

学 号 14284060xx  
等 第

# 苏州大学实验报告

## Modbus 仪表通信

院(系)名称: 电子信息学院

专业名称: 14 通信工程(嵌入式培养)

学生姓名: 某某某

课程名称: Linux 操作系统

2015-2016 学年第一学期

## 摘要

本设计利用虚拟仪器的软件开发工具 LabWindows/CVI 通过对串口读或写等操作与 ZW1402 仪表进行通信，设计一个可以自动化测量及数据处理的仪表通信系统。

本设计主要包括系统的用户界面设计及功能的实现过程。利用 LabWindows/CVI 创建图形用户界面，再通过 C 语言实现对各控件相应功能的处理。实现了仪表通信系统的控制，即在设计好的用户界面上点击相应的控件完成对仪表的各种操作。最后给出设计好的面板进行通信的测试结果。

关键词：ZW1402 仪表；虚拟仪器；LabWindows/CVI

合作者： 合作甲 14284060xx 14 通信工程 (嵌入式培养)  
合作乙 14284060xx 14 通信工程 (嵌入式培养)  
合作丙 14284060xx 14 通信工程 (嵌入式培养)

## Abstract

Software development tool of virtual instrument, LabWindows/CVI, in this report is used to design a communication system, which can measure and do data processing automatically, by writing or reading serial interface.

This report mainly describes the process of the design of User Interface and the process of implementing all the functions. LabWindows/CVI is used to establish the User Interface and C is used to implement the appropriate functions that handle corresponding operations on the controls. And the control of the communication system has been implemented. In other words, pressing the corresponding controls in the User Interface which is designed to control the instrument. Finally, it will show the test result of communication after running the designed panel.

**Key words:** ZW1402; virtual instrument; LabWindows/CVI

# 目录

<b>第 1 章 绪论</b>	<b>6</b>
1.1 系统设计的背景	6
1.2 系统设计的目的	6
1.3 系统设计的内容	7
<b>第 2 章 开发平台概述</b>	<b>8</b>
2.1 可行性	8
2.2 ZW1402 仪表及其通讯规约	8
2.3 LabWindows/CVI 概述	13
2.3.1 什么是 LabWindows/CVI?	13
2.3.2 为什么用 LabWindows/CVI	14
2.4 LabWindows/CVI 和其他虚拟仪器开发工具的比较	14
<b>第 3 章 系统功能实现</b>	<b>16</b>
3.1 系统模块	16
3.2 图形界面框图设计	16
3.2.1 主面板	16
3.2.2 子面板一	17
3.2.3 子面板二	18
3.2.4 基本控件用途及输入或显示控件属性	19
3.3 用户图形界面	20
3.4 控件具体功能实现	22
3.4.1 变量的定义	22
3.4.2 数据召测主面板部分	22
3.4.3 参数设置子面板部分	24
3.4.4 数据记录子面板部分	24
<b>第 4 章 系统结果分析</b>	<b>28</b>
4.1 运行分析	28

目录	5
4.2 结果分析 .....	30
第 5 章 总结	34
参考文献	36
附录 A 程序源代码	37
A.1 循环校验头文件 .....	37
A.2 数据变换头文件 .....	37
A.3 主程序 .....	40

# 第 1 章 绪论

## 1.1 系统设计的背景

近年来,随着家用电器品种越来越多,由此而产生的电磁干扰在对电子设备的正常使用和可靠性产生影响和危害的同时,也对人们的身体健康造成了直接影响。因此世界各国及有关国际组织建立了电磁兼容实验室,用来检测电子设备的电磁兼容性。青岛青智仪器有限公司生产的 ZW1402 电流表就用于电磁兼容实验室,它具有一定的通信功能,支持 RS232 [1] 或 RS485 通信,可以通过编写相应的软件和它通信,使其实现自动化测量 [2] 和数据处理。电磁兼容性的测量需要采集大量的数据,并且要记录和处理数据,但是目前没有现成的方案,因此本次的设计内容就是对 ZW1402 进行软件编程,实现自动化测量及数据处理。为了实现这些功能,并且有一个友好的用户界面,所有的设计内容都是基于 LabWindows/CVI 虚拟仪器软件开发软件。

在实际设计过程中,将 ZW1402 仪表通过串口 [3-6] 与计算机相连,根据仪表的通信规约,利用 LabWindows/CVI 开发工具上设计函数面板及各个控件,交互地进行 C 语言编程,实现对串口进行读或写操作,经过编译、链接生成可执行的应用程序,使得用户可以通过点击相应的控件实现对仪器的相应操作,从而可以方便地控制仪器的测量、数据记录及处理。

## 1.2 系统设计的目的

要测量电子设备的电磁兼容性,有一种方法是量度电源线上的传导干扰,可以通过测量电源线上的电流得到。为了得到一系列数据用于分析、研究,需要进行长时间自动化无人监守的测量,并且需要对测量结果进行记录和处理。

由于目前没有现成可用的方案,因此需要设计一个自动化测量系统,可以测量并采集大量数据,并且能够加以处理,用来分析测量对象的电磁兼容性。打算使用虚拟仪器开发工具 LabWindows/CVI 进行开发,实现对仪表进行控制。

### 1.3 系统设计的内容

本课题“青智 ZW1402 仪表数据通信程序设计”是设计一个可以自动化测量及数据处理的仪表通信系统。可以运用 LabWindows/CVI 软件开发工具设计控制面板，根据 LabWindows/CVI 提供的函数库，对各个控件编写相应的 C 程序，使其实现相应的功能。使用设计好的控制界面对仪表进行通信，召测数据、记录数据、设置参数和画图表。需要完成的内容包括：

1. 了解 ZW1402 仪表的使用方法及其通讯规约。
2. 软件开发工具 LabWindows/CVI 的掌握及熟练应用。
3. 串口通信原理的了解。
4. 利用 LabWindows/CVI 开发工具进行程序设计，与仪表进行通信，实现仪表控制和数据采集。
5. 实现数据单次记录和循环记录。
6. 实现数据的记录。
7. 对所记录的数据进行图像分析。

## 第 2 章 开发平台概述

### 2.1 可行性

ZW1402 电流表具有一定的通信功能，支持 RS232 或 RS485 通信，可以通过编写相应的软件和它通信，使其实现自动化测量和数据处理。

而 LabWindows/CVI 是面向计算机测控领域的交互式 C 语言软件，可以在多种操作系统 (如 Windows98/NT/2000、MacOS 和 UNIX) 下运行。它以 ANSI C 为核心，将功能强大、使用灵活的 C 语言与数据采集、分析和表达等测控专业工具有机结合。它的集成化开发、交互式编程方法、丰富的功能面板和库函数大大增强了 C 语言的功能，为熟悉 C 语言的开发人员开发检测、数据采集、过程监控等系统提供了一个理想的软件开发环境。

LabWindows/CVI 开发软件对每一个函数都提供了一个函数面板，程序员可以通过函数面板交互地输入函数的每个参数。在脱离主程序 C 源代码的情况下，可直接在函数面板中执行函数操作，并能方便地把函数语句嵌入 C 源代码中，还可通过变量声明窗口交互地声明变量。这种交互式编程技术大大地减少了源代码语句的键入量，减少了程序语句可能出现错误的机会，提高了工程设计的效率和可靠性。

LabWindows/CVI 提供的丰富的函数库，让我们只需了解在设计中用到的函数库的定义及用法，便能够通过它迅速、方便地编写各种不同功能控件的相应程序，通过事件驱动与回调函数方式来实现设计的控制界面的功能调用。

### 2.2 ZW1402 仪表及其通讯规约

ZW1402 电流表的测量对象为 45 Hz~50 Hz 的交流电流信号，信号的转换速率约 8000 次/秒，显示更新约 3 次/秒。仪表的测量量程为 0.015 A~15 A，分辨力为 0.001 A，与信号电源地隔离，实现浮置输入。仪表的显示窗口为 4 位 LED 显示，为保证测量数据的准确度，必须正确接入电流测试信号。当被测电流小于仪表电流量程时，可以直接接入；否则，必须经过电流互感器 CT 接入 [7]。

该仪表不仅提供扩展功能，方便用户扩展其智能特性，提供测试与控制的灵活性，而且提供满足一般工业要求的 MODBUS 规约 RTU 模式，可以自行编写或采用其它符合该规约要求的通讯控制软件构成监控系统。



仪表发送或者接收的消息都是由一个十六进制字符组成的，其通讯规约如下：

1、字节格式如图 2.1 所示。

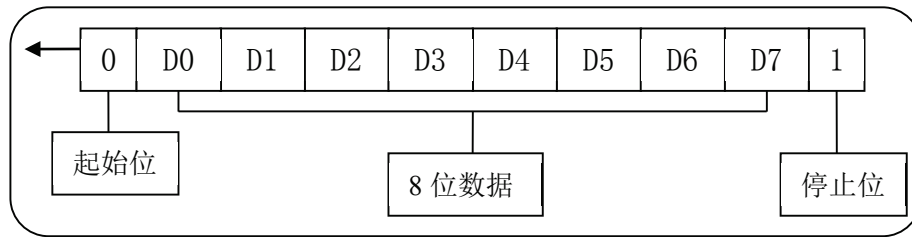


图 2.1: 字节格式

每字节含 8 位二进制码，传输时加上一个起始位(0)，一个停止位(1)，共 10 位。D0 是字节的最低有效位，D7 是字的最高有效位。先传低位，后传高位。

2、通讯数据格式：通讯时数据以字 (WORD—2 字节) 的形式回送，回送的每个字中，高字节在前，低字节在后，如果 2 个字连续回送 (如：浮点或长整形)，则高字在前，低字在后。格式如表 2.1 所示。

表 2.1: 通讯数据格式

数据类型	寄存器数	字节数	说明
字节数据	1	1	
整形数据	1	2	一次送回，高字节在前， 低字节在后
长整形数 浮点数据	2	4	分两个字送回，高字在前， 低字在后

3、帧格式：包括读取和设置仪表寄存器内容

(1) 读取仪表寄存器内容 (功能码 03H)

上位机发送的帧格式如表 2.2 所示。数据正常时，仪表回送的帧格式如表 2.3 所示。如果起始寄存器地址或寄存器个数错误，仪表回送如表 2.4 所示。

表 2.2: 上位机发送的帧格式

顺序	代码	示例	说明
1	仪表地址	1	仪表的通讯地址 (1-255)
2	03H	03H	功能码
3	起始寄存器地址高字节	10H	寄存器起始地址
4	起始寄存器地址低字节	00H	

顺序	代码	示例	说明
5	寄存器个数高字节	00H	寄存器个数
6	寄存器个数低字节	02H	
7	CRC16 校验高字节	C0H	CRC 校验数据
8	CRC16 校验低字节	CBH	

表 2.3: 回送的帧格式

顺序	代码	说明
1	仪表地址	仪表的通讯地址 (1-255)
2	03H	功能码
3	回送数据域字节数 (M)	
4	第一个寄存器数据	
.....	.....	
	第 N 个寄存器数据	
M+4	CRC 校验高字节	
M+5	CRC 校验低字节	

表 2.4: 仪表回送

顺序	代码	示例	说明
1	仪表地址	1	仪表的通讯地址 (1-255)
2	83H	83H	功能码
3	02H	02H	错误代码
4	CRC 检验高字节	C0H	
5	CRC 校验低字节	F1H	

## (2) 设置仪表寄存器内容 (功能码 16H 或 10H 或 06H)

功能码 06H 写单路，将一个字 (2 字节) 数据写入仪表寄存器中，上位机发送的帧格式如表 2.5 所示，如果写入正确，则仪表回送相同的数据。

功能码 16H 或 10H 写多路寄存器，上位机发送的帧格式如表 2.6 所示。如果写入成功的话，仪表回送帧格式如表 2.7 所示。

如果通讯地址或写入的数据有错，则仪表回送如表 2.8 所示。

表 2.5: 单路设置帧格式

顺序	代码	示例	说明
1	仪表地址	1	仪表的通讯地址 (1-255)
2	06H	06H	功能码
3	寄存器地址高字节	10H	寄存器地址 1000H
4	寄存器地址低字节	00H	
5	写入数据高字节	00H	写入数据 0CH
6	写入数据低字节	0CH	
7	CRC 校验高字节	8DH	CRC 校验数据 8D0FH
8	CRC 校验低字节	0FH	

表 2.6: 多路设置帧格式

顺序	代码	示例	说明
1	仪表地址	1	仪表的通讯地址 (1-255)
2	16H 或 10H	10H	功能码
3	寄存器起始地址高字节	1FH	寄存器地址 1F02H
4	寄存器起始地址低字节	02H	
5	寄存器个数高字节	00H	00H
6	寄存器个数低字节	02H	字节数据、整形数据: 01H 浮点数据、长整形数: 02H
7	字节数 (M)	4	字节数据: 01H 整形数据: 02H 浮点数、长整形数: 04H
8	数据高字节	42H	设置的浮点数为 100
	数据次高字节	C8H	
	数据次低字节	00H	
	数据低字节	00H	
M+8	CRC 校验高字节	6BH	CRC 校验数据 6BC0H
M+9	CRC 校验低字节	C0H	

表 2.7: 仪表回送帧格式

顺序	代码	示例	说明
1	仪表地址	1	仪表的通讯地址 (1-255)
2	16H 或 10H	10H	功能码
3	起始地址高字节	1FH	寄存器起始地址 1F02H
4	起始地址低字节	02H	
5	寄存器个数高字节	00H	寄存器个数 2
6	寄存器个数低字节	02H	
7	CRC 校验高字节	E7H	CRC 校验数据 E7DCH
8	CRC 校验低字节	DCH	

表 2.8: 仪表回送帧格式

顺序	代码	说明
1	仪表地址	仪表的通讯地址 (1-255)
2	96H 或 90H 或 86H	功能码 —针对 16H,10H,06H
3	03H	错误代码
4	CRC 校验高字节	
5	CRC 校验低字节	

#### 4、通讯数据循环冗余校验

(1) 循环冗余校验码 (CRC 码): 是数据通信领域中最常用的一种差错校验码, 其特征是信息字段和校验字段的长度可以任意选定。

(2) 生成 CRC 码的基本原理: 任意一个由二进制位串组成的代码都可以和一个系数仅为 '0' 和 '1' 取值的多项式一一对应。例如代码 1010111 对应的多项式为  $x^6+x^4+x^2+x+1$ , 而多项式为  $x^5+x^3+x^2+x+1$  对应的代码 101111。仪表的校验多项式为  $x^{16}+x^{12}+x^5+1$ 。

(3) CRC 码集选择的原则: 若设码字长度为  $N$ , 信息字段为  $K$  位, 校验字段为  $R$  位 ( $N = K + R$ ), 则对于 CRC 码集中的任一码字, 存在且仅存在一个  $R$  次多项式  $g(x)$ , 使得

$$V(x) = A(x)g(x) = x^R m(x) + r(x)$$

其中:  $m(x)$  为  $K$  次信息多项式,  $r(x)$  为  $R-1$  次校验多项式,  $g(x)$  称为生成多项式:

$$g(x) = g_0 + g_1 + g_2x^2 + \dots + g_{(R-1)}x^{(R-1)} + g_Rx^R$$

发送方通过指定的  $g(x)$  产生 CRC 码字, 接收方则通过该  $g(x)$  来验证收到的 CRC 码字。

(4) CRC 校验码软件生成方法: 借助于多项式除法, 其余数为校验字段。

5、通讯波特率: 通讯波特率可以在 300、600、1200、2400、4800、9600 之间选择。出厂时, 仪表已经设置某一波特率。

6、仪表地址: 仪表地址可以在 1-247 之间选择。仪表出厂时, 已经设置某一地址。

7、通讯功能码: 03H (召测数据) 16H (10H 或 06H) (数据设置)

## 2.3 LabWindows/CVI 概述

### 2.3.1 什么是 LabWindows/CVI?

LabWindows/CVI 是美国 NI 公司推出的面向仪器与测控过程的 C/C++ [8, 9] 交互式开发平台, 可以在多种操作系统 (如 Windows98/NT/2000、MacOS 和 UNIX) 下运行。它以 ANSI C 为核心, 将功能强大、使用灵活的 C 语言与数据采集、分析和表达等测控专业工具有机结合。它的集成化开发、交互式编程方法、丰富的功能面板和库函数大大增强了 C 语言的功能, 为熟悉 C 语言的开发人员开发检测、数据采集、过程监控等系统提供了一个理想的软件开发环境 [10-13]。

LabWindows/CVI 编程环境有四个主要的界面窗口:

- (1) 工程窗口 (Project Windows)。工程窗口中列出了组成该工程的所有文件。
- (2) 用户界面编辑窗口 (User Interface Editor Windows)。用户界面编辑窗口是用来创建、编辑用户界面的, 它所形成的文件为 \*.uir 文件。用户界面相当于真实仪器的操作面板, 一个用户界面文件至少要有—个面板, 面板上放置完全不同功能的控件。图形化用户界面编辑窗口提供了非常快捷的创建、编辑面板和控件属性的设置等功能, 可在短时间里创建出符合要求的图形界面。
- (3) 源代码编辑窗口 (Source Windows)。在源代码编辑窗口可创建或编辑 C 语言代码文件, 如编程所需的添加、删除、插入函数等基本编辑操作。
- (4) 函数面板 (Function Panel)。函数面板是用户界面的基础, 是对传统仪器中控制面板的虚拟。在面板上, 用户可以任意添加各种控件来建立用户界面。在源程序编辑窗口中, 如果要在程序某处插入函数, 只需从函数所在的库中选中该函数, 在弹出的函数面板中填入该函数所需的参数后即可完成函数的插入。

使用 LabWindows/CVI 可以完成以下工作:

- (1) 交互式程序开发。
- (2) 具有功能强大的库函数, 用来创建数据采集和仪器控制的应用程序。
- (3) 充分利用完备的软件工具进行数据采集、分析和显示。
- (4) 利用向导开发 IVI 仪器驱动程序和创建 ActiveX 服务器。
- (5) 为其他程序开发 C 目标模块、动态链接库 (DLL)、C 语言库。

### 2.3.2 为什么用 LabWindows/CVI

LabWindows/CVI 平台简单易用、功能强大,可适用于有一定 C 语言基础的科技人员,它不仅提供了对虚拟仪器的支持能力,还具有各种测试、控制和数值分析的能力,具有图形建模简单、控制功能强大、实时性强、编程容易等优点。从软件开发的角度来看,LabWindows/CVI 具有以下一些特点:

- (1) 基于标准 C 语言,简单易学。
- (2) 可视化、交互式的开发工具。具有人机交互界面编辑器,运用可视化交互技术实现“所见即所得”,使人机界面的实现直观简单。对每一个函数都提供了一个函数面板,用户可以通过函数面板交互地输入函数的每一个参数及属性值。这种交互式编程技术大大提高了工程设计的效率和可靠性。
- (3) 具有程序自动生成的能力,可减少软件开发过程中代码编写的工作量。设计好的人机交互界面(虚拟仪器面板)存储在后缀名为 .uir 的文件中。LabWindows/CVI 自动生成原码头文件 .h,自动声明界面对象常量及相关的回调函数,编程人员不必钻研这些技术。
- (4) 具有齐全的软件工具包及功能强大的函数库,通过简单调用库函数就能驱动相应的总线的各种仪器和硬件板卡。这些工具包和函数库具有更高的效率,他使得程序的编写更简洁、直观。
- (5) 完善的兼容性。借助于 LabWindows/CVI,有经验的 C/C++ 语言开发人员可以采用熟悉的 C 语言环境,如 VC, BC 等开发自己的虚拟仪器系统。另外,还可将仪器库函数及子程序编译成 32 位 DLL,以用于任中。
- (6) 多种灵活的函数调用手段。所提供的变量显示窗口可观察程序变量和表达式的变化情况,还提供了单步执行、断点执行、过程跟踪、参数检查、运行时内存检查等多种调试手段,可更容易地对程序进行调试、排错。
- (7) 强大的 Internet 功能,支持常用网络协议,方便网络仪器、远程测控仪器的开发。

## 2.4 LabWindows/CVI 和其他虚拟仪器开发工具的比较

虚拟仪器的开发工具比较广泛,目前比较流行的有面向对象的编程技术和图形编程技术。面向对象的可视化编程语言环境有 Visual C++、Visual Basic 等通用工具。但相比较图形编程语言来说,其编程难度较大,开发周期较长且不易进行更改、升级和维护等。而图形编程语言在这方面具有无可比拟的优势,它具有简单易学,开发周期短,开发出的应用程序界面美观,功能强大。其中专门用于测量和控制方面的虚拟仪器开发工具的代表产品有 HP 公司的 HP VEE、NI 公司的 LabVIEW、LabWindows/CVI 等。

HP VEE 是一种工程可视化编程语言,它对整个语言做了彻底的图形化处理,并提供了模块式的编程工具,同时还提供了数据流显示和程序流显示,非常适合于构成测试和测量应用程序。在仪器控制方面,HP VEE 提供了两种使用方式(软面板方式和直

接输入输出方式)。LabVIEW 的图形化程序设计是基于现代软件的面向对象技术和数据流技术而发展起来的。用户能够通过连接功能模块来快速开发自己的应用程序,甚至能够使用多路数据通道,实现同步操作。LabWindows/CVI 是用 C 语言构建仪器系统的交互式软件开发环境。其可以模块化方式对 C 语言进行编辑、编译、连接和调试。LabWindows/CVI 软件把 C 语言的有力与柔性同虚拟仪器的软件工具库结合起来,包含了 GPIB、RS232、VXI 总线、数据采集和分析库,且用户自己开发的程序可在不同的平台上移植。

由于设计人员有 C 语言编程功底,且希望本设计可以更加深入、有效率,所以在众多软件工具中选择 LabWindows/CVI 作为本设计的开发工具。

## 第3章 系统功能实现

### 3.1 系统模块

整个测量控制系统的模块如图 3.1 所示。

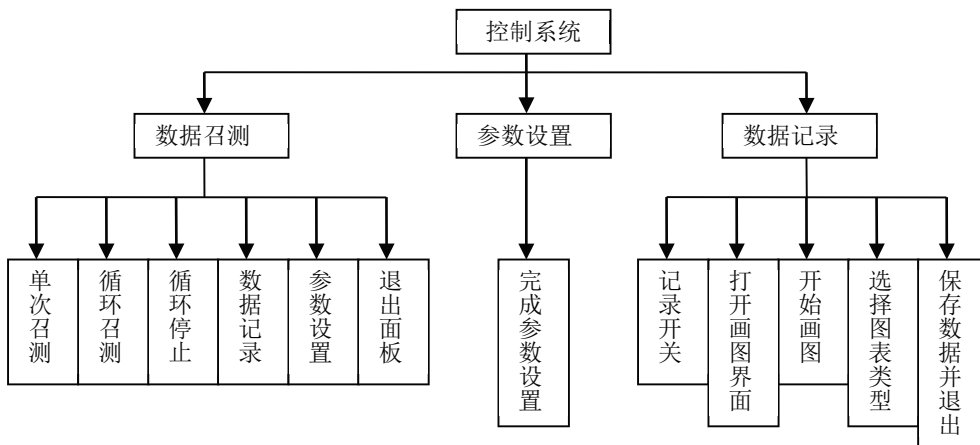


图 3.1: 系统模块

### 3.2 图形界面框图设计

传统仪器面板上的器件都是实物，而且是用手动和触摸进行操作的，而虚拟仪器的面板控件是外形与实物相像的图标，“通”、“断”、“放大”等对应着相应的软件程序。这些软件已经设计好了，用户不必设计，只需选用代表该软件程序的图形控件即可，有计算机的鼠标对其进行操作。因此，设计虚拟面板的过程就是在面板设计窗口中摆放所需的控件，然后编写相应的程序。面板和控件是虚拟仪器的重要组成部分。一个虚拟仪器可包含多个仪器面板，每个面板中可以包含不同的控件，而面板本身也是一个控件。

#### 3.2.1 主面板

主面板-数据召测面板设计的系统框图如图 3.2 所示。



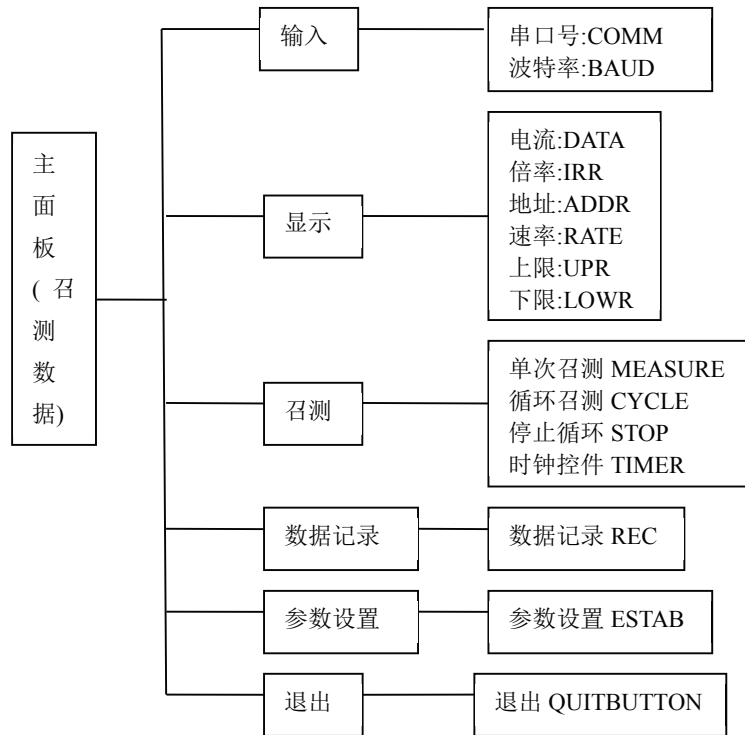


图 3.2: 主面板系统框图

首先建立一个名为 `comm.uir` 的用户界面，在此基础上建一个名为“数据召测”的主面板。在面板上，两个循环控件作为串口号和波特率的输入，让用户在设定的一组数值中选择，以防用户输入不规范的数值；六个数字控件用来显示从仪表读到的数据，分别是电流、电流倍率 (CT)、仪表地址、电流值的上下限，除了地址和速率这两个控件的数据类型设定为整形外，其余的都设定为包含 3 个小数点的浮点型；六个命令按钮控件，分别用来控制数据召测、循环召测、停止循环召测、数据记录、参数设置和退出，其中数据记录和参数设置这两个控件都会打开相应的子面板；一个时钟控件用来控制数据的循环召测，时间间隔设定为 0.333 秒，与仪表刷新率一致；两个信息框控件，一个用来显示面板的功能，一个用来根据通讯的进行显示通讯状态。

### 3.2.2 子面板一

子面板—参数设置面板设计的系统框图如图 3.3 所示。

在 `comm.uir` 的用户界面上，又创建一个名为“参数设置”的子面板。在面板上，两个循环控件作为串口号和波特率的输入，让用户在设定的一组数值中选择，以防用户输入不规范的数值；三个数字控件用来显示要对仪表设置的参数的数值，分别是电流倍率 (CT)、电流值的上下限，都设定为包含 3 个小数点的浮点型；一个命令按钮控件，用来控制设置界面的退出；一个信息框控件，用来显示面板的功能。

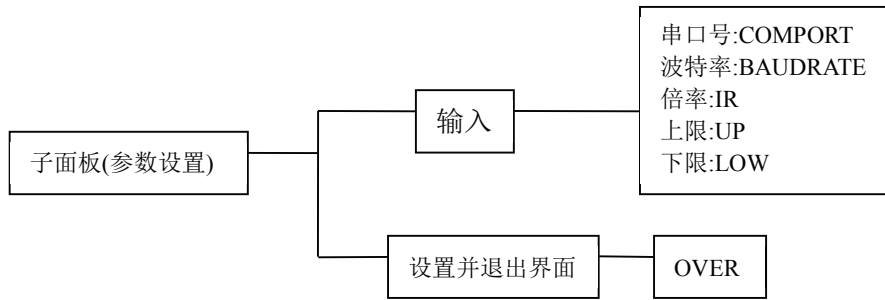


图 3.3: 子面板 (参数设置) 系统框图

### 3.2.3 子面板二

子面板 -数据记录面板设计的系统框图如图 3.4 所示。

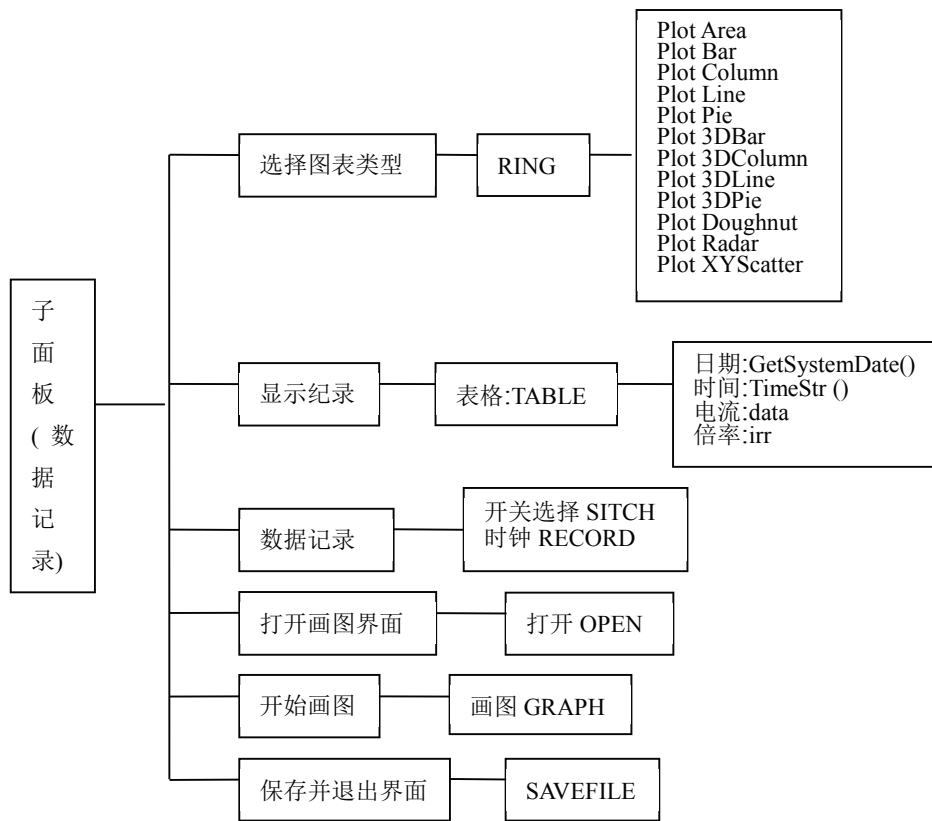


图 3.4: 子面板 (数据记录) 系统框图

在名为 comm.uir 的用户界面上，再创建一个名为“数据记录”的子面板。在面板上，一个表格控件用来逐一地显示数据，该表格一共有 4 列，标题分别为日期、时间、电流和倍率；一个触发按钮控件作为控制开始或停止数据记录的时钟控件的触发；一个循环控件，用来让用户选择图表的类型，共设定有 12 种图表类型；一个时钟控件用来控制数据记录的开始或停止，时间间隔设定为 1 秒，方便记录；三个命令按钮控件，分别

用来控制画图界面的打开、开始画图和退出；一个信息框控件，用来显示面板的功能。

### 3.2.4 基本控件用途及输入或显示控件属性

基本的控件用途如表 3.1 所示，主要控件的属性设置如表 3.2 所示。

表 3.1: 基本的控件用途

控件名	回调函数	用途	控件名	回调函数	用途
MEASURE	Measure	单次召测	OVER	Over	对仪表进行参数设置
CYCLE	Cycle	循环召测	RING	ChartSelect	选择图表的类型
STOP	Stop	停止循环	SWITCH	Switch	触发时钟控件运行
TIMER	Time	控制循环	RECORD	Record	记录数据
REC	Rec	打开数据记录面板	OPEN	Open	打开画图界面
ESTAB	Estab	打开参数设置面板	GRAPH	Graph	画图
QUITBUTTON	QuitCallback	退出主面板	SAVEFILE	SaveFile	保存数据并退出面板

表 3.2: 输入或显示控件的属性设置

ConstName	Lable	控件类型	用途
COMM	端口	Ring	在主面板，供用户选择设定好的串口号
COMPORT			在设置参数时供用户选择串口号
BAUD	波特率	Ring	在主面板，供用户选择设定好的波特率
BAUDRATE			在设置参数时供用户选择波特率
DATA	电流	Numeric	显示从仪器读到的电流数值
ADDR	地址	Numeric	显示从仪器读到的通讯地址
IRR	CT	Numeric	显示从仪器读到的电流倍率
IR			在设置参数时供用户设置电流倍率

ConstName	Lable	控件类型	用途
UPR	上限	Numeric	显示从仪器读到的电流上限
UP			在设置参数时供用户设置电流上限
LOWR	下限	Numeric	显示从仪器读到的电流下限
LOW			在设置参数时供用户设置电流下限
TABLE	-	Table	显示逐一记录的数据

### 3.3 用户图形界面

主界面-数据召测面板如图 3.5 所示，主要显示可供用户使用的界面，通过主面板，用户可以使用该虚拟仪器。子面板-参数设置面板如图 3.6 所示，主要用来显示对仪表参数的设置内容，这个子面板是由主面板的“设置”控件触发打开的。子面板-数据记录面板如图 3.7 所示，主要用来显示并记录从仪表循环读取的电流读数，这个子面板是由主面板的“数据记录”控件触发打开的。

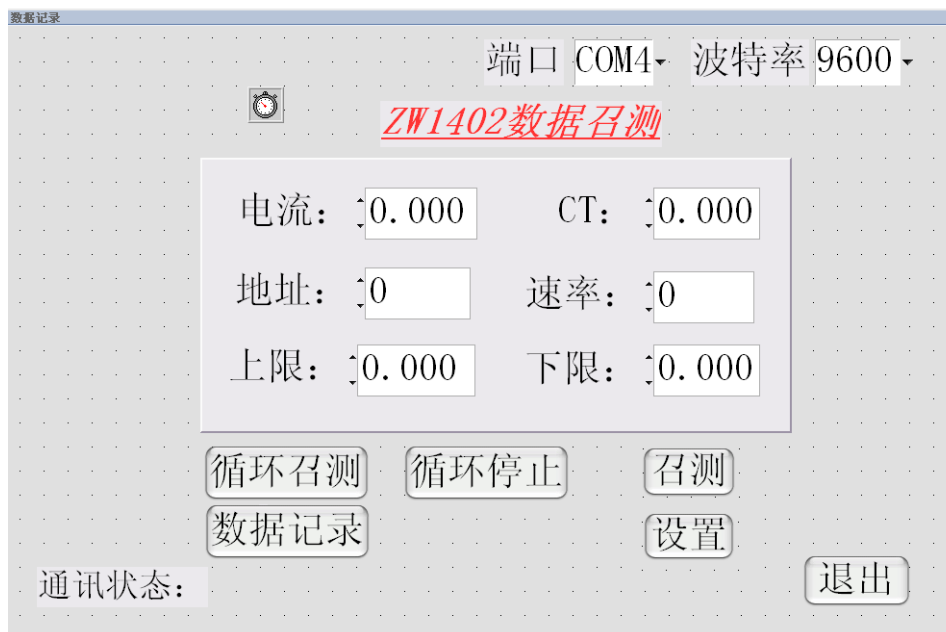


图 3.5: 主界面-数据召测

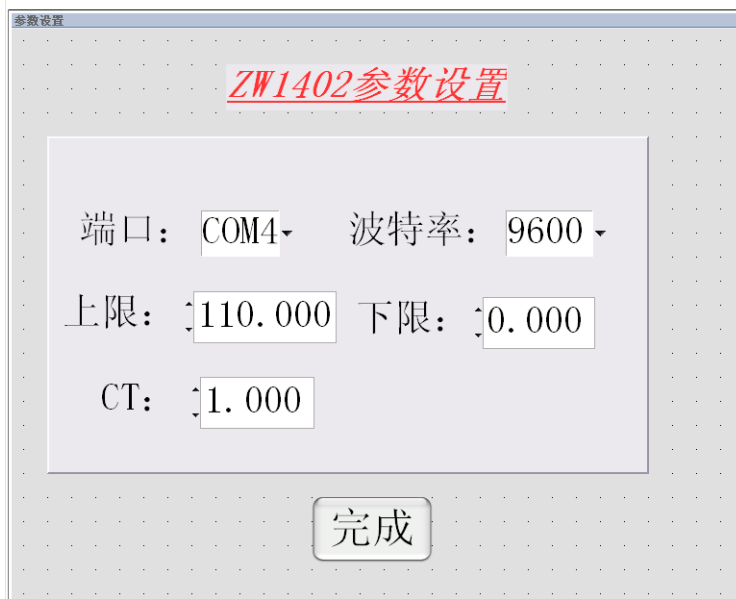


图 3.6: 子面板 - 参数设置



图 3.7: 子面板 - 数据记录

## 3.4 控件具体功能实现

### 3.4.1 变量的定义

在数据召测主面板中定义的变量如表 3.3 所示，在参数设置子面板定义的变量如表 3.4 所示。

在主面板中除了 comm 和 baud 是用户输入的，其他都是通过串口从仪表读进来的，而参数设置面板中的变量都是供用户输入来设置仪表参数的。函数 SetCtrlVal() 是用来显示面板上数字控件的数值，可以将从串口读取的数据显示在数字控件上；而函数 GetCtrlVal() 则是用来获得面板上数字控件的数值，可以读取该控件上的数值，并存到指定的变量中，以供程序使用。

表 3.3: 数据召测主面板的变量定义

变量	数据类型	初值	变量	数据类型	初值
data	float	0.000	upr	float	0.000
irr	float	0.000	lowr	float	0.000
addr	int	0	comm	int	4
rate	int	0	baud	int	9600

表 3.4: 参数设置子面板的变量定义

变量	数据类型	初值	变量	数据类型	初值
comport	int	4	up	float	110.000
baudrate	int	9600	low	float	0.000
ir	float	1.000			

### 3.4.2 数据召测主面板部分

#### 3.4.2.1 单次召测

单次召测的具体实现过程如图 3.8 所示。

首先用函数 OpenComConfig() 打开串口，对串口的端口号、波特率、奇偶校验位、数据位、输入及输出队列长度等进行相应的设置。然后根据仪表上位机的帧格式，读取仪表的电流，即将输出队列通过函数 ComWrt() 写到串口，给仪表发送上位机帧格式，再利用函数 ComRd() 从串口读取数据，判断读取的帧格式是否正确，如果正确就在相应的控件显示，否则在“通讯状态”处显示“通讯超时”提示用户出现通讯错误。接

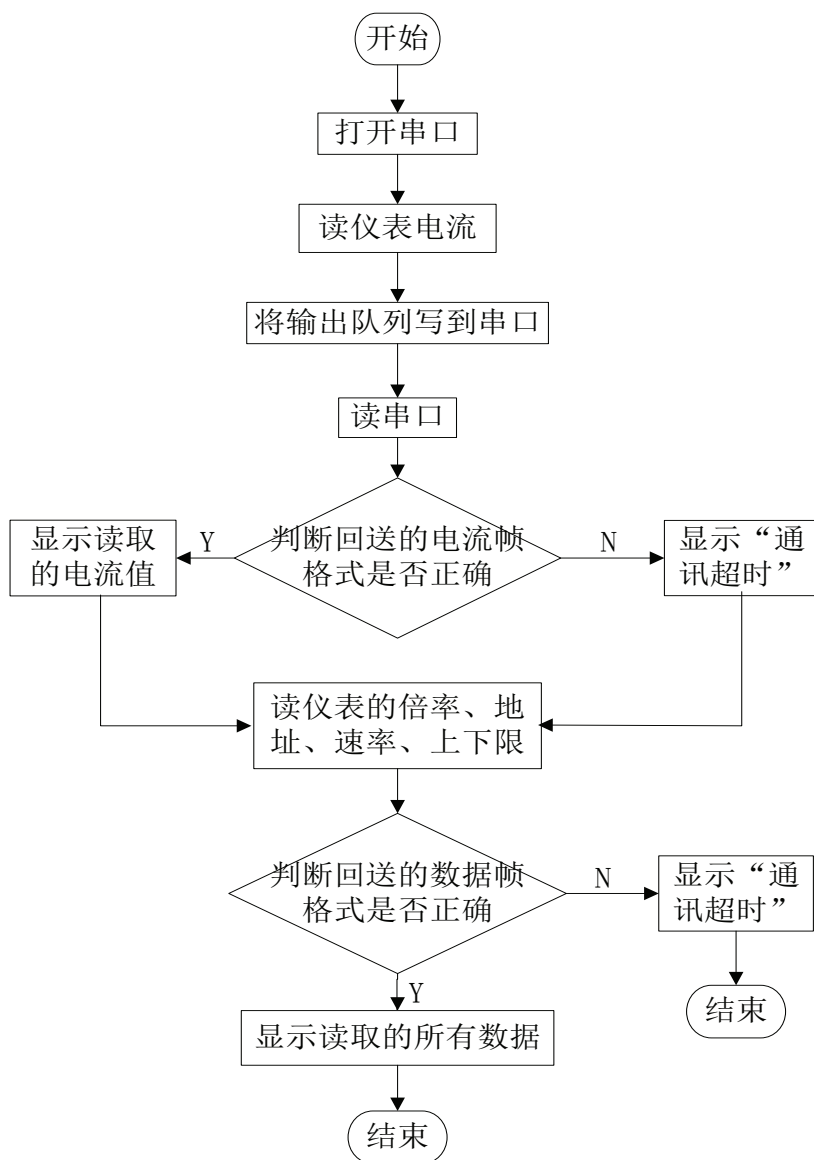


图 3.8: 单次召测流程图

着继续读仪表的其他参数值，如倍率、地址、速率及电流上下限。判断每一个数据的帧格式是否正确，如果正确就依次显示，否则按照相应的参数显示其通讯超时信息。

### 3.4.2.2 循环召测

循环召测的具体实现过程如图 3.9 所示。

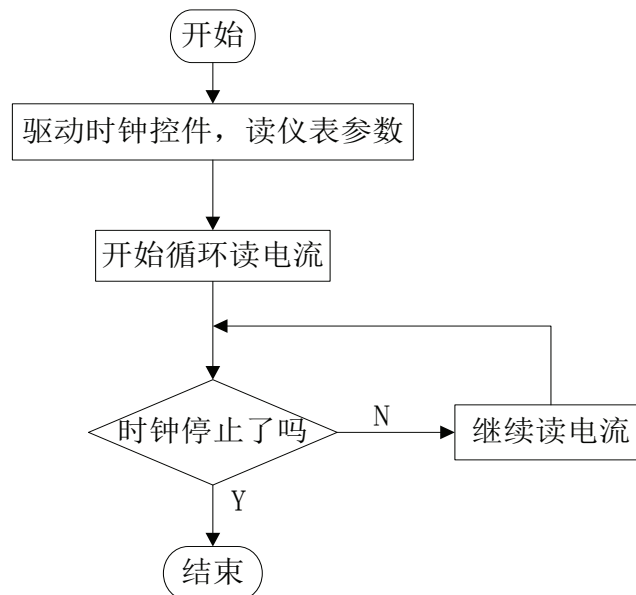


图 3.9: 循环召测流程图

首先改变时钟的属性 `ATTR_ENABLED`，驱动时间控件，使得在每一个时钟周期到来时读电流，并显示在相应的数字控件上，同时通过函数 `RdData()` 将仪表的其他参数都进来并显示在相应的控件上。判断时钟控件是否还在运行，如果是就继续循环召测电流，否则就停止召测。

### 3.4.3 参数设置子面板部分

参数设置的具体实现过程如图 3.10 所示。

首先点主面板的“设置”按钮，弹出参数设置面板，用户可对面板上的参数进行设置，然后单击“完成”按钮，即可退出子面板，并在主面板显示通讯正确与否，如果正确就显示“成功”，否则就显示“通讯错误”。

### 3.4.4 数据记录子面板部分

#### 3.4.4.1 数据记录

数据记录的具体实现过程如图 3.11 所示。



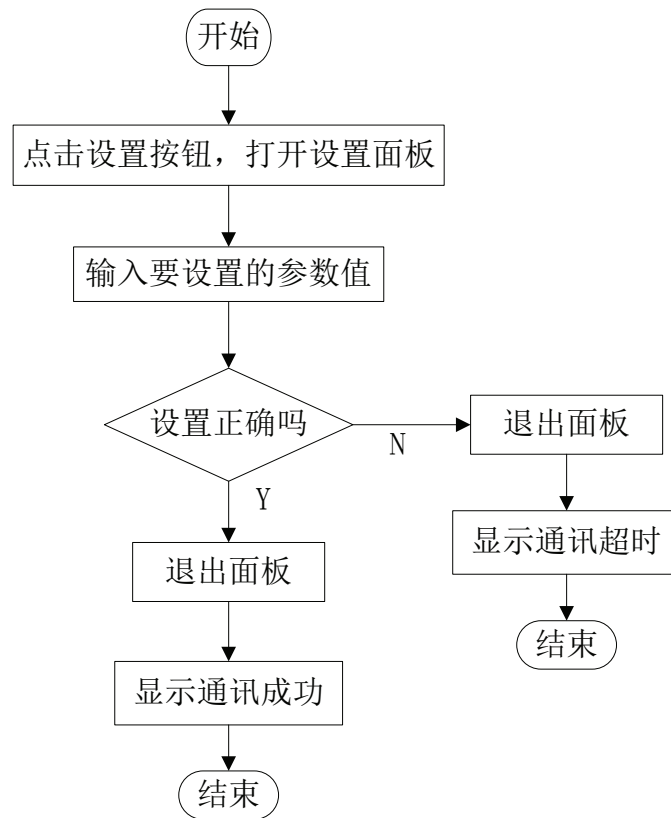


图 3.10: 参数设置流程图

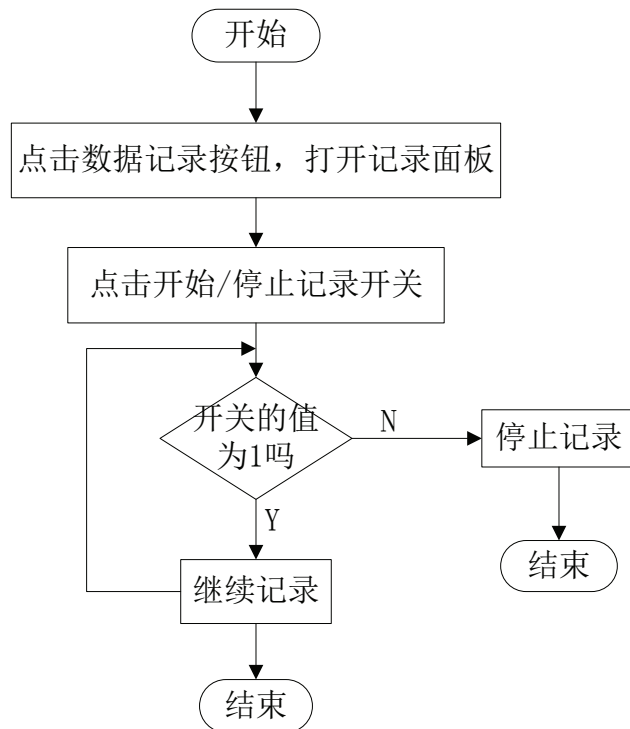


图 3.11: 数据记录流程图

首先点主面板的“数据记录”按钮，弹出数据记录面板。再打开的子面板中点击“开始/停止记录”的开关控件，触发时钟控件开始运行，在表格里逐一记录数据。若要停止记录，可以在此点击开关控件，即停止记录数据。

#### 3.4.4.2 画图

画图的具体实现过程如图 3.12 所示。

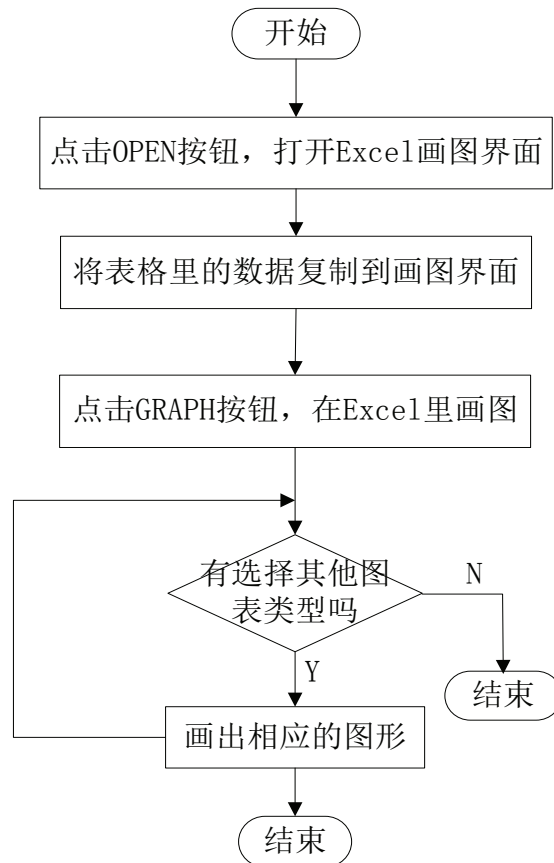


图 3.12: 画图流程图

点击 OPEN 按钮，打开 Excel 画图界面，将表格上的数据复制到 Excel 上。再点击 GRAPH 按钮，在 Excel 画图。如果要选择其他形状的图形，可以在面板上的选择控件选择，可以在 Excel 里画出相应的图形。

#### 3.4.4.3 保存数据

保存数据的具体实现过程如图 3.13 所示。

点击 SAVE 按钮，弹出保存对话框，在对话框内选择保存路径，输入文件名，然后保存，就开始将表格上的数据一条一条保存，直至保存到最后一行。

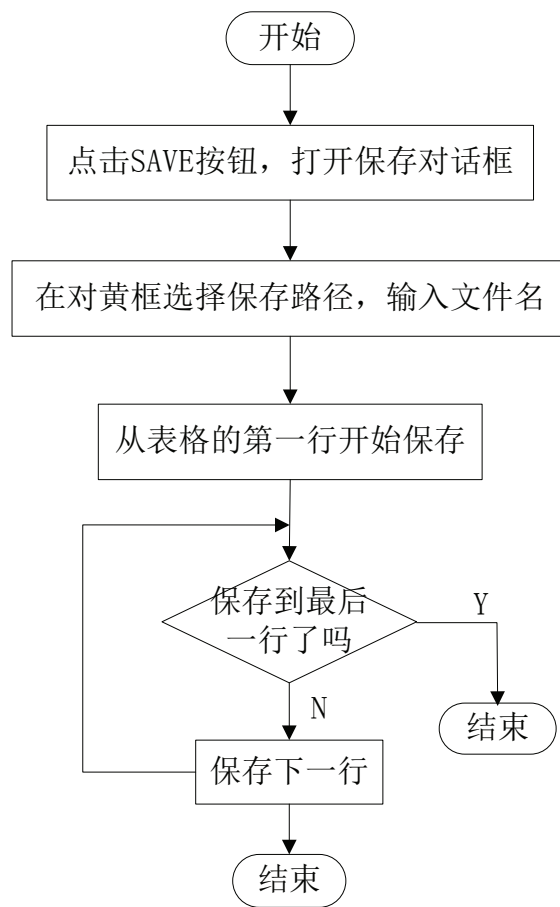


图 3.13: 保存数据流程图

## 第 4 章 系统结果分析

### 4.1 运行分析

将仪表接上笔记本电脑的电源，程序运行后，在主面板上选择串口 COM4，波特率 9600，点击“召测”，可以看到面板上显示从仪表读到的电流和其他参数值，并且在“通讯状态”处显示“成功”，其图形界面如图 4.1 所示。



图 4.1: 单次召测界面

点击“设置”，则弹出一个参数设置面板，如图 4.2 所示。在该面板设置仪表的各种参数，再点击“完成”，子面板消失，则返回主面板，并在“通讯状态”处显示“设置成功”，如图 4.3 所示。

点击“循环召测”，电流一栏的数据会自动地不停变化，原先隐藏的“停止循环”和“数据记录”的按钮出现。点击“数据记录”，则弹出数据记录面板，点击选中“开始/停止



图 4.2: 参数设置界面



图 4.3: 设置成功

记录”，则开始记录数据。再次点击开关控件，则停止记录，“OPEN”按钮由初始的模糊变清晰，成为可用按钮，此时界面如图 4.4 所示。点击“OPEN”按钮则弹出 Excel 界面，并且将表格上的数据复制到 Excel 中，如图 4.5 所示。在数据记录面板张点击“GRAPH”，则在 Excel 界面画出相应的图形，如图 4.6 所示。在供选择图表类型的控件选择不同的选项，则可相应画出不同类型的图形，例如 XY 轴分散图，如图 4.7 所示。



图 4.4: 数据记录界面

将画图界面关掉，点击“SAVE”按钮，则弹出保存对话框，选择保存路径，输入文件名，即可保存。保存对话框如图 4.8 所示，保存结果如图 4.9 所示。保存后，数据记录面板被关闭，重新返回主面板。点击“停止循环”，则电流数值停在当前数值，不再变化。点击“退出”即可关闭主面板。

## 4.2 结果分析

本次设计的目的是实现自动化测量和数据处理，由程序运行过程来看，该系统可以实现单次测量即手动测量，也可以实现循环测量即自动化测量，还可以对仪表的参数进行设置。在循环测量的基础上，可以实现数据记录，并且可以逐条更新显示。对表格记录的数据可以进行不同类型图形的画图分析，最后还可以保存数据。综上，本次设计基本完成预计任务。但由于时间和精力有限，也存在一些不足，例如数据记录

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	20090528	10:38:01	0.281449														
2	20090528	10:37:09	0.293891														
3	20090528	10:37:11	0.304782														
4	20090528	10:37:11	0.290999														
5	20090528	10:37:12	0.286451														
6	20090528	10:37:13	0.303548														
7	20090528	10:37:14	0.288571														
8	20090528	10:37:15	0.304134														
9	20090528	10:37:16	0.292234														
10	20090528	10:37:18	0.302993														
11	20090528	10:37:18	0.299075														
12	20090528	10:37:19	0.28829														
13	20090528	10:37:20	0.299642														
14	20090528	10:37:21	0.300505														
15	20090528	10:37:22	0.296858														
16	20090528	10:37:38	0.293053														
17	20090528	10:37:39	0.290804														
18	20090528	10:37:40	0.290483														
19	20090528	10:37:42	0.289116														
20	20090528	10:37:43	0.2887														
21	20090528	10:37:43	0.301998														
22	20090528	10:37:44	0.29185														
23	20090528	10:37:45	0.289437														
24	20090528	10:37:51	0.284889														
25	20090528	10:37:52	0.29345														
26	20090528	10:37:54	0.289985														
27	20090528	10:37:54	0.288624														
28	20090528	10:37:56	0.288429														
29	20090528	10:38:02	0.287686														
30	20090528	10:38:03	0.287062														
31																	
32																	

图 4.5: 画图界面

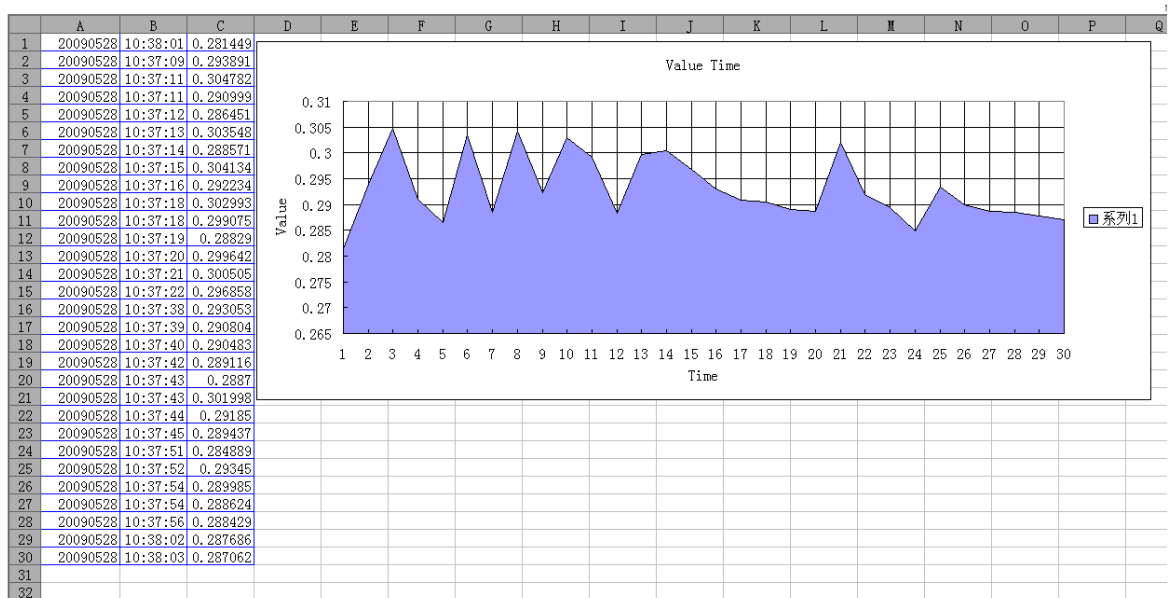


图 4.6: 画图

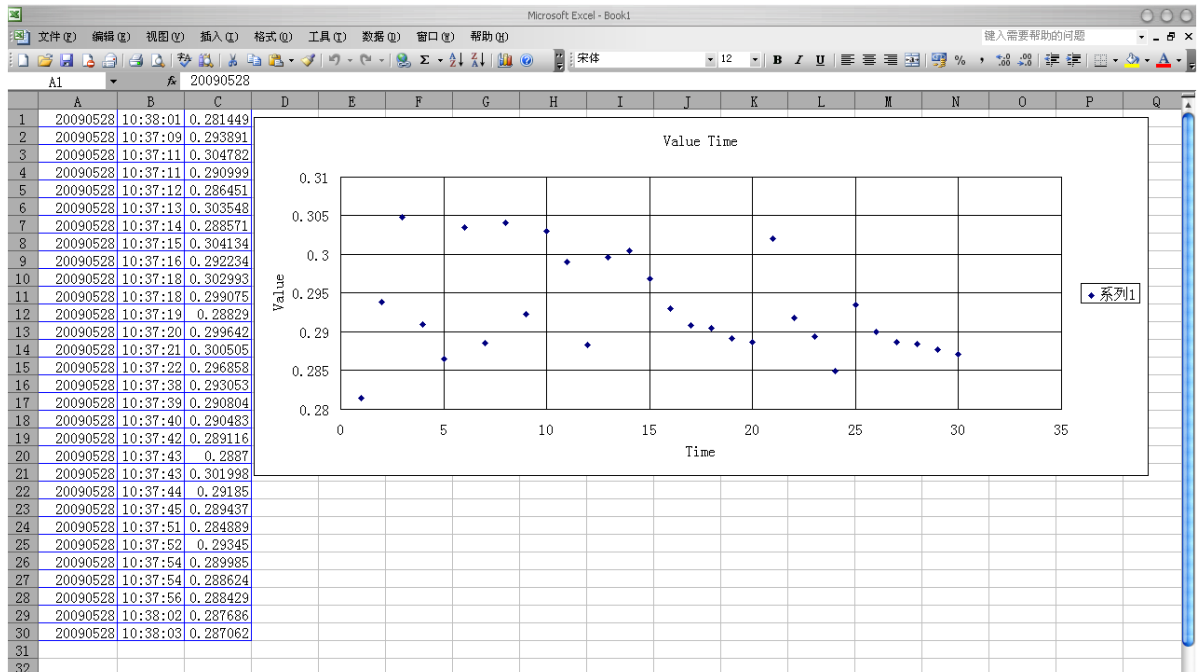


图 4.7: XY 轴分散图

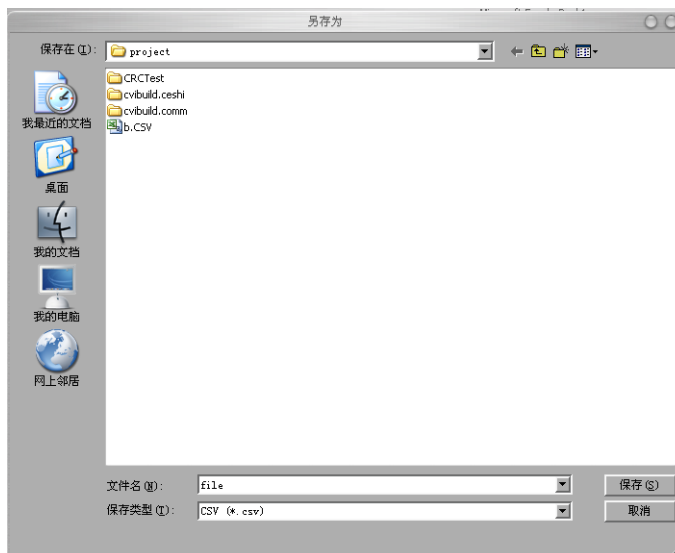
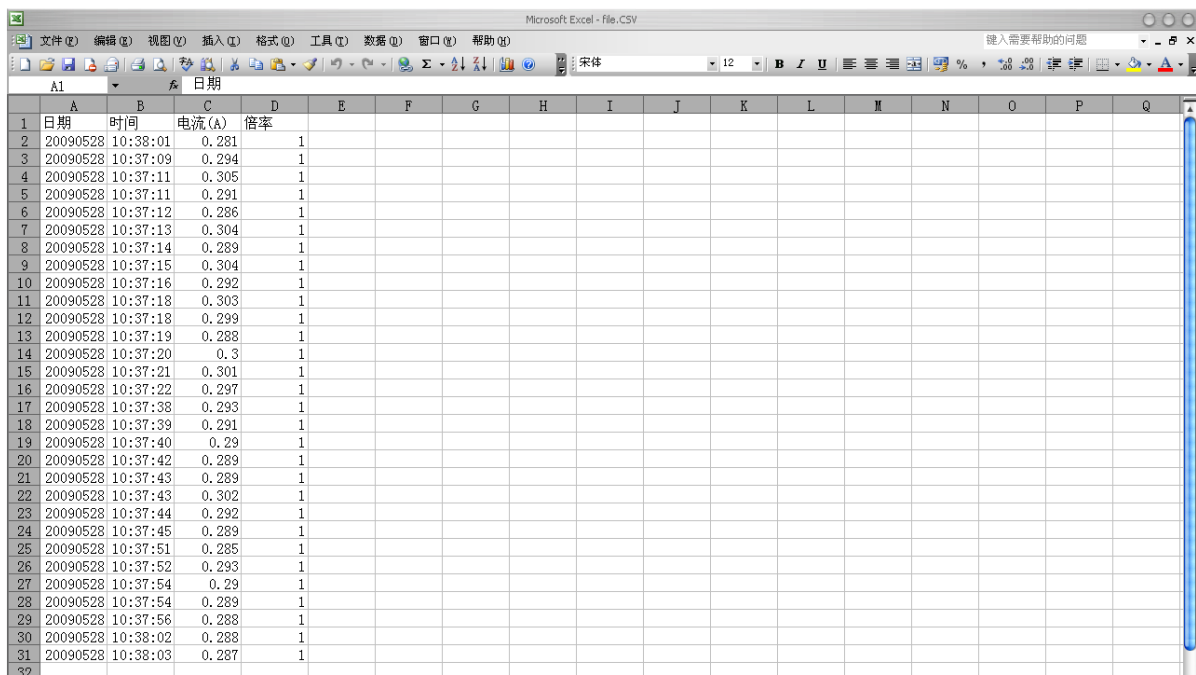


图 4.8: 保存对话框





The screenshot shows a Microsoft Excel spreadsheet with the following data:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	日期	时间	电流 (A)	倍率													
2	20090528	10:38:01	0.281	1													
3	20090528	10:37:09	0.294	1													
4	20090528	10:37:11	0.305	1													
5	20090528	10:37:11	0.291	1													
6	20090528	10:37:12	0.286	1													
7	20090528	10:37:13	0.304	1													
8	20090528	10:37:14	0.289	1													
9	20090528	10:37:15	0.304	1													
10	20090528	10:37:16	0.292	1													
11	20090528	10:37:18	0.303	1													
12	20090528	10:37:18	0.299	1													
13	20090528	10:37:19	0.288	1													
14	20090528	10:37:20	0.3	1													
15	20090528	10:37:21	0.301	1													
16	20090528	10:37:22	0.297	1													
17	20090528	10:37:38	0.293	1													
18	20090528	10:37:39	0.291	1													
19	20090528	10:37:40	0.29	1													
20	20090528	10:37:42	0.289	1													
21	20090528	10:37:43	0.289	1													
22	20090528	10:37:43	0.302	1													
23	20090528	10:37:44	0.292	1													
24	20090528	10:37:45	0.289	1													
25	20090528	10:37:51	0.285	1													
26	20090528	10:37:52	0.293	1													
27	20090528	10:37:54	0.29	1													
28	20090528	10:37:54	0.289	1													
29	20090528	10:37:56	0.288	1													
30	20090528	10:38:02	0.288	1													
31	20090528	10:38:03	0.287	1													
32																	

图 4.9: 保存结果

的时间间隔不能自由设定；画图时只能取少于或等于 30 个数据，不能随着记录的数据个数灵活变化；系统的功能相对简单，没有实现更深入的功能扩展和数据分析。

## 第5章 总结

在本次设计我曾遇到的如下一些问题：

### 1、size\_t strlen (const char string[])

strlen() 函数在计算输入或输出数组的字节数时具有遇零终止特性。在进行串口通信时由于传输字符串中含有“0”，所以一度造成通信错误。最后采用了人工计数方法加以克服。但人工计数方法具有繁杂的缺陷，不够智能化。

### 2、CRC 循环校验

CRC 循环校验从要进行校验的数组中的第一字节开始至 CRC 校验高字节前面的字节数据结束。CRC 校验的头文件中默认的变量数据类型为 unsigned short int，而主程序中的相关变量的数据类型与默认类型不同，导致 CRC 循环校验环节不能正常运行。

### 3、仪表波特率

由于仪表显示字符限制，厂家把仪表可选择的波特率分别编号，并且在仪表用户手册进行了说明。由于读数前没有认真阅读仪表用户手册，造成读数的误认。所以在读取仪表波特率时要对照仪表用户手册的编号表。

### 4、更新数据记录表格

记录数据的方法很多，初步方案为保存到 Excel，但由于相关的库函数不具有实时特性，所以修正方案，将数据写在面板的 Table 中。由于数据不能显示在空白的 Table 中，所以必须对 Table 进行初始化。经实验证明，初始化后再运行更新程序就可以从第一行开始依次显示数据了。开始由于采用 Toggle Button 的两个值控制数据记录，只能控制开始，所以系统只可以自动记录有限次数据。受循环召测的开始和停止的启发，选择在数据记录面板上添加时钟控件，用 Toggle Button 的两个值控制时钟的运行，通过对时钟控件的设置选择需要记录的数据。

### 5、保存数据

为了使界面更加友好，在保存数据时采用了信息对话框。由于缺少头文件 cmdlg.h，致使无法识别头文件中定义的数据类型 OPENFILENAME。

本次实验，从刚开始的毫无头绪、手忙脚乱到中间阶段的不断尝试和修改，到最后的按照预期的完成任务，我在老师的指导和自己的努力下学习到了很多知识。虽然本次设计基本符合预期要求，但是还是存在不足。针对这些不足和需要改进的地方，我

将在以后的学习中深入地研究相关方面的知识，争取更加熟练的掌握相关技术。

## 参考文献

- [1] 范逸之, 陈立元. Delphi 与 RS-232 串行通信控制 [M]. 北京: 清华大学出版社, 2002.
- [2] 赵炯, 熊肖磊, 周奇才. 自动化控制系统中通信协议设计研究 [J]. 计算机工程, 2002, 28(8): 83 – 85.
- [3] 谢瑞和. 串行技术大全 [M]. 北京: 清华大学出版社, 2003.
- [4] 方皓. 串行通信协议剖析与驱动程序设计研究 [D]. 上海: 同济大学, 2007.
- [5] AXELSON J. 串行端口大全 [M]. 北京: 中国电力出版社, 2001.
- [6] ELLEITHY K, ISKANDER M, KARIM M, et al. Advances in Computer, Information, and Systems Sciences, and Engineering: Proceedings of IETA 2005, TeNe 2005, EIAE 2005[C]. Dordrecht: Springer, 2006.
- [7] 青岛青智仪器有限公司. ZW1400 系列仪表用户手册 [K]. 青岛: 青岛青智仪器有限公司, 2008.
- [8] 苏瑞, 张春芳, 王立武. C 语言程序设计 [M]. 北京: 清华大学出版社, 2009.
- [9] KRUGLINSKI D J. Visual C++ 技术内幕 [M]. 潘爱民, 王国印, 译. 北京: 清华大学出版社, 2005.
- [10] 史君成, 张淑伟, 律淑珍. LabWindows 虚拟仪器设计 [M]. 北京: 国防工业出版社, 2007.
- [11] 张毅刚, 乔立岩. 虚拟仪器软件开发环境 LabWindows/CVI 6.0 编程指南 [M]. 北京: 机械工业出版社, 2002.
- [12] 刘君华. 基于 LabWindows/CVI 的虚拟仪器设计 [M]. 北京: 电子工业出版社, 2003.
- [13] 刘君华. 虚拟仪器编程语言 LabWindows/CVI 教程 [M]. 北京: 电子工业出版社, 2001.

## 附录 A 程序源代码

### A.1 循环校验头文件

```
////////////////////////////////////  
////  
//Name:      test.h  
//Purpose:   calculate the value of Cyclic Redundancy Check(CRC)  
//Author:    chen congmei  
//Created:   28/05/2009 09:06:12  
//Copyright: chen congmei  
////////////////////////////////////  
////  
/*=====*/  
计算子程序 CrC  
=====*/  
unsigned short int CrCCal(unsigned short int Data, unsigned short int  
    GenPoly, unsigned short int CrCData)  
{  
    unsigned short int TmpI;  
    Data*=2;  
    for(TmpI=8;TmpI>0;TmpI--)  
    {  
        Data=Data/2;  
        if((Data ^ CrCData)&1)CrCData=(CrCData/2)^ GenPoly;  
        else CrCData/=2;  
    }  
    return CrCData;  
}
```

### A.2 数据变换头文件

```
////////////////////////////////////  
////  
//Name:      convert.h
```

```

//Purpose:  convert one data type to another data type
//Author:   chen congmei
//Created:  28/05/2009  09:06:12
//Copyright: chen congmei
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////
/*=====
           公共变量
=====*/
union
{
    unsigned char uc[4];
    int          lda;
    unsigned int  ul;
    float        fda;
}un_4b;

union
{
    unsigned char uc[2];
    short int     ida;
    unsigned short int ui;
}un_2b;

int          lda;
short int    ida;
float        real;
unsigned char uca[4];
unsigned char ucb[2];

/*=====
浮点数据转换为字节数据
入口数据: real 放入要转换的浮点数据;
出口数据: 转换的四字节数据在 uca[] 中 顺序是从低 (uca[0]) 到高 (uca[3])

real=1.0; 转换为字节 uca[0]-uca[3]=0,0,0x80,0x3f
=====*/
void FtoB(void)
{
    un_4b.fda=real;
    uca[0]=un_4b.uc[3];
    uca[1]=un_4b.uc[2];
    uca[2]=un_4b.uc[1];
}

```

```

    uca[3]=un_4b.uc[0];

}

/*=====
字节数据转换为浮点数据
入口数据: 要转换的四字节数据在 uca[] 中 顺序是从低 (uca[0]) 到高 (uca[3])
出口数据: real 存放的为已转换的浮点数据;

数据 uca[0]-uca[3]=0,0,0x80,0x3f 转换为浮点 real=1.0
=====*/
void BtoF(void)
{
    un_4b.uc[0]=uca[3];
    un_4b.uc[1]=uca[2];
    un_4b.uc[2]=uca[1];
    un_4b.uc[3]=uca[0];
    real=un_4b.fda;
}

/*=====
长整形数据转换为字节数据
入口数据: 要转换的长整形放在 lda 中
出口数据: 转换完的四字节数据在 uca[] 中 顺序是从高 (uca[0]) 到第 (uca[3])

长整数据 lda=1000 转换的字节数据 uca[0]-uca[3]=0xe8,0x03,0,0
=====*/
void LtoB(void)
{
    un_4b.lda=lda;
    uca[0]=un_4b.uc[0];
    uca[1]=un_4b.uc[1];
    uca[2]=un_4b.uc[2];
    uca[3]=un_4b.uc[3];
}

/*=====
字节数据换为长整形数据转
入口数据: 转换完的四字节数据在 uca[] 中 顺序是从高 (uca[0]) 到第 (uca[3])
出口数据: 转换完毕的长整形放在 lda 中
    字节数据 uca[0]-uca[3]=0xe8,0x03,0,0 转换的长整形数 lda=1000
=====*/

```

```

void BtoL(void)
{
    un_4b.uc[0]=uca[3];
    un_4b.uc[1]=uca[2];
    un_4b.uc[2]=uca[1];
    un_4b.uc[3]=uca[0];
    lda=un_4b.lda;
}

/*=====
整形数据换为字节数据
入口数据: 要转换的整形放在 ida 中
出口数据: 转换完的 2 字节数据在 ucb[] 中 顺序是从高 (ucb[0]) 到第 (ucb[1])

要转换的整形数据 ida=1000, 转换的字节数据 ucb[0]-ucb[1]=0xe8,0x03
=====*/
void ItoB(void)
{
    un_2b.lda=ida;
    ucb[0]=un_2b.uc[1];
    ucb[1]=un_2b.uc[0];
}

/*=====
字节数据转换为整形数据
入口数据: 要转换的 2 字节数据在 ucb[] 中 顺序是从高 (ucb[0]) 到第 (ucb[1])
出口数据: 转换完毕的整形放在 ida 中
字节数据 ucb[0]-ucb[1]=0xe8,0x03 转换的整形数 ida=1000
=====*/
void BtoI(void)
{
    un_2b.uc[0]=ucb[1];
    un_2b.uc[1]=ucb[0];
    ida=un_2b.lda;
}

```

### A.3 主程序

```

////////////////////////////////////
/////
//Name:      comm.c
//Purpose:   communicate with ZW1402

```



```
//Author: chen congmei
//Created: 28/05/2009 09:06:12
//Copyright: chen congmei
/////////////////////////////////////////////////////////////////
/////

#include <ansi_c.h>
#include <stdlib.h>
#include <windows.h>
#include "asynctmr.h"
#include <formatio.h>
#include <rs232.h>
#include <cvirte.h>
#include <utility.h>
#include <userint.h>
#include <commdlg.h>
#include "comm.h"
#include "convert.h"
#include "test.h"
#include "ExcelReport.h"
#include "excel2000.h"

static CAObjHandle applicationHandle = 0;
static CAObjHandle workbookHandle = 0;
static CAObjHandle worksheetHandle = 0;
static CAObjHandle chartHandle = 0;
static int running = 0;
static int copytableDone = 0;
static int PlotType = ExRConst_GalleryArea;
static int panel_handle;
static int config_handle;
static int file_handle;

unsigned short int testout;
unsigned short int testin;
int i,j;
char devicename[30];
static int comport = 4,
          baudrate = 9600,
          status = 1;
static float data;
int portindex, RS232Error, config_flag, file_flag, addr,
    rate, baud, comm, flag, flag1, flag2, flag3;
```

```
float up, low, ir, upr, lowr, irr;
int numberOfRows, year, month, day;
Point cellp;
char date[11];

void DisplayRS232Error (void);
void SetConfigParms (void);
void GetConfigParms (void);
void SetFloat (void);
void SetUp (void);
void SetLow (void);
void SetIr (void);
void SetDevice (void);
void GetFloat (void);
void GetInt (void);
void Communi (void);
void RdData (void);

int main (int argc, char *argv[])
{
    if (InitCVIRTE (0, argv, 0) == 0)
        return -1;
    if ((panel_handle = LoadPanel (0, "comm.uir", PANEL)) < 0)
        return -1;

    DisplayPanel (panel_handle);
    RunUserInterface ();
    DiscardPanel (panel_handle);
    CloseCVIRTE ();
    return 0;
}

void SetConfigParms (void)
{
    SetCtrlVal (config_handle, CONFIG_COMPORT, comport);
    SetCtrlVal (config_handle, CONFIG_BAUDRATE, baudrate);
    SetCtrlVal (config_handle, CONFIG_UP, up);
    SetCtrlVal (config_handle, CONFIG_LOW, low);
    SetCtrlVal (config_handle, CONFIG_IR, ir);
    SetCtrlIndex (config_handle, CONFIG_COMPORT, portindex);
}
```

```
}

// Get the port configuration parameters.
void GetConfigParms (void)
{
    GetCtrlVal (config_handle, CONFIG_COMPORT, &comport);
    GetCtrlVal (config_handle, CONFIG_BAUDRATE, &baudrate);
    GetCtrlVal (config_handle, CONFIG_UP, &up);
    GetCtrlVal (config_handle, CONFIG_LOW, &low);
    GetCtrlVal (config_handle, CONFIG_IR, &ir);
    GetCtrlIndex (config_handle, CONFIG_COMPORT, &portindex);
#ifdef _NI_unix_
    devicename[0]=0;
#else
    GetLabelFromIndex (config_handle, CONFIG_COMPORT, portindex,
                      devicename);
#endif
}

//open the configuration panel.
int CVICALLBACK Estab (int panel, int control, int event,
                      void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
            config_handle = LoadPanel (panel_handle, "comm.uir", CONFIG);
            InstallPopup (config_handle);
            if (config_flag) /* Configuration done at least once.*/
                SetConfigParms ();
            else /* 1st time.*/
                config_flag = 1;
            break;
    }
    return 0;
}

//close the configuration panel.
int CVICALLBACK Over (int panel, int control, int event,
                     void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
```

```
        case EVENT_COMMIT:
            GetConfigParms ();
            DisableBreakOnLibraryErrors ();
            RS232Error = OpenComConfig (comport, devicename, baudrate, 0,8,
                1, 512, 512);
            EnableBreakOnLibraryErrors ();
            if (RS232Error == 0)
            {
                SetDevice ();
            }
            else
                DisplayRS232Error ();

            DiscardPanel (config_handle);
            break;
    }
    return 0;
}

void SetUp (void)
{
    unsigned char out[13] = {1,0x10,0,0,0,2,4};
    unsigned char in[8];
    out[2] = 0x19;
    out[3] = 0;
    real = up;
    FtoB();
    out[7]=uca[0];
    out[8]=uca[1];
    out[9] = uca[2];
    out[10] = uca[3];
    testout = 0xffff;
    for(i = 0;i < 11;i = i + 1)testout = CrcCal(out[i],0xa001,testout);
    out[11] = testout%256;
    out[12] = testout/256;
    FlushInQ (comport);
    ComWrt (comport, out, 13);
    ComRd (comport, in, 8);
    if(in[1] == 0x10) flag1 = 0;
    if(in[1] == 0x90) flag1 = 1;
}

void SetLow (void)
```

```
{
    unsigned char out[13] = {1,0x10,0,0,0,2,4};
    unsigned char in[8];
    out[2] = 0x19;
    out[3] = 2;
    real = low;
    FtoB();
    out[7]=uca[0];
    out[8]=uca[1];
    out[9] = uca[2];
    out[10] = uca[3];
    testout = 0xffff;
    for(i = 0;i < 11;i = i + 1)testout = CrcCal(out[i],0xa001,testout);
    out[11] = testout%256;
    out[12] = testout/256;
    FlushInQ (comport);
    ComWrt (comport, out, 13);
    ComRd (comport, in, 8);
    if(in[1] == 0x10) flag2 = 0;
    if(in[1] == 0x90) flag2 = 1;
}
```

```
void SetIr (void)
```

```
{
    unsigned char out[13] = {1,0x10,0,0,0,2,4};
    unsigned char in[8];
    out[2] = 0x1f;
    out[3] = 2;
    real=ir;
    FtoB();
    out[7]=uca[0];
    out[8]=uca[1];
    out[9] = uca[2];
    out[10] = uca[3];
    testout = 0xffff;
    for(i = 0;i < 11;i = i + 1)testout = CrcCal(out[i],0xa001,testout);
    out[11] = testout%256;
    out[12] = testout/256;
    FlushInQ (comport);
    ComWrt (comport, out, 13);
    ComRd (comport, in, 8);
    if(in[1] == 0x10) flag3 = 0;
    if(in[1] == 0x90) flag3 = 1;
}
```

```

}

void SetDevice (void)
{
    SetUp ();
    SetLow ();
    SetIr ();
    flag = flag1 + flag2 + flag3;
    if(flag1 == 1)SetCtrlVal (panel_handle, PANEL_TEXT, " 通讯状态: 上限
        :          CRC 校验错! ");
    if(flag2 == 1)SetCtrlVal (panel_handle, PANEL_TEXT, " 通讯状态:
        下限:          CRC 校验错! ");
    if(flag3 == 1)SetCtrlVal (panel_handle, PANEL_TEXT, " 通讯状态:
        CT: CRC 校验错! ");
    if(flag == 0)SetCtrlVal (panel_handle, PANEL_TEXT, " 通讯状态: 设置成
        功! ");
}

//communicate with the device.
void Communi (void)
{
    unsigned char outbuf[8] = {1,3,0,0,0,0};
    unsigned char inbuf[9];
    outbuf[2] = 0x10;
    outbuf[3] = 0x02;
    outbuf [5] = 2;
    testout=0xffff;
    for(i = 0;i < 6;i = i+1)testout = CrcCal(outbuf[i],0xa001,testout);
    outbuf[6] = testout%256;
    outbuf[7] = testout/256;
    FlushInQ (comport);
    ComWrt (comport, outbuf, 8);
    ComRd (comport, inbuf, 9);
    if(inbuf[1] == 3)
    {
        SetCtrlVal (panel_handle, PANEL_TEXT, " 通讯状态: 成功! ");
        uca[0] = inbuf[3];
        uca[1] = inbuf[4];
        uca[2] = inbuf[5];
        uca[3] = inbuf[6];
        BtoF();
    }
}

```

```

    if(inbuf[1] == 0x83)
    {
        SetCtrlVal (panel_handle, PANEL_TEXT, " 通讯状态: 电流:
            通讯超时! ");
    }
    data = real;
    SetCtrlVal (panel_handle, PANEL_DATA, data);
}

void GetIr (void)
{
    unsigned char outbuf[8] = {1,3,0,0,0,0};
    unsigned char inbuf[9];
    outbuf[2] = 0x1f;
    outbuf[3] = 0x02;
    outbuf [5] = 2;
    testout=0xffff;
    for(i = 0;i < 6;i = i+1)testout = CrcCal(outbuf[i],0xa001,testout);
    outbuf[6] = testout%256;
    outbuf[7] = testout/256;
    FlushInQ (comport);
    ComWrt (comport, outbuf, 8);
    ComRd (comport, inbuf, 9);
    if(inbuf[1] == 3)
    {
        SetCtrlVal (panel_handle, PANEL_TEXT, " 通讯状态: 成功! ");
        uca[0] = inbuf[3];
        uca[1] = inbuf[4];
        uca[2] = inbuf[5];
        uca[3] = inbuf[6];
        BtoF();
    }
    if(inbuf[1] == 0x83)
    {
        SetCtrlVal (panel_handle, PANEL_TEXT, " 通讯状态:      CT:
            通讯超时! ");
    }
    irr = real;
    SetCtrlVal (panel_handle, PANEL_IRR, irr);
}

void GetUp (void)

```

```
{
    unsigned char outbuf[8] = {1,3,0,0,0,0};
    unsigned char inbuf[9];
    outbuf[2] = 0x19;
    outbuf[3] = 2;
    outbuf [5] = 2;
    testout=0xffff;
    for(i = 0;i < 6;i = i+1)testout = CrcCal(outbuf[i],0xa001,testout);
    outbuf[6] = testout%256;
    outbuf[7] = testout/256;
    FlushInQ (comport);
    ComWrt (comport, outbuf, 8);
    ComRd (comport, inbuf, 9);
    if(inbuf[1] == 3)
    {
        SetCtrlVal (panel_handle, PANEL_TEXT, " 通讯状态: 成功! ");
        uca[0] = inbuf[3];
        uca[1] = inbuf[4];
        uca[2] = inbuf[5];
        uca[3] = inbuf[6];
        BtoF();
    }
    if(inbuf[1] == 0x83)
    {
        SetCtrlVal (panel_handle, PANEL_TEXT, " 通讯状态:
            下限: 通讯超时! ");
    }
    lowr = real;
    SetCtrlVal (panel_handle, PANEL_LOWR, lowr);
}

void GetLow (void)
{
    unsigned char outbuf[8] = {1,3,0,0,0,0};
    unsigned char inbuf[9];
    outbuf[2] = 0x19;
    outbuf[3] = 0;
    outbuf [5] = 2;
    testout=0xffff;
    for(i = 0;i < 6;i = i+1)testout = CrcCal(outbuf[i],0xa001,testout);
    outbuf[6] = testout%256;
    outbuf[7] = testout/256;
```



```
FlushInQ (comport);
ComWrt (comport, outbuf, 8);
ComRd (comport, inbuf, 9);
if(inbuf[1] == 3)
{
    SetCtrlVal (panel_handle, PANEL_TEXT, " 通讯状态: 成功! ");
    uca[0] = inbuf[3];
    uca[1] = inbuf[4];
    uca[2] = inbuf[5];
    uca[3] = inbuf[6];
    BtoF();
}
if(inbuf[1] == 0x83)
{
    SetCtrlVal (panel_handle, PANEL_TEXT, " 通讯状态:
        上限:          通讯超时! ");
}

upr = real;
SetCtrlVal (panel_handle, PANEL_UPR, upr);
}

void GetAddr (void)
{
    unsigned char outbuf[8] = {1,3,0,0,0,0};
    unsigned char inbuf[9];
    outbuf[2] = 0x1f;
    outbuf[3] = 0x51;
    outbuf[5] = 1;
    testout = 0xffff;
    for(i=0;i<6;i=i+1)testout=CrcCal(outbuf[i],0xa001,testout);
    outbuf[6]=testout%256;
    outbuf[7]=testout/256;
    FlushInQ (comport);
    ComWrt (comport, outbuf, 8);
    ComRd (comport,inbuf,7);
    if(inbuf[1] == 3)
    {
        SetCtrlVal (panel_handle, PANEL_TEXT, " 通讯状态: 成功! ");
        ucb[0]=inbuf[3];
        ucb[1]=inbuf[4];
        BtoI();
    }
}
```

```

    }
    if(inbuf[1] == 0x83)
    {
        SetCtrlVal (panel_handle, PANEL_TEXT, " 通讯状态:           地址
                :           通讯超时! ");
    }
    addr = ida;
    SetCtrlVal (panel_handle, PANEL_ADDR, addr);
}

void GetBaud (void)
{
    unsigned char outbuf[8] = {1,3,0,0,0,0};
    unsigned char inbuf[9];
    outbuf[2] = 0x1f;
    outbuf[3] = 0x52;
    outbuf[5] = 1;
    testout = 0xffff;
    for(i=0;i<6;i=i+1)testout=CrcCal(outbuf[i],0xa001,testout);
    outbuf[6]=testout%256;
    outbuf[7]=testout/256;
    FlushInQ (comport);
    ComWrt (comport, outbuf, 8);
    ComRd (comport,inbuf,7);
    if(inbuf[1] == 3)
    {
        SetCtrlVal (panel_handle, PANEL_TEXT, " 通讯状态: 成功! ");
        ucb[0]=inbuf[3];
        ucb[1]=inbuf[4];
        BtoI();
    }
    if(inbuf[1] == 0x83)
    {
        SetCtrlVal (panel_handle, PANEL_TEXT, " 通讯状态:
                速率:           通讯超时! ");
    }
}

switch (ida)
{
    case 0: rate = 300;break;

```

```
        case 1: rate = 600;break;
        case 2: rate = 1200;break;
        case 3: rate = 2400;break;
        case 4: rate = 4800;break;
        case 5: rate = 9600;break;
    }
    SetCtrlVal (panel_handle, PANEL_RATE, rate);
}

void RdData (void)
{
    SetCtrlVal (panel_handle, PANEL_COMM, comport);
    SetCtrlVal (panel_handle, PANEL_BAUD, baudrate);
    GetIr ();
    GetUp ();
    GetLow ();
    GetAddr ();
    GetBaud ();
}

int CVICALLBACK Measure (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
            OpenComConfig (comport, devicename, baudrate, 0, 8, 1, 512, 512);
            Communi ();
            RdData ();
            break;
    }
    return 0;
}

int CVICALLBACK Cycle (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
            SetCtrlAttribute (panel_handle, PANEL_STOP, ATTR_VISIBLE, 1);
            SetCtrlAttribute (panel_handle, PANEL_REC, ATTR_VISIBLE, 1);
            SetCtrlAttribute(panel_handle,PANEL_TIMER,ATTR_ENABLED,1);
    }
}
```

```
        OpenComConfig (comport, devicename, baudrate, 0, 8, 1, 512, 512);
        RdData ();
        break;
    }
    return 0;
}
```

*//measure datas cycly.*

```
int CVICALLBACK Time (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_TIMER_TICK:
            OpenComConfig (comport, devicename, baudrate, 0, 8, 1, 512, 512);
            Communi();
            break;
    }
    return 0;
}
```

*//record the data.*

```
int CVICALLBACK Rec (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
            file_handle = LoadPanel (panel_handle, "comm.uir", FILE);
            InstallPopup (file_handle);
            break;
    }
    return 0;
}
```

*//stop measure cycly.*

```
int CVICALLBACK Stop (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
```

```
        case EVENT_COMMIT:
            SetCtrlAttribute(panel_handle,PANEL_TIMER,ATTR_ENABLED,0);
            SetCtrlAttribute (panel_handle, PANEL_REC, ATTR_VISIBLE, 0);
            break;
    }
    return 0;
}

//Display error information to the user.
void DisplayRS232Error (void)
{
    char ErrorMessage[200];
    switch (RS232Error)
    {
        default :
            if (RS232Error < 0)
            {
                Fmt (ErrorMessage, "%s<RS232 error number %i", RS232Error);
                MessagePopup ("RS232 Message", ErrorMessage);
            }
            break;
        case 0 :
            MessagePopup ("RS232 Message", "No errors.");
            break;
        case -2 :
            Fmt (ErrorMessage, "%s", "Invalid port number (must be in
                the " "range 1 to 8).");
            MessagePopup ("RS232 Message", ErrorMessage);
            break;
        case -3 :
            Fmt (ErrorMessage, "%s", "No port is open.\n"
                "Check COM Port setting in Configure.");
            MessagePopup ("RS232 Message", ErrorMessage);
            break;
        case -99 :
            Fmt (ErrorMessage, "%s", "Timeout error.\n\n"
                "Either increase timeout value,\n"
                "    check COM Port setting, or\n"
                "    check device.");
            MessagePopup ("RS232 Message", ErrorMessage);
            break;
    }
}
```

```
//close the comm panel. */
int CVICALLBACK QuitCallback (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
            SetCtrlAttribute(panel_handle,PANEL_TIMER,ATTR_ENABLED,0);
            QuitUserInterface(0);
            break;
    }
    return 0;
}

int CVICALLBACK SaveFile (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{
    int fileHandle, numberOfRows;
    Rect cellRange;
    char *logtime[2];
    float curval[2];

    //获取用户选择保存的文件名称。
    OPENFILENAME ofn; // 公共对话框结构。
    TCHAR szFile[MAX_PATH]; // 保存获取文件名称的缓冲区。
    TCHAR strDefExt[3] = "csv";

    // 初始化选择文件对话框。
    ZeroMemory(&ofn, sizeof(ofn));
    ofn.lStructSize = sizeof(ofn);
    ofn.lpstrFile = szFile;
    ofn.lpstrFile[0] = '\\0';
    ofn.nMaxFile = sizeof(szFile);
    ofn.lpstrFilter =
        "CSV (*.csv)\0*.CSV\0Text (*.txt)\0*.TXT\0All (*.*)\0*.*\0";
    ofn.nFilterIndex = 1;
    ofn.lpstrFileTitle = NULL;
    ofn.nMaxFileTitle = 0;
    ofn.lpstrInitialDir = NULL;
    ofn.lpstrDefExt = strDefExt;
```

```
switch (event)
{
    case EVENT_COMMIT:
        if (GetSaveFileName(&ofn))
        {
            fileHandle = OpenFile (szFile, VAL_WRITE_ONLY, VAL_TRUNCATE,
                                   VAL_ASCII)
            FmtFile (fileHandle, " 日期, 时间, 电流 (A), 倍率\n");
            GetNumTableRows (file_handle, FILE_TABLE, &numberOfRows);
            cellRange.height = 1;
            cellRange.width = 2;
            for (j = 1; j <= numberOfRows; j++)
            {
                cellRange.top = j;
                cellRange.left = 1;
                GetTableCellRangeVals (file_handle, FILE_TABLE,
                                         cellRange, logtime, VAL_ROW_MAJOR);
                cellRange.left = 3;
                GetTableCellRangeVals (file_handle, FILE_TABLE,
                                         cellRange, curval, VAL_ROW_MAJOR);

                FmtFile (fileHandle, "%s,%s,%f[p3],%f[p3]\n", logtime[0],
                        logtime[1], curval[0], curval[1]);
            }
            FreeTableValStrings (logtime, 2);
            CloseFile (fileHandle);
        }

        if (worksheetHandle)
            CA_DiscardObjHandle(worksheetHandle);
        if (chartHandle)
            CA_DiscardObjHandle(chartHandle);
        if (workbookHandle)
        {
            ExcelRpt_WorkbookClose(workbookHandle, 0);
            CA_DiscardObjHandle(workbookHandle);
        }
        if (applicationHandle)
        {
            ExcelRpt_ApplicationQuit(applicationHandle);
            CA_DiscardObjHandle(applicationHandle);
        }
        DiscardPanel (file_handle);
    }
```

```
        break;
    }
    return 0;
}

void InitTable (void)
{
    GetSystemDate (&month, &day, &year);
    GetNumTableRows (file_handle, FILE_TABLE, &numberOfRows);
    cellp.x = 1; cellp.y = 1;
    Fmt(date, "%i[w4]%i[w2p0]%i[w2p0]", year, month, day);
    SetTableCellVal (file_handle, FILE_TABLE, cellp, date);
    cellp.x++;
    SetTableCellVal (file_handle, FILE_TABLE, cellp, TimeStr ());
    cellp.x++;
    SetTableCellVal (file_handle, FILE_TABLE, cellp, data);
    cellp.x++;
    SetTableCellVal (file_handle, FILE_TABLE, cellp, irr);
}

int CVICALLBACK Switch (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
            GetCtrlVal (file_handle, FILE_SWITCH, &status);
            switch (status)
            {
                case 1: InitTable ();
                    SetCtrlAttribute(file_handle, FILE_RECORD, ATTR_ENABLED, 1);
                    break;
            case 0: SetCtrlAttribute(file_handle, FILE_RECORD, ATTR_ENABLED, 0);
                    SetCtrlAttribute(file_handle, FILE_OPEN, ATTR_DIMMED, 0);
                    break;
            }
            break;
    }
    return 0;
}

int CVICALLBACK Record (int panel, int control, int event,
```



```

        void *callbackData, int eventData1, int eventData2)
    {

        switch (event)
        {
            case EVENT_TIMER_TICK:
                GetSystemDate (&month, &day, &year);
                GetNumTableRows (file_handle, FILE_TABLE, &numberOfRows);
                Communi ();
                InsertTableRows (file_handle, FILE_TABLE,
                    numberOfRows + 1, 1, VAL_USE_MASTER_CELL_TYPE);
                cellp.x = 1; cellp.y = numberOfRows + 1;
                Fmt(date, "%i[w4]%i[w2p0]%i[w2p0]", year, month, day);
                SetTableCellVal (file_handle, FILE_TABLE, cellp, date);
                cellp.x++;
                SetTableCellVal (file_handle, FILE_TABLE, cellp, TimeStr ());
                cellp.x++;
                SetTableCellVal (file_handle, FILE_TABLE, cellp, data);
                cellp.x++;
                SetTableCellVal (file_handle, FILE_TABLE, cellp, irr);
                break;
        }
        return 0;
    }

int CVICALLBACK Open (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
            ExcelRpt_ApplicationNew(1, &applicationHandle);
            ExcelRpt_WorkbookNew(applicationHandle, &workbookHandle);
            ExcelRpt_WorksheetNew(workbookHandle, -1, &worksheetHandle);
            ExcelRpt_WriteDataFromTableControl (worksheetHandle, "A1:C30",
                file_handle, FILE_TABLE);
            ExcelRpt_RangeBorder (worksheetHandle, "A1:C30",
                ExRConst_Continuous, 255, ExRConst_Thin,
                ExRConst_InsideHorizontal |
                ExRConst_InsideVertical | ExRConst_EdgeBottom
                |ExRConst_EdgeLeft
                |ExRConst_EdgeRight|ExRConst_EdgeTop);
            SetCtrlAttribute (file_handle, FILE_GRAPH, ATTR_DIMMED, 0);
    }
}

```

```

        break;
    }
    return 0;
}

int CVICALLBACK Graph (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
            ExcelRpt_SetCellRangeAttribute (worksheetHandle, "A2:A10",
                ER_CR_ATTR_COLUMN_WIDTH, 5.0);
            ExcelRpt_ChartAddtoWorksheet (worksheetHandle, 2.0 * 72, 7.0,
                10.0*72, 4.0*72, &chartHandle);
            ExcelRpt_ChartWizard (chartHandle, worksheetHandle, "C1:C30",
                PlotType,0, 0, 0, 0, 1, "Value Time",
                "Time", "Value", NULL);
            ExcelRpt_SetCellRangeAttribute (worksheetHandle, "A1",
                ER_CR_ATTR_COLUMN_WIDTH, 10.0);
            ExcelRpt_SetChartAttribute (chartHandle,
                ER_CH_ATTR_PLOTAREA_COLOR,0xffffffff);
            SetCtrlAttribute (file_handle, FILE_RING, ATTR_DIMMED, 0);
            running = 1;
            break;
    }
    return 0;
}

int CVICALLBACK ChartSelect (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
            GetCtrlVal (file_handle, FILE_RING, &PlotType);
            if (running) {
                ExcelRpt_ChartWizard (chartHandle, worksheetHandle, "C1:C30",
                    PlotType,0, 0, 0, 0, 1, "Value Time",
                    "Time", "Value", NULL);
            }
            break;
    }
}

```

```
    return 0;  
}
```