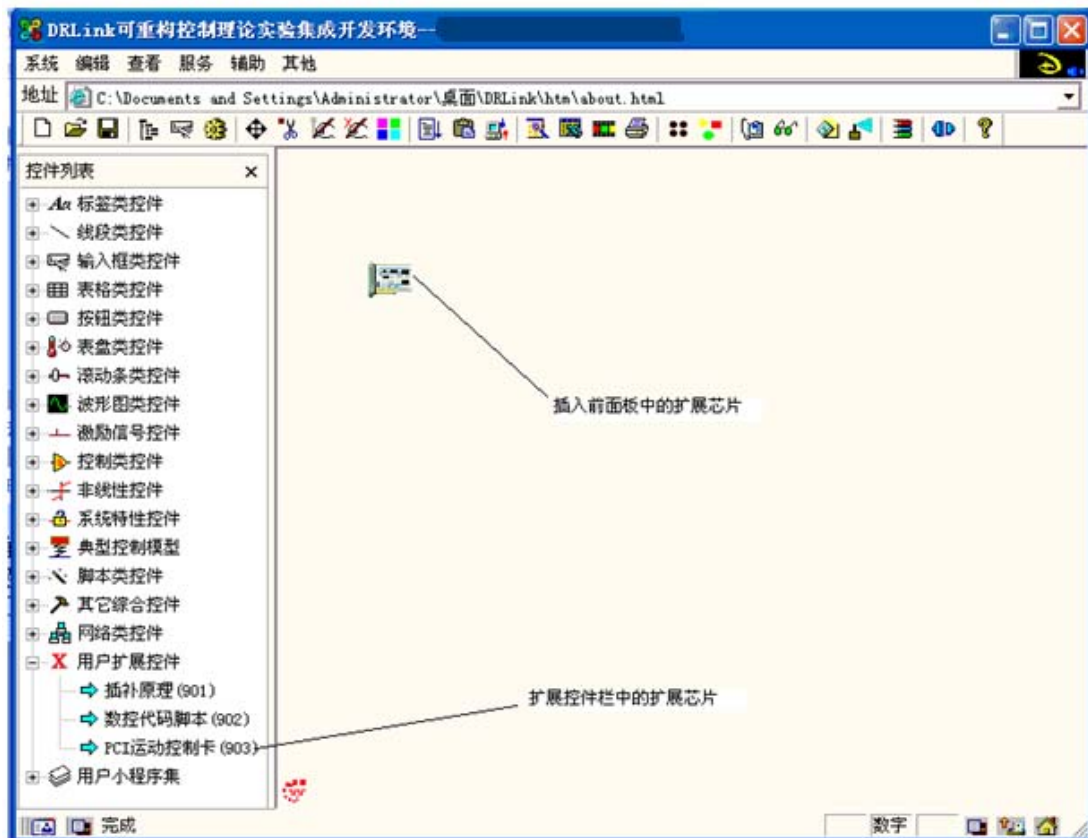
extern "C" void* PASCAL EXPORT addAPI
(long id, char *p1, char *p2, char *p3, char *p4, char *p5, char *p6, char *p7, char *p8,
char *p9, char *p10, char *p11, char *p12, char *p13, char *p14, char *p15, char *p16,
char *p17, char *p18, char *p19, char *p20, char *p21, char *p22, char *p23, char *p24,
char *p25, char *p26, char *p27, char *p28, char *p29, char *p30, char *p31, char *p32
);
extern "C" void PASCAL EXPORT saveAPI(void *hl);
extern "C" int PASCAL EXPORT delAPI(void *hl, int id);
extern "C" void PASCAL EXPORT installVI(char *title);

```

所设计的动态链接库只要添加了这四个标准接口函数就可以作为扩展插件添加到 DRVI/DRlink

扩展插件工具条中,但若要与DRVI/DRLink总线交换数据则需要在程序体中调用总线接口函数。若要接受总线驱动的驱动则需要在程序体中添加对WM_COMMAND事件的支持。

DRVI/DRLink提供了用Visual C++设计的软件芯片样例,这种扩展空间在使用方法上和灵活程度、方便程度上与DRVI/DRLink提供的软件芯片完全一样。



动态链接库插件设计是一个很专业的软件设计工作,但具体实现起来并不复杂。下面介绍在公司模板下使用VS2010开发Visual C++来设计动态链接库的一种方法。

A) 首先,从光盘中复制DRVI/DRLink控件模板到本地计算机中。

B) 打开VS2010,点击打开项目。选择复制的DRVI/DRLink控件模板。

C) 在DRVI/DRLink控件模板中,在源代码中DRVI/DRLink的控件接口已经添加完成。

点开Run.cpp文件:

1. `BOOL CRun::Create(char *winpos, char *p1, char *p2, char *p3, char *p4, char *p5, char *p6, HWND hWnd, UINT nID)` 函数在控件被拖入到前面板中的时候被DRVI/DRLink调用。

2. `void CRun::OnLButtonDown(UINT nFlags, CPoint point)` 在控件上点击左键该函数即被调用。

3. `void CRun::OnRButtonDown(UINT nFlags, CPoint point)` 在控件上点击右键该函数即被调用。

4. `BOOL CRun::OnCommand(WPARAM wParam, LPARAM lParam)` 该函数接受与DRVI/DRLink之间的数据交换。

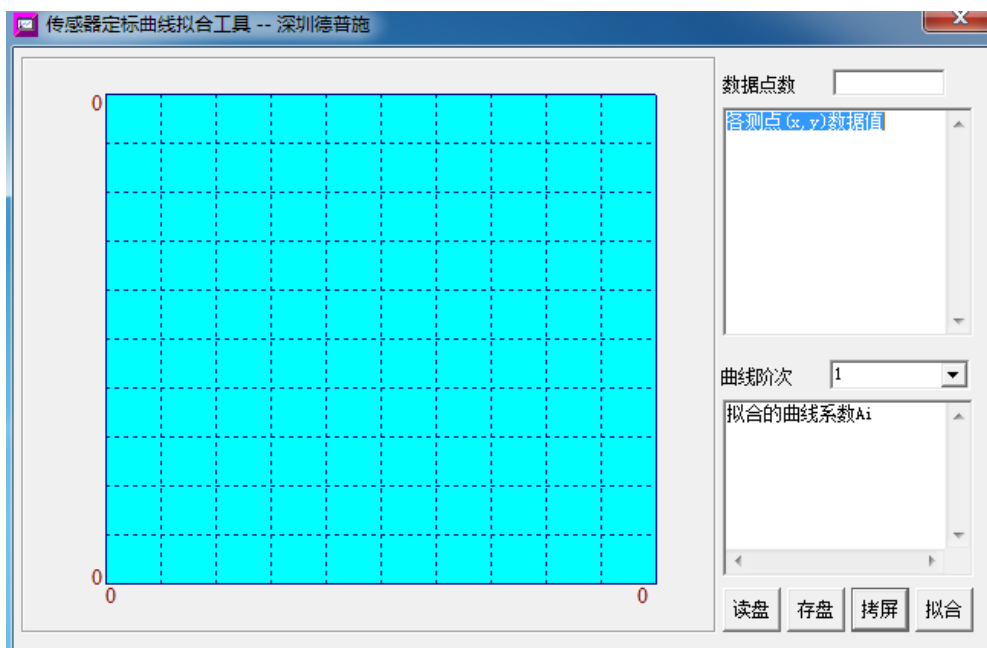
5. `void CRun::OnPaint()` 该函数在前面板上绘制控件的样式。

D) 用户自行在程序相对应的函数中添加自己的处理函数修改部分代码,并将程序编译链接为动态链接库,将编译好的DLL文件拷贝到DRVI/DRLink目录下,选择用户扩展工具栏即可看到自己编写的控件。

第五章、相关工具使用说明

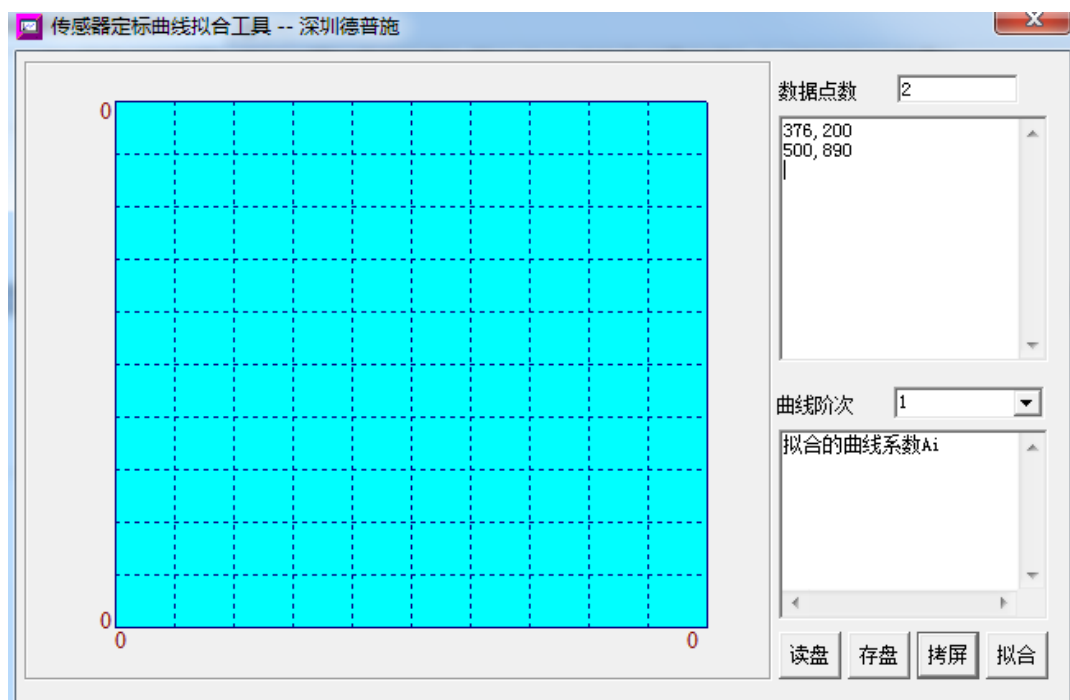
5.1 DRVI/DRLink 传感器定标曲线拟合工具使用说明

点击 DRVI/DRLink 中的“传感器定标曲线拟合工具”菜单项就可以弹出 DRVI/DRLink 中的传感器定标曲线拟合工具，如下图所示：



1) 向定标数据输入窗口中输入定标数据，其中 X 为 A/D 可测量数据，Y 为对应的工程量。

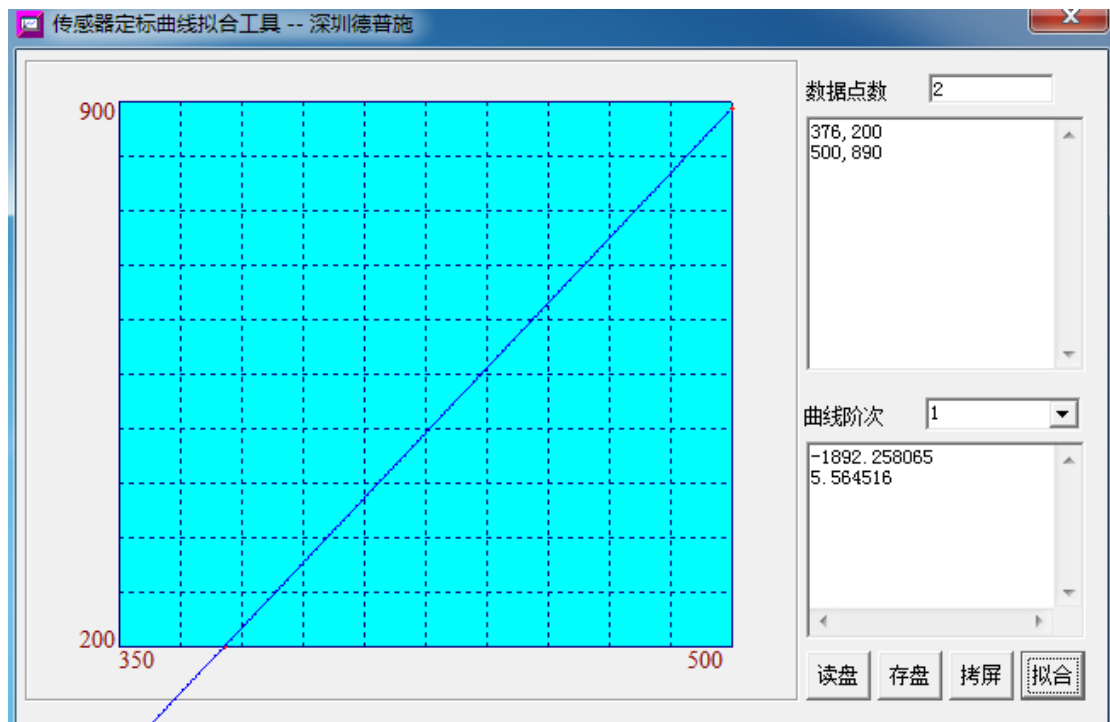
例如，对力传感器进行标定，放上 200g 砝码时，测量值为 376；放上 500g 砝码时，测量值为 890；起数据输入形式如下：



2) 点击拟合按钮，定标曲线采用线性多项式进行拟合：

$$Y=A0+A1*X+A2*X^2+A3*X^3+.....$$

添入拟合曲线阶次，点击“拟合”按钮就可以得到定标曲线系数。将其带入定标曲线芯片中就可以对测量值进行定标和工程单位转换。再点击“拟合曲线”显示拟合曲线效果。

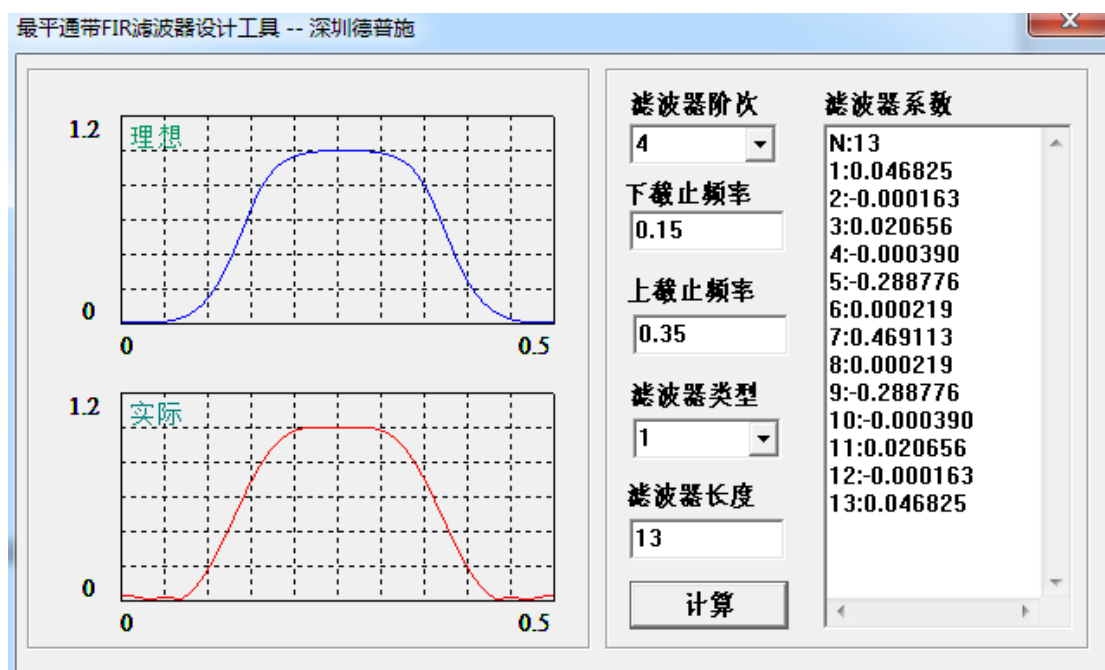


5.2 DRVI/DRLink 数字滤波器设计工具使用说明

点击 DRVI/DRLink 中的“**数字滤波器设计工具**”就可以弹出 DRVI/DRLink 数字滤波器设计工具菜单，如下图所示：



1) FIR 滤波器设计 (优化算法): 点击该条目弹出用优化算法设计 FIR 滤波器系数模块, 如下图所示:



滤波器阶次: 理想滤波器阶次, 决定了滤波器过渡带的衰减率, 取值 1-20。

下截止频率: 理想滤波器归一化下截止频率, 取值 0-0.5。

上截止频率: 理想滤波器归一化上截止频率, 取值 0-0.5。

滤波器类型: 设计时选用的 FIR 滤波器模型类型。

滤波器类型: 1=奇数系数长度, 对称模型。

滤波器类型: 2=偶数系数长度, 对称模型。

滤波器类型: 3=奇数系数长度, 反对称模型。

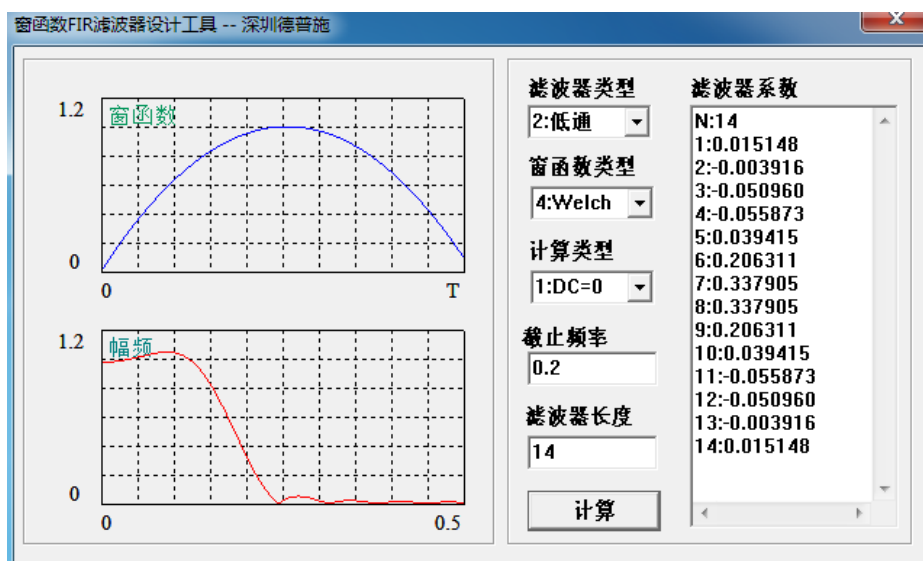
滤波器类型: 4=偶数系数长度, 反对称模型。

滤波器长度: 所设计的 FIR 滤波器系数长度。

输入设计参数后, 点击“计算”按钮就可以设计出所需的 FIR 滤波器, 将“滤波器系数”编辑窗中的数据拷贝到剪贴板上, 然后粘贴到 FIR 滤波器芯片中就可以对信号进行滤波运算。

2) FIR 滤波器设计 (窗函数法): 点击该条目弹出门窗函数算法设计 FIR 滤波器系数模块, 如下图所示

示:



滤波器类型：1：低通，2：高通

窗函数类型：1：Rectangle，2：Parzen，3：Hanning，4：Welch，5：Hamming，6：Blackman

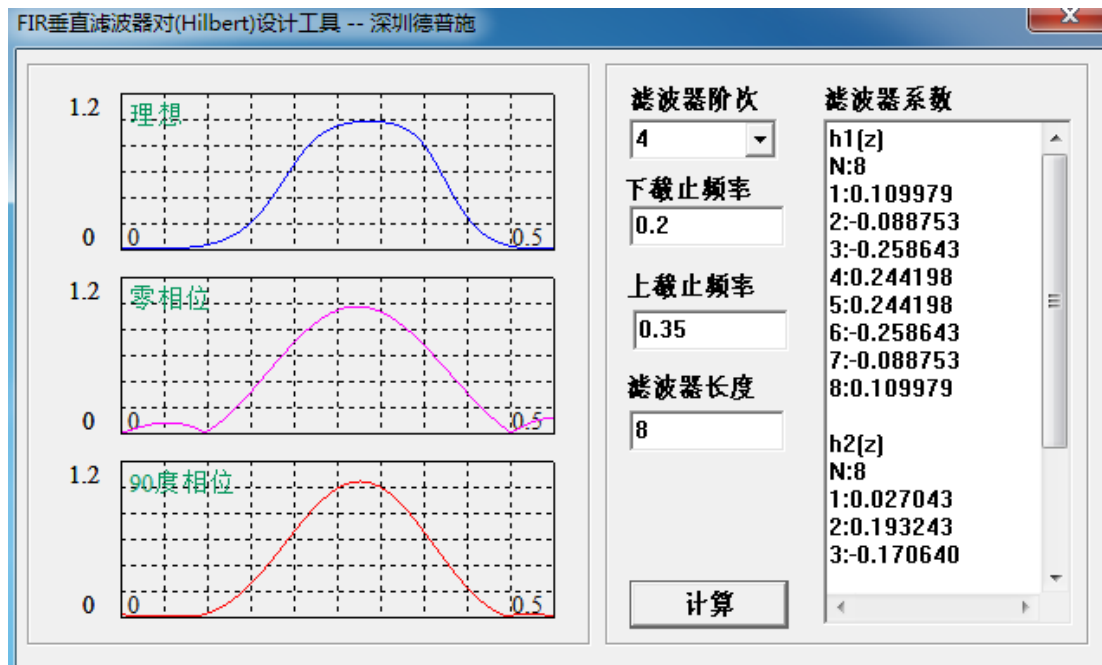
计算类型：1：DC=0，2：DC=1

截止频率：所涉及的低通、高通滤波器截止频率

滤波器长度：所设计的 FIR 滤波器系数长度。

输入设计参数后，点击“计算”按钮就可以设计出所需的 FIR 滤波器，将“滤波器系数”编辑窗中的数据拷贝到剪贴板上，然后粘贴到 FIR 滤波器芯片中就可以对信号进行滤波运算。

3) FIR 垂直滤波器设计：点击该条目弹出 FIR 垂直滤波器系数模块，如下图所示：



滤波器阶次：理想滤波器阶次，决定了滤波器过渡带的衰减率，取值 1-20。

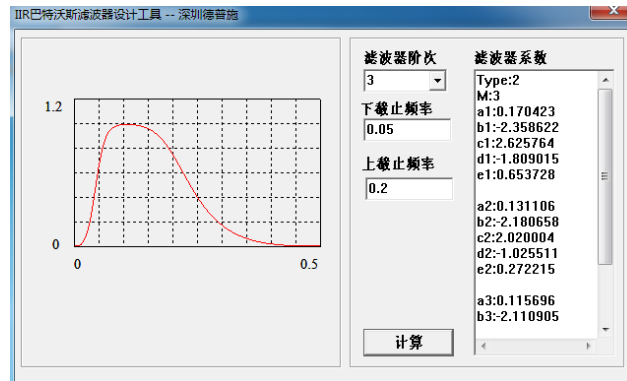
下截止频率：理想滤波器归一化下截止频率，取值 0-0.5。

上截止频率：理想滤波器归一化上截止频率，取值 0-0.5。

滤波器长度：所设计的 FIR 滤波器系数长度。

输入设计参数后，点击“计算”按钮就可以设计出所需的 FIR 垂直滤波器。FIR 垂直滤波器有零相位滤波器 $h_1(z)$ 和 90° 相位滤波器 $h_2(z)$ 组成，他们构成了 Hilbert 变换，可以对信号进行包络检波处理。将“滤波器系数”编辑窗中的数据拷贝到剪贴板上，然后粘贴到包络分析芯片中就可以对信号进行包络检波运算。

4) IIR 巴特沃斯滤波器设计：点击该条目弹出 IIR 巴特沃斯滤波器系数模块，如下图所示：



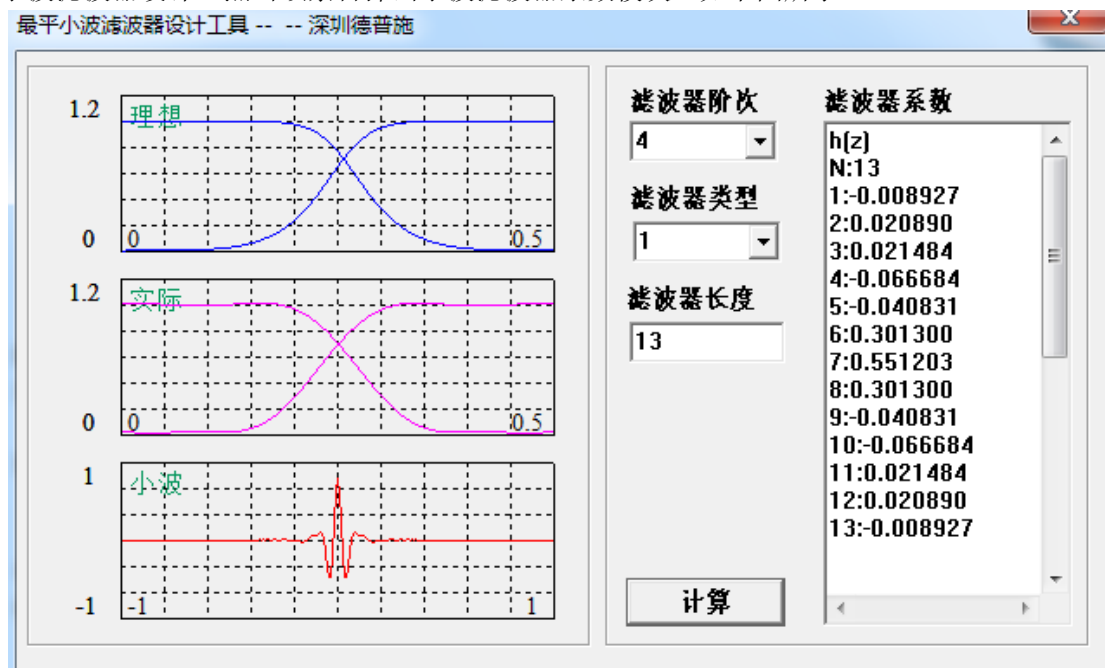
滤波器阶次：Butterworth 滤波器阶数，决定了滤波器过渡带的衰减率，取值 1-20。

下截止频率：理想滤波器归一化下截止频率，取 0-0.5。

上截止频率：理想滤波器归一化上截止频率，取 0-0.5。

输入设计参数后，点击“计算”按钮就可以设计出所需的 Butterworth 滤波器。将“滤波器系数”编辑窗中的数据拷贝到剪贴板上，然后粘贴到 Butterworth 滤波器芯片中就可以对信号进行包络检波运算。

5) 小波滤波器设计：点击该条目弹出小波滤波器系数模块，如下图所示：



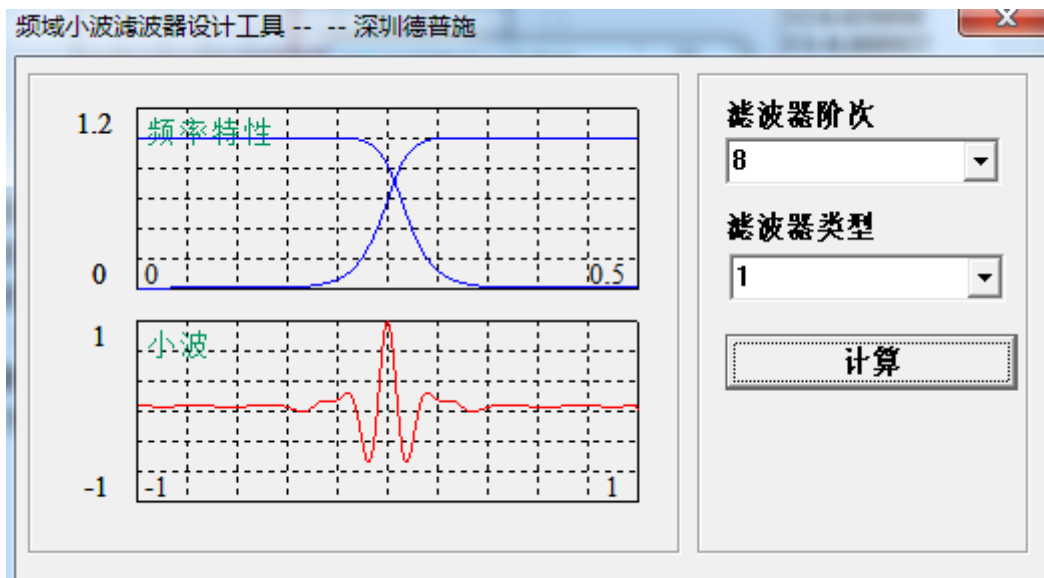
滤波器阶次：理想滤波器阶次，决定了滤波器过渡带的衰减率，取值 1-20。

滤波器类型：1：对称小波，2：反对称小波

滤波器长度：所设计的小波滤波器系数长度。

输入设计参数后，点击“计算”按钮就可以设计出所需的小波滤波器。将“滤波器系数”编辑窗中的数据拷贝到剪贴板上，然后粘贴到小波分析芯片中就可以对信号进行小波变换。

6) 频域小波滤波器设计：点击该条目弹出频域小波滤波器系数模块，如下图所示：



滤波器阶次：理想滤波器阶次，决定了滤波器过渡带的衰减率，取值 1-20。

滤波器类型：1：对称小波，2：反对称小波，3：复数小波

输入设计参数后，点击“计算”按钮就可以设计出所需的频域小波滤波器。将选定的滤波器阶次、滤波器类型输入到频域小波分析芯片中就可以对信号进行小波变换。

5.3 DRVI/DRLink 帮组信息浏览器使用说明

点击 DRVI/DRLink 快捷工具栏中的帮助  按钮 就可以弹出 DRVI/DRLink 帮助信息浏览器，显示用 CHM 格式书写的帮助信息，如下图



DRVI/DRLink 帮助信息浏览器是在标准 chm 帮助格式的浏览器，他继承自微软标准帮助浏览器

模板，同时添加了与 DRVI/DRLink 的互动操作。

用 DRVI/DRLink 帮助信息浏览器浏览含 DRVI/DRLink 脚本文件和链接相关网站、网页十分方便。它可以与 DRVI/DRLink 控件帮助自动关联。由于 DRVI/DRLink 是标准 CHM 帮助的继承，其操作方法非常简单在此不做叙述。

5.4 AVI 教学短片制作工具使用说明

AVI 教学短片制作工具的作用是：在 DRVI/DRLink 前面板上添加一个 AVI 文件制作工具条，将 DRVI/DRLink 中实验脚本的运行过程以 AVI 动画短片的方式记录下来，以便于以多媒体的方式在课堂上进行讲解。

AVI 文件制作工具条



- 1.新建 AVI 教学短片文件，并记录数据流：选择新建的 AVI 文件名和图像压缩方式，一般选 Microsoft Video 1.
 - 2.手动添加一幅图片到打开的 AVI 数据流中：手动添加 DRVI/DRLink 前面板上当前的显示图片到打开的 AVI 图像流中。
 - 3.打开自动屏幕捕获开关：开启图像自动捕获开关，由关联的数据线进行自动图像捕获，并添加到打开的 AVI 图像流中
 - 4.关闭自动屏幕捕获开关：关闭图像自动捕获开关。
 - 5.保持 AVI 数据流到 AVI 文件：关闭打开的 AVI 图像流，并将其保存到文件中。
- 播放制作好的 AVI 文件：打开制作完成的 AVI 文件，预览其效果。

国家技术创新基金项目

方便灵活的可视化编程

强大的兼容性与开放性

远程信号采集设备共享

国家级精品课程核心内容