接 口 技 术 课 程 设 计

设计报告

题目名称：	**构造自动测量系统**
所在学院：	**XXXXXX 学院**
所学专业：	**XXXXXXX**
学生姓名：	XXXXX　XXXXXXX
指导老师：	**XXXXX**

（二〇一九年陆月）

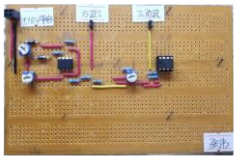
摘要

现在，计算机的应用已经相当普遍，传统的测量仪器逐渐被数字化测量仪器所取代。连接的手工测量工作很多都更新换代为由计算机控制的自动测试系统，这是电子测量领域发展的必然趋势。

经过对本课程设计的分析，本构造自动测量系统将通过 MFC 上位机完成程控接口控制数字信号发生器输出激励信号，采用单文档方式显示响应信号波形，以达到课程设计的完成相关要求。信号发生器的程控接口，采用 RS232 硬件接口连接，根据程控命令格式与信号发生器通信，控制输出激励信号的幅度、频率等参数。同样地，使用 USB-C 与示波器相连，实现接口程序控制示波器工作。

通过 MFC 编写上位机显示采集到的信号波形，提供键鼠操作图像的缩放、保存等功能。经实际测试，不断调节激励信号的显示 X、Y 轴及参数，系统测量显示已达到稳定性、快速性、准确性的基本要求，能够较好的满足一般测试要求。

关键字： MFC；RS232；数字信号发生器；示波器；数字程控



目 录

第一部分 课程设计概述

1.1 课程设计的目的与任务

1.使用智能仪器构造自动测量系统

自动测量是计算机控制下的测量过程，必须对单次测量有充分理解，才能实现连续的自动过程。

基本的测量过程都是：

产生激励信号 -> 经过被测系统 -> 产生输出的响应信号

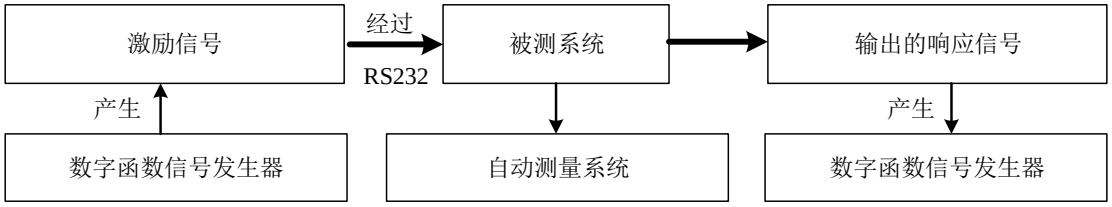


图 1.1 测量原理图

对激励信号与响应信号的分析，可以获得被测系统的特性。例如向直流分压电路施加一个电压信号，在输出端得到对应的分压电压。比较输出和输入的电压大小，可以得到该电路的分压比等特性。

2.使用 MFC 实现程序结构

实验使用 MFC 应用程序实现，对话框或 SDI、MDI 都可以，自己选择。

典型的测量系统有：

交流电路的频率特性；

二极管的电压-电流特性测量；

直流放大特性的测量等等。

要求：

基本要求是没有被测系统，直接使用示波器测量信号发生器的输出。

1.2 课程设计题目

构造自动测量系统

1.3 设计功能要求

（1）在 windows 系统下通过接口与仪器实现通信

常用的仪器接口有： USB，RS232C，以太网，GPIB，PCI 等。

实验室中提供的有： 使用 RS232C 的程控信号发生器，数字直流电压表 PZ-150 使用 USB 的存储示波器

接口只是硬件的实现，更重要的是程控命令的功能和定义。

（2）在程序界面上显示激励信号；

（3）在程序界面上显示测量到的响应信号

智能示波器采用的是 USB 接口，但是使用 USB 中 CDC 方式构造了虚拟串口，因此在程序中表现为串口编程。

采用边测量边画点的方法，可以画出响应波形。

（4）在程序界面上显示测量系统的测量获得的结果特性

按照被测量特性的定义要求，由激励与响应计算出特性曲线。

（5）扩展部分自选测量内容，设计被测系统（如滤波器的频率特性），实现完整的测量系统。

1.4 课程设计的内容与要求

（1）设计构造自动测量系统；

（2）编写测量系统软件；

（3）软硬件调试；

（4）写课程设计报告。

1.5 实验仪器设备及器件

1）PC 机，Windows 10；

2）Visual C++ 6.0；

3）ATTEN ATF20B 数字信号发生器，Tektronix TDS1012C-EDU 示波器。

第二部分 设计方案工作原理

经过对本课程设计的分析，本构造自动测量系统将通过 MFC 上位机完成程控接口控制数字信号发生器输出激励信号，采用单文档方式显示响应信号，以达到课程设计的完成相关要求。信号发生器的程控接口，采用 RS232 硬件接口连接，根据程控命令格式与信号发生器通信，控制输出激励信号的幅度、频率等参数。同理，使用 USB-C 与示波器相连，实现接口程序控制示波器工作。

通过 MFC 编写上位机显示采集到的信号波形，提供按键操作缩放、保存等功能。不断调节激励信号的显示 X、Y 轴及参数，使系统测量显示达到稳定性、快速性、准确性的基本要求。

2.1 预期实现目标定位

本构造自动测量系统将通过 MFC 上位机完成程控接口控制数字信号发生器输出激励信号，采用单文档方式显示响应信号，以达到课程设计的完成相关要求。更为具体地，硬件接口 RS232 一端连接 PC，一端连接 ATTEN ATF20B 数字信号发生器的程控接口，根据信号发生器的程控指令格式，编写程序发送指令控制信号波形的产生，调节信号的幅度、频率。

MFC 编写示波器显示程序，显示信号的波形。同理，使用 RS232 与示波器相连，实现接口程序控制示波器工作。系统结构框图如图 2.1 所示。

2.2 技术方案分析

2.2.1 系统框图

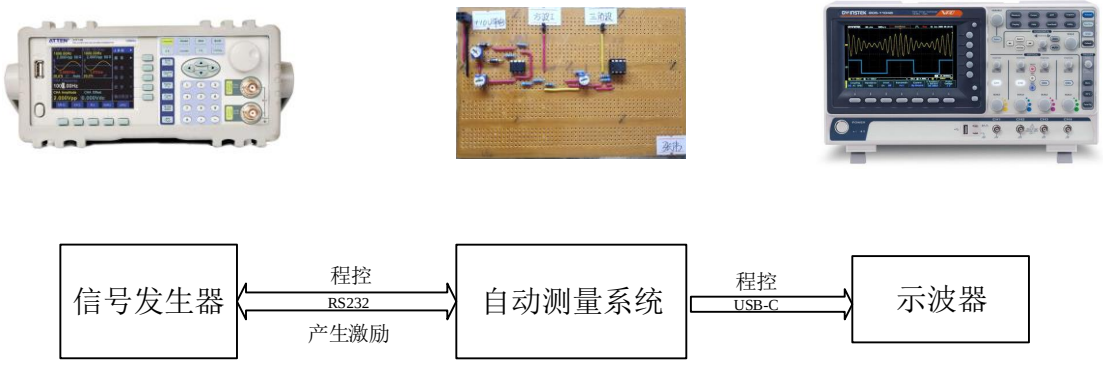


图 2.1 系统框图

2.2.2 信号发生器

1.概述

ATF20B DDS 函数信号发生器采用数字合成技术，具有快速完成测量工作所需的高性能指标和众多的功能特性。其简单而功能明晰的前面板设计和中/英文液晶显示界面能使您更便于操作和观察，可扩展的选件功能，可使您获得增强的系统特性。

图 2.2 ATF20B DDS 函数信号发生器

2.函数信号发生器技术指标

1）输出 B 特性

波形特性

波形种类	正弦波，方波，三角波，锯齿波，脉冲等 32 种波形
波形长度	1024 点 采样速率：12.5MSa/S
波形幅度分辨率	8bits
方波占空比	1%~99%

频率特性

频率范围	正弦波：40mHz~1MHz 其它波形：10mHz~100kHz
频率分辨率	10mHz
频率准确度	±（1×10-5 10mHz）

幅度特性

幅度范围	50mVpp~20Vpp (高阻) 分辨率：20mVpp
输出阻抗	50Ω

猝发特性

载波信号	B 路信号
触发信号	TTL_B 路信号
猝发计数	1-65000 个周期
猝发方式	内部 TTL，外部，单次

2）TTL 输出特性

●波形特性：方波，上升下降时间≤20nS

●频率特性：40mHz~1MHz

●幅度特性：TTL，CMOS 兼容，低电平 < 0.3V，高电平 > 4V

2.2.3 程控方式

1.接口应用

目前国内外中高档测量仪器几乎全部带有程控接口。不管任何种类，任何型号的仪器，只要带有这种接口，就可以使用一条电缆线把它们与计算机连接起来。组成一个自动测试系统。

在测量过程中，系统内各种仪器之间通过接口和电缆线进行数据交换和传输。根据事先编制好的测试程序，计算机准确地控制各种仪器进行协调一致的工作。例如，首先命令信号发生器给被测对象提供一个合适的信号，再命令频率计，电压表测量出相应的频率数据和电压数据，就后由计算机作数据处理，最后送打印机打印出测试服告。这就使得各种繁琐复杂的测试任务全部由测试系统自动完成，测试人员只要编制好测试程序就可以得到测试结果了。不但节省了人力，提高了效率，而且测试结果准确可靠，减少了人为的差错和失误，甚至可以完成一些手工测量无法完成的工作。

2.程控命令

程控命令是计算机通过接口向被控设备发送的系列 ASCII 码字符串，被控设备根据程控命令进行工作。每台仪器的程控命令都有各自规定的格式和定义，用户在编写应用程序时必须严格遵守这些规定，才能准确地控制这台仪器完成各项工作。

控制命令：1　　控制命令：2(不带数值、单位)　　查询命令：

CHAFREQ1L31.MHz　　CHA：SQUAR　　CHA：?AFREQ

↑↑↑　　↑↑↑↑　　↑↑↑　　↑↑↑

1 5 2　　5 3　5 4　　1 5 2　　1 6 2

1：功能命令　2：选项命令 3：数据命令 4：单位命令 5：分隔符 6：查询符

程控命令有功能命令、选项命令、数据命令、单位命令、分隔符、查询符组成，有限命令可以不带数据命令和单位命令。

1）命令编码

仪器的程控命令分为功能命令、选项命令、数据命令和单位命令四部分，如下表所示。功能命令和选项命令使用大写英文字母组成，其定义和一起的功能和选项一一对应，数据命令由 0-9 十个数字、小数点和负号组成。单位命令根据数据的性质来选择，使用规定的大写和小写英文字母组成，出表中规定的命令之外，其他字符串都不允许使用，否则将会出错。

功能命令表：

功能	命令	功能	命令
A 路单频	CHA	A 路猝发	ABURST
B 路单频	CHB	B 路猝发	BBURST
A 路扫频	FSWP	返回本地	LOCAL
A 路扫幅	ASWP	系统	SYS
外部测频	COUNT	TTL	TTL

2）结束符

一个程控位令字符串中的字将总数不得超过 60 个，每个字符串末尾都必须加结束符 Chr（10），表示一个字符串结束，否则会产生错误。

3）分隔符

程控命令中功能命令与选项命令之间、选项命令与数据命令之间、数据命令与单位命令之间、功能命令与查询符之间，必须插入间隔符。

4）查询符

在选项命令前加查询符（？），将返回对应的数值及单位，无单位只返回数值。

5）数据命令

数据命令，是大长度在 10 个字符。

6）控制命令

程控命令几乎可以控制仪器的全部功能，以上位软件机显示的控制功能为准。

7）串口控制

串控制，先发送机器地址，选择对应的机器，再发送程控命令。

8）联机操作

先发送接口控制选择命令，例如：选择 RS232 接口，无发送“地址+ RS232”，本机切换到 RS232 程控模式。例如：仪器的系统菜单的程控地址是 88 (十进制)，则发送“88RS232”，退出程控模式，发送“88LOCAL”返回按键操作，或按[系统]键返回按键操作，否则按键不能操作，其他的远程控制也不可用。

3.应用程序

所谓应用程序，也就果在自动测试系统中，测试人员为了准确地控制各种仪器设备而给计算机(系统控制者)编制的控制程序。

1）进入程控

开机后仪器工作在手动操作状态，当接受到计算机的程控命令后，仪器进人程控操作状态，仪器全部按键失去作用，一起只能根据计算机发出的程控命令进行工作，如果需要恢复手动操作状态，计算机可以发送“返回本地”命令“LOCAL”，仪器回到手动操作状态，全部按键恢复功能。

2）编程要点

在使用程控命令编写应用程序时应注意以下几点：

要点 1：必须严格遵守仪器的程控命令码，包话命令码字符的大小写；

要点 2：应该首先熟悉仪器的手动规作。

3）应用实例

下面给出一些不同类型的程控命令应用实例，仅供参考。

例 1：A 用单频输出，正弦波形，频率 1MHz，程控命令加下：

RS232 模式：88CHA AWAVE：0No.

BBCHA AFREQ1 MHE

例 2：B 路单频输出：三角波，频率 1kHz，程控命令如下：

RS212 模式：88CHB：BWAVE：2No.

88BCHB BTRIG 1 kHz

例 3：A 路单频输出，脉冲波形，占空比 25%，程控命令如下

RS232 模式 88CHA ADUTY25%

2.2.4 数字示波器

1.概述

GDS-1000B 系列示波器同时具备 100MHz 以及 70Mhz 两种频宽选择，搭载四通道或者两通道之模拟信号输入端。其中，单一通道实时采样率达 1GSa/s，存储器长度为每通道独立 10Mpts。GDS-1000B 同时具备每秒 50000 次波形更新率，让使用者更精确的观察到波形的细微变化。

图 2.3 GDS-1000B 示波器

2.函数信号发生器技术指标

		GDS-1072B	GDS-1074B	GDS-1102B	GDS-1104B
垂直系统	通道数	2 通道+1 外部触发通道	4 通道	2 通道+1 外部触发通道	4 通道
	带宽	DC~70MHz (-3dB)	DC~70MHz (-3dB)	DC~100MHz (-3dB)	DC~100MHz (-3dB)
	上升时间	5ns	5ns	3.5ns	3.5ns
	带宽限制	20MHz	20MHz	20MHz	20MHz
	垂直分辨率	8 bit: 1mV~10V			
	输入耦合	AC, DC, GND			
	输入阻抗	1MΩ// 约 16pF			
	直流精确度	±3%			
	极性	正向 &反向			
	最大输入电压	300V (DC+AC Peak), CAT I			
	偏移范围	1mV/div : ±1.25V			
		2mV/div ~ 100mV/div : ±2.5V			
		200mV/div ~ 10V/div : ±125V			
	波形信号处理	+, -, ×, ÷, FFT, FFTrms , 用户自定义			
		FFT: 1MPTS 点分辨率. FFT 垂直刻度提供 Linear RMS 或 dBV RMS. FFT 窗函数提供 Rectangular, Hamming, Hanning, 以及 Blackman-Harris.			

3.触发系统

触发系统	来源	CH1 ,CH2, CH3*,CH4*,Line, EXT** （*仅适合四通道机种，**仅适合双通道机种）
	触发模式	自动 (低于 100 ms/div 支持慢扫滚动模式), 一般, 单次
		边沿, 脉波宽度, 视频, Pulse Runt, 上升&下降沿, 交替, 事件延迟(可选择 1~65535 事件),时间延迟(可选择 4ns~10s 时间), 总线
	触发延迟时间	4ns ~ 10s
	耦合选项	AC,DC,LF rej. ,HF rej. ,Noise rej.
	灵敏度	1 div
外部触发	范围	±15V
	灵敏度	DC ~ 100MHz 约 100mV；100MHz~200MHz 约 150mV
	输入阻抗	1MΩ±3%// 16pF

4.显示系统

显示系统	显示器	7" TFT LCD 彩色显示, 800 水平 × 480 垂直 (WVGA)
	插补点方式	Sin(x)/x
	波形显示方式	点, 向量, 可调余晖 (16ms~4s), 无限余晖
	波形更新率	最快每秒 50,000 次波形更新
	显示模式	YT ; XY
	显示网格线	8 x 10 格

5.接口

接口	USB Port	USB 2.0 高速 host 接口 X1, USB 高速 2.0 device 接口 X1
	以太网络	RJ-45 接口, 10/100Mbps with HP Auto-MDIX（仅适用于 GDS-1074B,GDS-1104B)
	Go-NoGo BNC	最大 5V /10mA TTL 开集极输出
	安全锁孔	机器背面提供标准 Kensington 安全插槽

2.3 功能指标实现方法

2.3.1 实现方案分析

本自动测量系统主要分为三个部分：波形采集、RS232 通信、波形显示 MFC 编程。

第一部分：波形采集。了解信号发生器的单次触发原理，编写上位机程序控制其产生激励信号。

第二部分：RS232 编程。采用 RS232 串行接口的通信控件 SDI 程序设计，完成与信号发生器之间通信，完成程控应用程序编写。

第三部分：示波器功能编程。

综合各方面的因素考虑，该系统能够实现，并能满足课程设计的各项要求。

2.3.2 各部分实现

1.波形采集

根据函数发生器的程控方式，用 RS232 串口通信线连接，按程控命令格式发送指令控制示波器的输出信号波形。

2.RS232 编程

遵循 RS232 通信协议，起始位、8 位数据位、校验位，设置相应波特率（本系统采用 9600），实现数据和指令的传输。

3.波形显示

采用 MFC 单文档模式，使用 Static 静态文本编辑框作为波形画布，通过设置 butter 的消息响应函数完成波形显示的调整。

第三部分 核心硬件设计实现

3.1 关键部分性能分析

本自动测量系统设计主要涉及 RS232 通信控件的程序设计，根据数字函数信号发生器的程控命令格式要求，发送控制命令，对信号发生器进行信号的幅度、频率、占空比的调节。主要设计部分的原理和说明详见第二部分中 2.3.1 实现方案分析部分和 2.2.3 RS232 MFC 编程部分。

3.2 接口说明

3.2.1 RS232 接口

仪器可以选配 RS232 接口，符合 ELA-RS232 标准的规定，这是一种串行异步通信接口，它具有传输距离远、传输线少的特点，一般计算机上都带有这种接口。

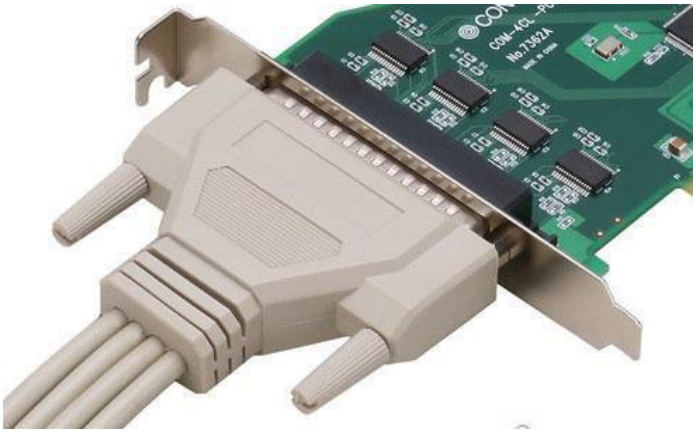


图 3.1 RS232 硬件接口

3.2.2 技术指标

1）接口电平

逻辑“0”:+5V~+15V；逻辑“1”：-5V--15V。

2）传输格式

传输信息的每一帧数据由 11 位组成:1 个起地位(逻辑 0) ，8 个数据位(ASCII 码)，1 个标志位(地址字节为逻辑 1，数据字节为逻辑 0)，1 个停止位(逻辑 1)。

3）传输速率

数据采用异步串行传输，默认传输速率为 19200bts/s.可由软件设定。

4）接口连接

将 RS232 传输电缆的一端插入仪器后面板上的 RS232 接口插座，另一端插入计算机上 COM1 或 COM2 插座。

5）系统组成

最多 99 台仪器连接电话的总长度不能期过 100 米。

6）适用范围

适用于一般电气干扰不太严量的实验宣成生产环值。

7）地址信息

仪器进入程控状态后，开始接受计算机发出的信息，根据标志位判断是地址信息还是数据信息。如果收到的是地址信息，判断是不是本机地址，如果不是本机地址，则不接者此后的任何数据信息，继续等待计算机发来的地址信息。如果判断是本机地址，对开始接收此后的数据信息，直到计算机发来下一个地址信息再重新进行判断。

8）数据信息

接收数据信息之后，进行判断并且存储，如果收到的字符是换行符 Chr(10)，则认为此次数据信息接收完毕，收器便开始逐条执行此次程控合令规定的操作。

3.2.3 数字信号发生器接口

ATF20B DDS 函数信号发生器提供 RS232 外接硬件接口，供程控模式使用。

图 3.2 数字信号发生器接口

3.3 被测系统搭建

3.3.1 多波形整体设计

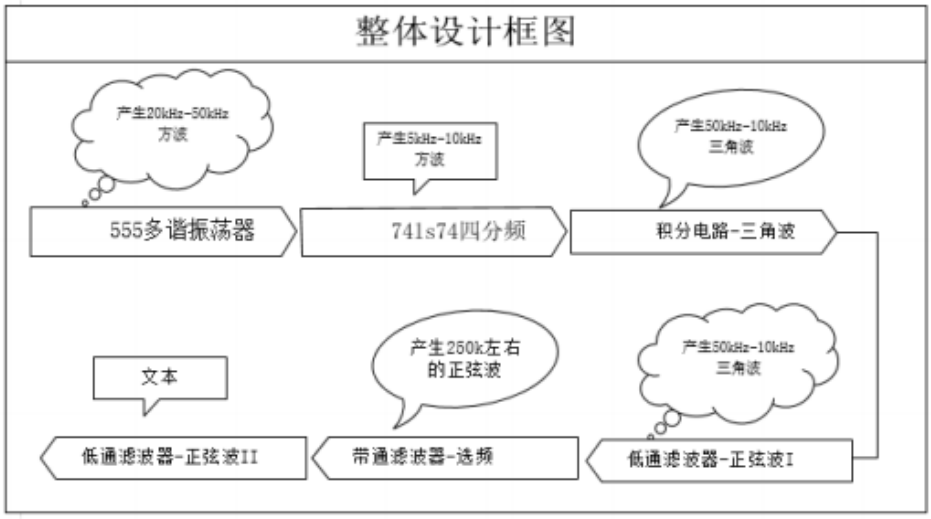


图 3.3 多波形产生器设计框图

3.3.2 单元电路设计

1. 555 多谐振荡器

由 555 定时器和外接元件 R1、R2 和 C 构成的多谐振荡器，2 脚与 6 脚直接相连，电路没有稳态，只有两个暂稳态，电路也不需要外加触发信号，利用电源通过 R1、R2 向电容 C 充电，使电路产生震荡，电容在 1/3VCC 和 2/3VCC 之间充电和放电，其电路原理图如图 3.4 所示。

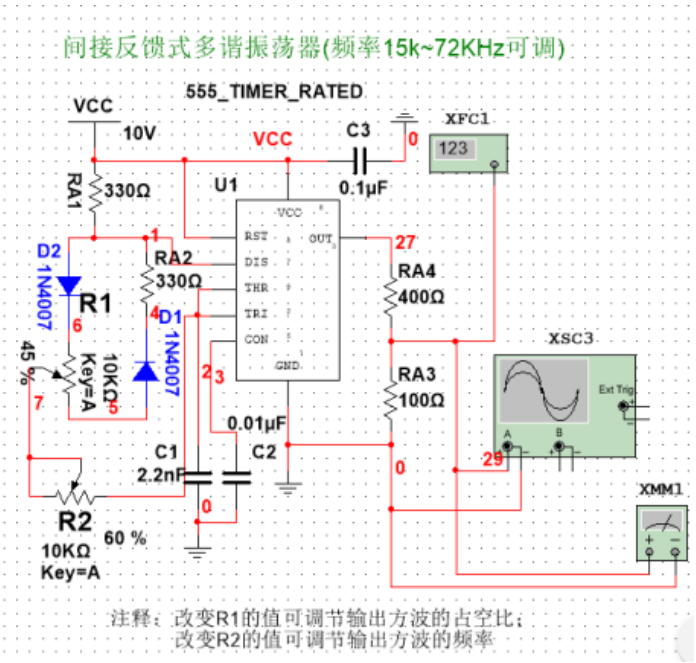


图 3.4 555 多谐振荡电路

2. 74LS74 分频电路

一个 74LS74 集成芯片有两个 D 触发器，一个 D 触发器可以组成一个二分频电路，把其中的一个 D 触发器的 Q 非输出端接到 D 输入端，时钟信号输入端 CLOCK 接时钟输入信号，这样每来一次CLOCK 脉冲，D 触发器的状态就会翻转一次，每两次CLOCK 脉冲就会使 D 触发器输出一个完整的正方波，这就实现了信号二分频。二分频电路输入信号过零上升沿每到来一次二分频器状态翻转一次便可得到二分频，把两个 D 触发器串联起来，就是四分频电路。于是基本方波信号就被分频成了 5kHz-10 kHz 的方波，然后经过分压电路，就得到 5kHz-10 kHz 的方波幅值为 1V 的方波 II。74LS74 四分频电路原理如图 3.5 所示。

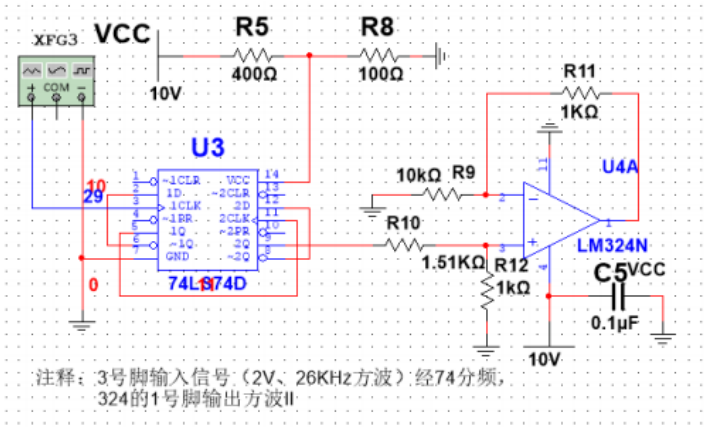


图 3.5 74HS74 分频电路

3.低通滤波器

低通滤波器由两节 RC 滤波电路和同相比例放大电路组成，其中同相比例放大电路具有输入阻抗高，输出阻抗低的特点。低通滤波器是容许低于截止频率的信号通过，但高于截止频率的信号不能通过的电子滤波装置。低通滤波器的作用是抑制高频信号，通过低频信号。简单理解，可认为是通低频、阻高频。低通滤波器包括有源低通滤波器和无源低通滤波器，无源低通滤波器通常由电阻、电容组成，也有采用电阻、电感和电容组成的。有源低通滤波器一般由电阻、电容及运算放大器构成，这里所用的是有缘低通滤波器。低通滤波器电路图如图 3-6。

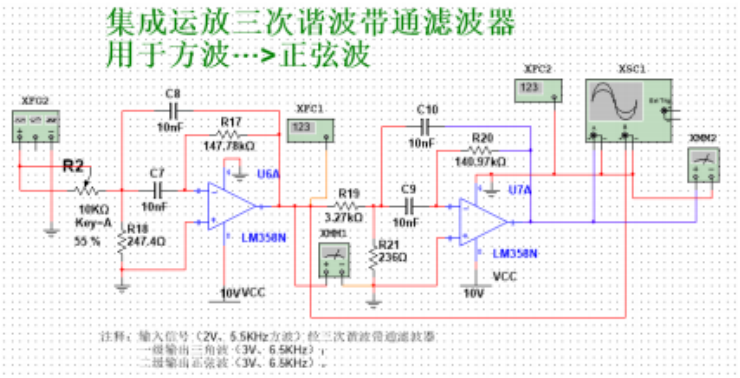


图 3.6 低通滤波器电路原理图

第四部分 系统软件设计分析

4.1 系统总体工作流程

应用软件的编制和调试，使用 Visual C++ 6.0 软件编程时，项目开发流程和其它软件开发项目的流程较为相似。

Windows 提供了通信控件 MSCOMM，编写基于对话框的 RS232 串行通信控件程序。

1） 建立项目

打开 VC++6.0，建立一个基于对话框的 MFC 应用程序 MSCommDlg；

2）在项目中插入 MSComm 控件；

3）利用 ClassWizard 定义 CMSComm 类作为对话框类的成员变量；

4）在对话框中添加控件

向主对话框中添加一个编辑框用于输入控制命令和功能命令，添加一个静态文本框作为接收显示反馈数据。

5）添加串口事件消息处理函数 OnComm()；

6）添加初始化代码和发送、接收响应函数；

4.2 程序设计思路

本构造自动测量系统将通过 MFC 上位机完成程控接口控制数字信号发生器输出激励信号，采用单文档方式显示响应信号，以达到课程设计的完成相关要求。信号发生器的程控接口，采用 RS232 硬件接口连接，根据程控命令格式与信号发生器通信，控制输出激励信号的幅度、频率等参数。同理，使用 RS232 与示波器相连，实现接口程序控制示波器工作。

通过 MFC 编写上位机显示采集到的信号波形，提供按键操作缩放、保存等功能。

4.3 示波器显示类

本构造自动测量系统将通过编写一个示波器动态数据显示类，完成响应信号显示、数据捕捉、波形调整（缩放、平移、网格）、颜色设置。

4.3.1 程序结构

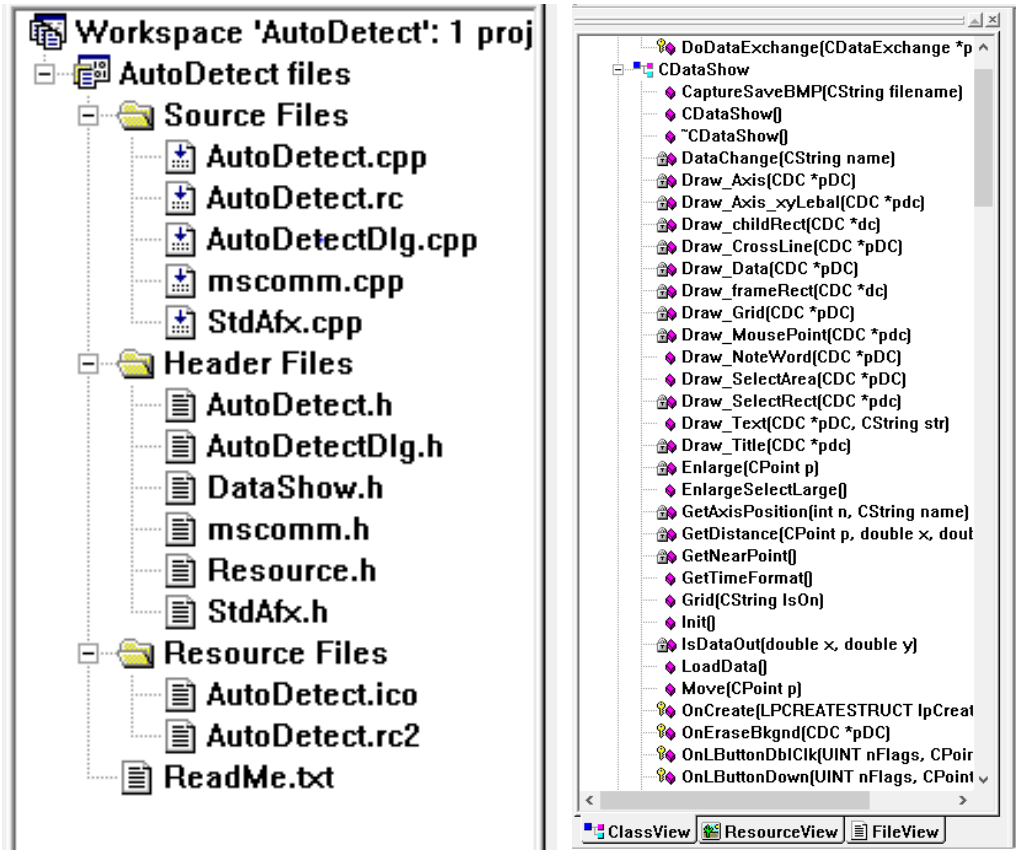


图 4. 2 数据显示类结构图

1.包含头文件 DataShow.h 和 DataShow.cpp 文件。

2.在对话框中加一个静态文本控件 CStatic，拖出一个适合大小的矩形。矩形不能太小，否则上面的文字将显得过大。

3.改变其 ID,比如将 IDC_STATIC 改为 IDC_SHOW,同时一定要把控件属性的“通知”勾选上，否则无法响应一些消息函数。

4.在 ClassWizard 中为静态控件 IDC_SHOW 生成一个成员变量 m_show.

5.找到定义 m_show 的地方，将 CStatic 改为 CDataShow.同时在相应的头文件上包含类的头文件：#include "DataShow.h"。

4.3.2 主要功能

1.动态显示数据

在类中定义了两个大小为 2000 的 double 型数值。可显示小于 2000 组的 X,Y 数据。动态显示的原理为实时更新数据。只要更新 x,y 数据，在显示界面就会及时显示。

2.数据捕捉功能

当鼠标靠近数据点时，十字线会绘制一个红色的矩形捕捉框。

3.支持曲线平移，缩放，网格开关，时间显示以及文字说明。平移时，按住右键拖动鼠标；缩放时按住鼠标中键拖动鼠标。

4.支持颜色设置，具体可看弹出菜单

双击鼠标右键，弹出菜单可进行相应的操作，如图 4.3 所示。

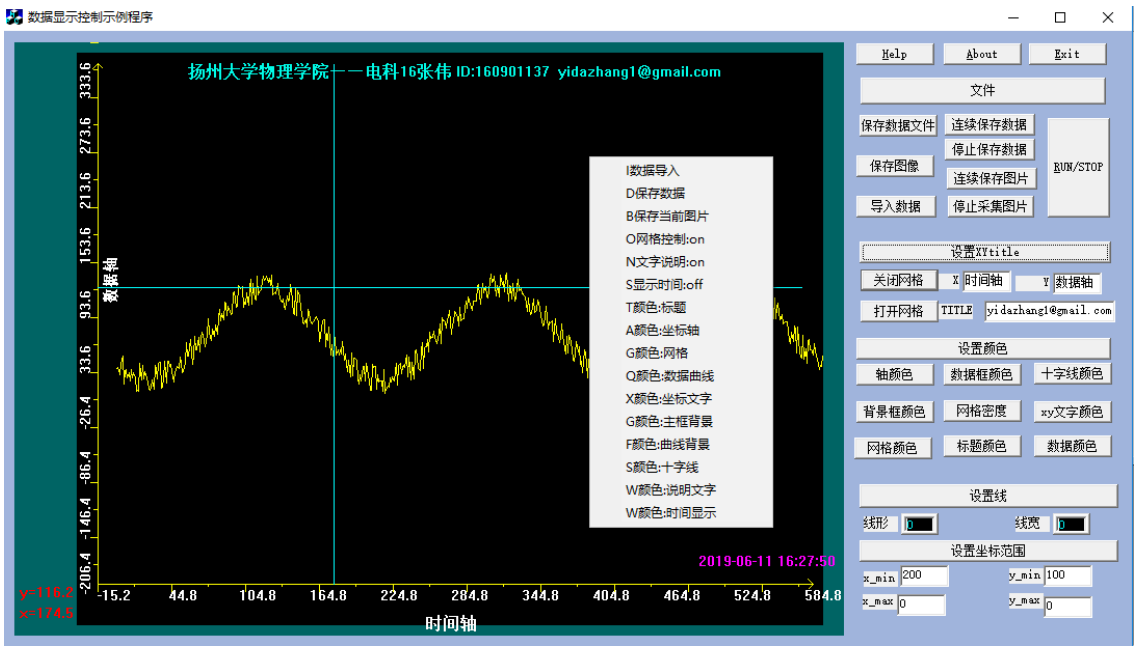


图 4. 3 鼠键操作示意图

5.类主要封装了一些设置函数和文件存储读取函数，我在具体样例程序中列举了一些函数。具体函数可见点 h 文件，上面有具体说明。

4.4 关键模块程序清单

4.4.1 信号发生器初始化

```
void CAutoDetectDlg::OnInit()
{
// TODO: Add your control notification handler code here
    if(m_Comm1.GetPortOpen())
        m_Comm1.SetPortOpen(FALSE);

    m_Comm1.SetCommPort(5);
    m_Comm1.SetInBufferSize(1024);
    m_Comm1.SetOutBufferSize(1024);
```

```
    m_Comm1.SetSettings("9600,n,8,1");
    m_Comm1.SetInputMode(1);
    m_Comm1.SetRThreshold(1);
    m_Comm1.SetInputLen(0);
    if( !m_Comm1.GetPortOpen())
        m_Comm1.SetPortOpen(TRUE);
    else
        AfxMessageBox("cannot open serial port");
    m_Comm1.GetInput();
}
//将示波器获取的 CString 转换为 double
double CAutoDetectDlg::CStringtodouble(CString str)
{
    double f;
    int au = (str[7]-48)*10+(str[8]-48);

    CString Strnum = str.Left(5);
    f = atof(Strnum);

    if(str[6]==45)
    {
        f = f/pow(10,au);
    }
    else
    {
        f = f*pow(10,au);
    }
    return f;
}
```

4.4.2 RS232 发指令

```
//通过串口发送 16 进制的起始位或终止位
void CAutoDetectDlg::SendHex(int t_flag)
{
    CByteArray hexdata;
    hexdata.RemoveAll();
    hexdata.SetSize(1);
    if(t_flag == 0)
        hexdata.SetAt(0,0x00);
    else
        hexdata.SetAt(0,0x01);
    m_Comm1.SetOutput(COleVariant(hexdata));
}
```

4.4.3 示波器初始化

```
void CAutoDetectDlg::OnSignalInit()
{
	// TODO: Add your control notification handler code here
	if(m_Comm1.GetPortOpen())
		m_Comm1.SetPortOpen(FALSE);
	m_Comm1.SetCommPort(7);
	m_Comm1.SetInBufferSize(1024);
	m_Comm1.SetOutBufferSize(1024);
	m_Comm1.SetSettings("9600,n,8,1");
	m_Comm1.SetInputMode(1);
	m_Comm1.SetRThreshold(1);
	m_Comm1.SetInputLen(0);
	if( !m_Comm1.GetPortOpen())
		m_Comm1.SetPortOpen(TRUE);
	else
		AfxMessageBox("cannot open serial port");
	m_Comm1.GetInput();
}
//-------------------------------------------------------
//设定信号发生器的参数
void CAutoDetectDlg::OnSignalSet()
{
	OnSignalInit();
	UpdateData(TRUE);
	flag=6;
	SendHex(1);
	m_Comm1.SetOutput(COleVariant("APPL:SIN "+m_s_freq+" KHZ,"+m_s_amp+" Vpp,0.0
V\n"));
	SendHex(0);
}
```

4.4.4 示波器显示程序

```
//画图函数，绘制坐标系
void CAutoDetectDlg::OnFresh()
{
	// TODO: Add your control notification handler code here
	CWnd *pwnd = GetDlgItem(IDC_STATIC_DRAW);
	CDC *pdc = pwnd->GetDC();
	pwnd->Invalidate();
	pwnd->UpdateWindow();
	pdc->Rectangle(0, 0, 380, 450);
```

```
    CPen *ppen = new CPen;
    ppen->CreatePen(PS_SOLID, 2, RGB(65,105,225));
    CGdiObject *pOldpen = pdc->SelectObject(ppen);

    pdc->MoveTo(30, 10);
    pdc->LineTo(30, 430);
    pdc->MoveTo(30, 10);
    pdc->LineTo(25, 25);
    pdc->MoveTo(30, 10);
    pdc->LineTo(35, 25);
    pdc->MoveTo(30, 430);
    pdc->LineTo(350, 430);
    pdc->LineTo(335, 425);
    pdc->MoveTo(350, 430);
    pdc->LineTo(335, 435);

    CFont font;
    VERIFY(font.CreatePointFont(100,"黑体",pdc));
    pdc->SelectObject(&font);
    pdc->TextOut(3,10,"AMP");
    pdc->TextOut(18,427,"0");
    pdc->TextOut(340,435,"Freq");

    for(int i=20;i<300;i+=10)
    {
            pdc->MoveTo(i+10,430);
            pdc->LineTo(i+10,433);
    }
    for(i=430;i>20;i-=10)
    {
        if((i/10%2)==1)
        {
            pdc->MoveTo(30,i);
            pdc->LineTo(25,i);
        }
        else
        {
            pdc->MoveTo(30,i);
            pdc->LineTo(28,i);
        }
    }
    pdc->TextOut(2,24,"400");
}
```

4.5 调试分析

4.5.1 总体说明

在调试时，由于子程序有很多，有时没法将每一个子模块都运行到，自然也无法及时发现其中的疏漏。为了解决这个问题，更好地查找错误，我解决的方法是将各个模块分别进行调试。例如在调试某一模块时，先将其他模块用“ //”暂时屏蔽掉，直到各个子模块都健康运行后，再整体调试，这样我们更加容易找出错误，增加效率。

另外，我印象最深的问题是，数据接收和数据发送总会出现问题，后来采用时分复用的方式解决此问题，相对同学使用线程的方法节省了系统资源，提高了处理效率。

经实际测试，本设计基本完成了设计要求，系统能够实现对信号发生器输出激励信号的频率、幅度等参数的设定，控制示波器的显示方式，能够较好的满足被测多波形发生电路的实际自动测量要求能功能。限于我实际水平的限制，书中难免有疏漏与不妥之处，诚恳地希望老师批评指正。

4.5.2 程控功能展示

图 4.4 程控功能示意图

从上面程控信号发生器展示图的结果，可以看出：

本自动测量系统能够较好的实现对信号发生器输出激励信号的幅度、频率、占空比等参数的调控，能偶获取信号的参数值。

4.5.3 示波器显示

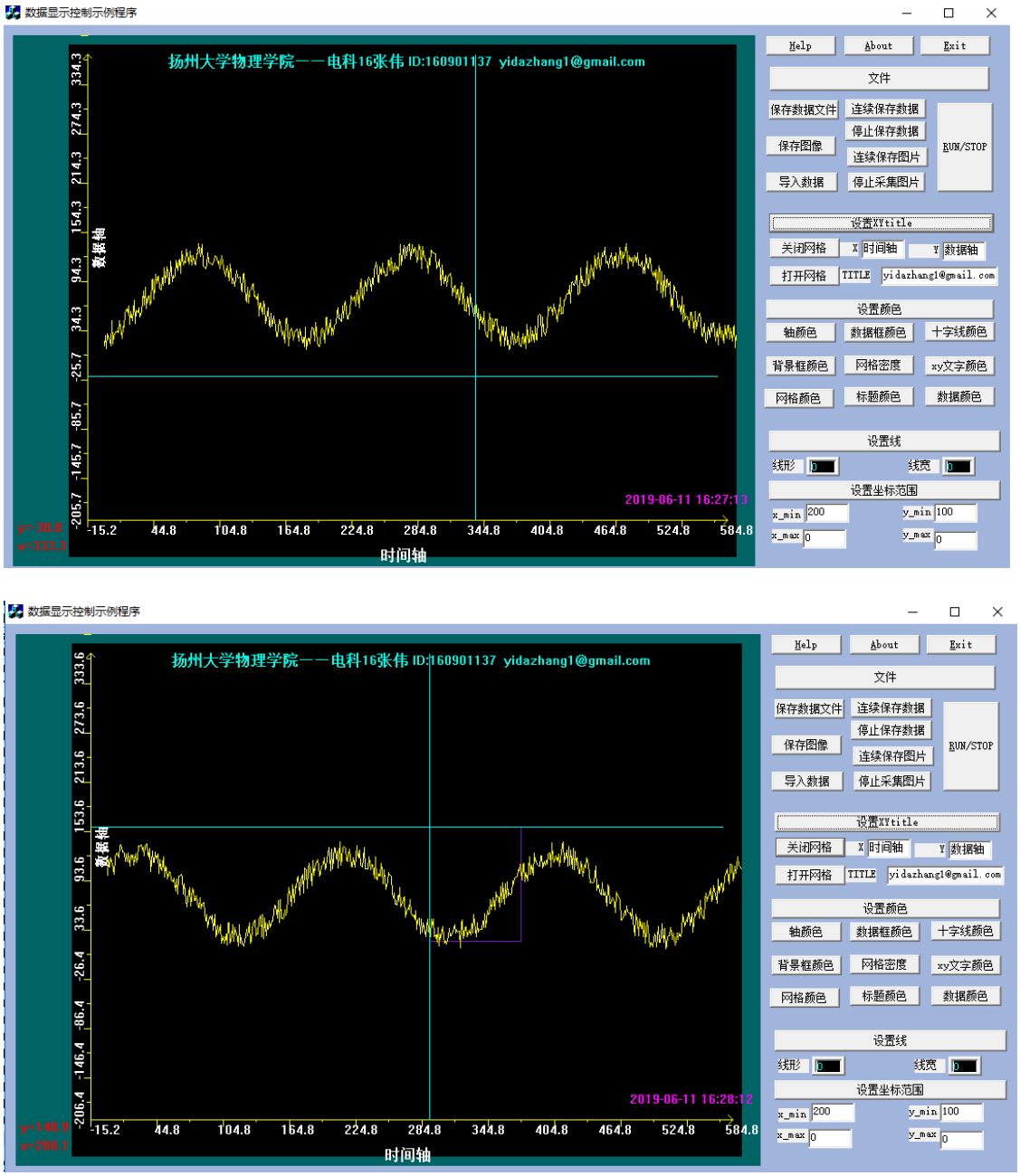


图 4.5 示波器功能展示图

根据自动测量系统的实际需要，采用 MFC 单文档编程方式实现响应信号显示。由图 4.5 所示，示波器上位机可以实现测量波形的显示，网格、参数测量、波形的保存、界面显示优化、波形局部缩放等功能。

第五部分 心得体会

通过本次课程设计，我从自动测量系统的设计与搭建中深深的体会到接口对于一个嵌入式系统来说是多么重要。接口可以说是一个系统的重要桥梁，在工作中协同软硬件按照指定的方案运行。在设计过程中 ，我们主要学习体会了单个模块的搭建与编程 ，例如信号发生器的初始化、示波器的初始化，线程接收子程序等。在这个系统搭建过程中，不但要将这些子模块有机的结合在一起 ，还要让它们较好的协调起来 ，按照我们思路运行，可以说是比较困难的。

由于我经验不足，接口协议的 MFC 编程是不能想当然的。我最容易犯的错误就是不经论证就去按照自己觉得可行的思路去进行，往往导致系统不能正常工作。虽然遇到了许多困难，但是在老师和同学的帮助下，我还是完成了这次的课程设计。通过本次课程设计， 我进一步了解了系统搭建的过程和系统软件编程的步骤，为今后的学习打下良好的基础。

在这里我要感谢我的指导老师张老师。老师工作很忙，但还是在我做课程设计的时间里一直关心我的进展。从设计方案的确定和修改，实现过程中遇到的困惑，及后来的详细设计等过程中都给了我很大的支持和关注。

本次课程设计让我把理论应用到了实践，同时通过课程设计，也加深了我对专业理论知识的理解和掌握。在解决问题的过程中，我查阅了大量专业书籍，获得了许多专业知识， 拓展了视野，提高了我的理论水平和实际的动手能力，并让我学会了解决问题的方法，激发了我的探索精神。这样的课程设计是很好的锻炼机会，课程设计使我深入的了解到了实践能力对于工科学生的重要性，增强了我们的实践动手能 力。

第六部分 附录

Ⅰ参考文献

[1] 仇谷烽，张京，曹黎明.基于 Visual C++的 MFC 编程.北京：清华大学出版社, 2015.01
[2] 揣锦华，袁琪. 面向对象程序设计与 VC++实践. 北京：清华大学出版社, 2016.02
[3] ATF20B DDS 函数信号发生器使用说明书, 泰克
[4] GDS-1000B Programming Manual_EN Rev 1.10 201511 安泰信

Ⅱ被测系统图

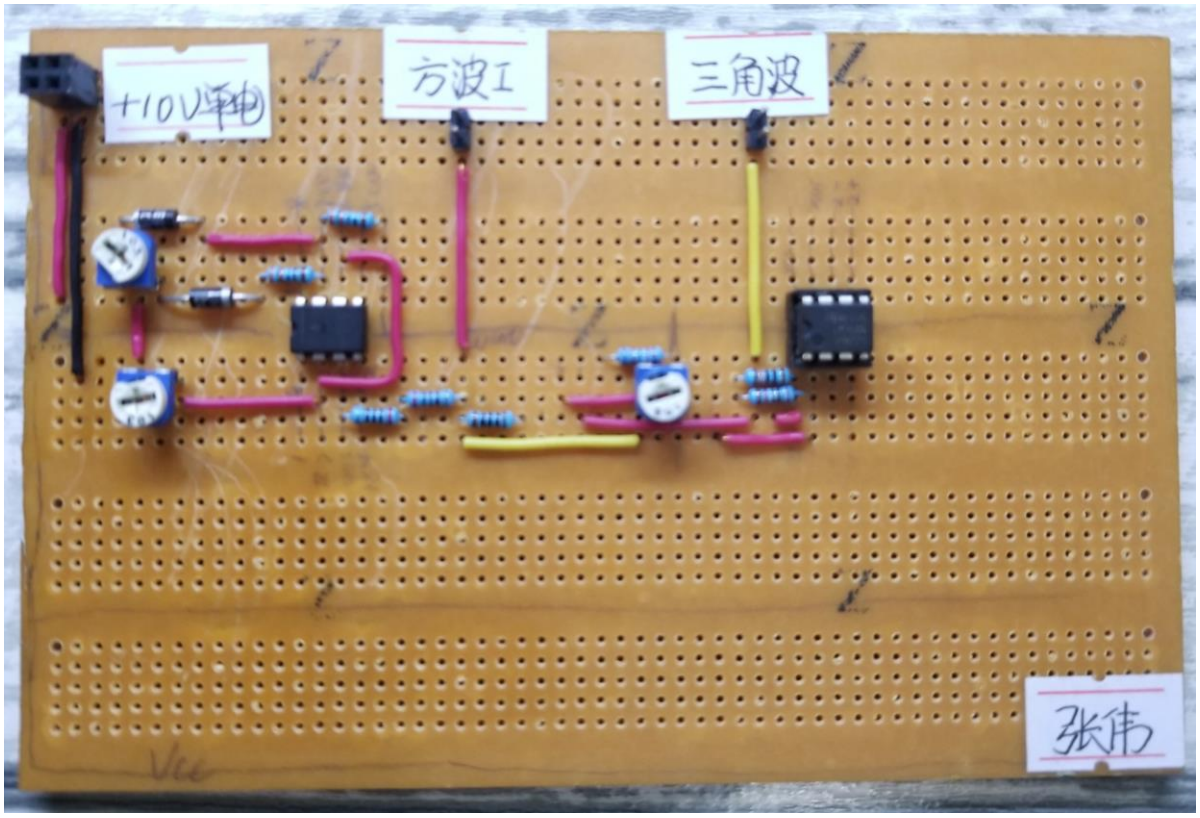


图 i-1 被测系统实物图

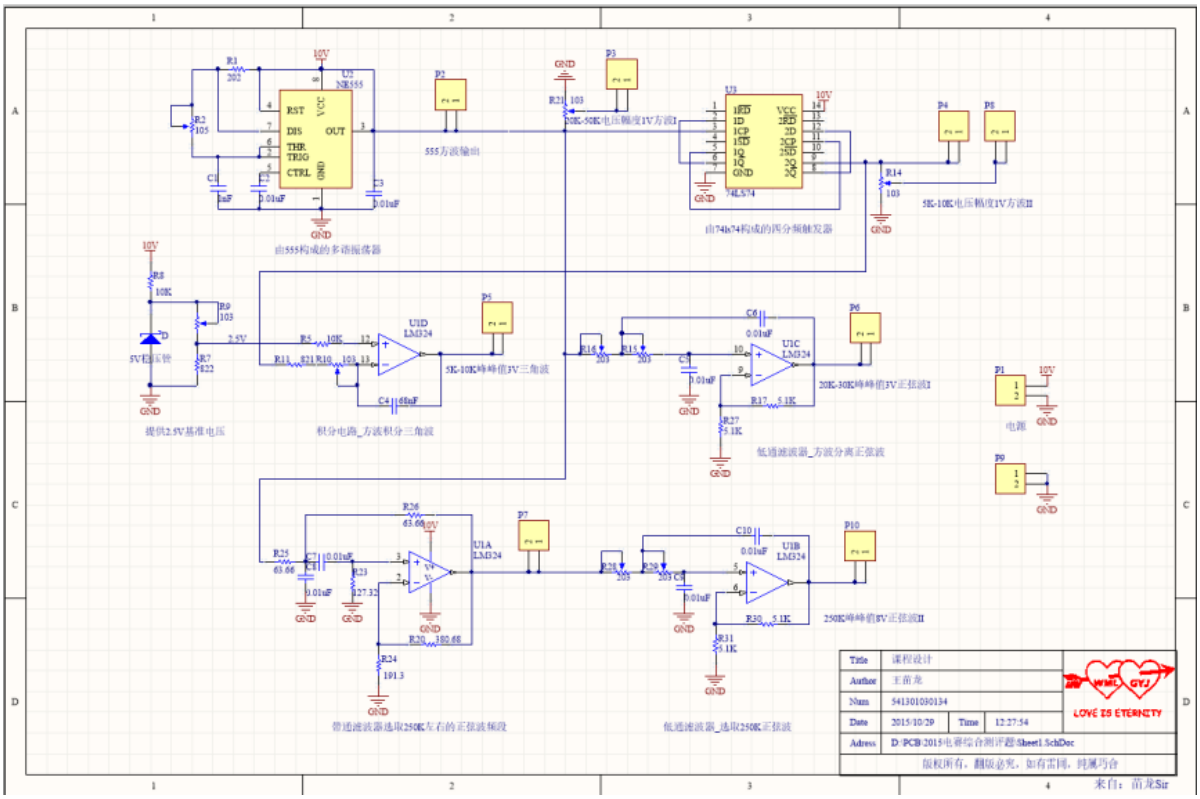


图 i-2 被测系统电路原理图

III 源代码

```
// AutoDetectDlg.cpp : implementation file
#include "stdafx.h"
#include "AutoDetect.h"
#include "AutoDetectDlg.h"
#include "math.h"
#include "DataShow.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
// CAboutDlg dialog used for App About
class CAboutDlg : public CDialog
{
public:
	CAboutDlg();

// Dialog Data
	//{{AFX_DATA(CAboutDlg)
	enum { IDD = IDD_ABOUTBOX };
	//}}AFX_DATA
```

```
	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CAboutDlg)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);	// DDX/DDV support
	//}}AFX_VIRTUAL

// Implementation
protected:
	//{{AFX_MSG(CAboutDlg)
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
	//{{AFX_DATA_INIT(CAboutDlg)
	//}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CAboutDlg)
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
	//{{AFX_MSG_MAP(CAboutDlg)
		// No message handlers
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()
// CAutoDetectDlg dialog
CAutoDetectDlg::CAutoDetectDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CAutoDetectDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CAutoDetectDlg)
	m_period = _T("");
	m_rise = _T("");
	m_mean = _T("");
	m_amp = _T("");
	m_s_amp = _T("");
	m_s_freq = _T("");
	m_s_wave = _T("");
	//}}AFX_DATA_INIT
```

```
    // Note that LoadIcon does not require a subsequent DestroyIcon in Win32
    m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CAutoDetectDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CAutoDetectDlg)
    DDX_Text(pDX, IDC_Det_PERIOD, m_period);
    DDX_Text(pDX, IDC_Det_RISE, m_rise);
    DDX_Text(pDX, IDC_Det_MEAN, m_mean);
    DDX_Text(pDX, IDC_Det_AMP, m_amp);
    DDX_Control(pDX, IDC_MSCOMM1, m_Comm1);
    DDX_Text(pDX, IDC_SIGNAL_AMP, m_s_amp);
    DDX_Text(pDX, IDC_SIGNAL_FREQ, m_s_freq);
    DDX_Text(pDX, IDC_SIGNAL_WAVE, m_s_wave);
    //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAutoDetectDlg, CDialog)
    //{{AFX_MSG_MAP(CAutoDetectDlg)
    ON_WM_SYSCOMMAND()
    ON_WM_PAINT()
    ON_WM_QUERYDRAGICON()
    ON_BN_CLICKED(IDC_GETPAR, OnGetpar)
    ON_BN_CLICKED(IDC_INIT, OnInit)
    ON_BN_CLICKED(IDC_Get_AMP, OnGetAMP)
    ON_BN_CLICKED(IDC_Get_MEAN, OnGetMEAN)
    ON_BN_CLICKED(IDC_Get_Period, OnGetPeriod)
    ON_BN_CLICKED(IDC_Get_RISE, OnGetRISE)
    ON_BN_CLICKED(IDC_SIGNAL_INIT, OnSignalInit)
    ON_BN_CLICKED(IDC_SIGNAL_SET, OnSignalSet)
    ON_BN_CLICKED(IDC_FRESH, OnFresh)
    ON_BN_CLICKED(IDC_AUTO, OnAuto)
    ON_BN_CLICKED(IDC_INITIAL, OnInitial)
    ON_BN_CLICKED(IDC_DRAW_PIC, OnDrawPic)
    ON_BN_CLICKED(IDC_INITIAL2, OnHelp)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CAutoDetectDlg message handlers
BOOL CAutoDetectDlg::OnInitDialog()
{
```

```
    CDialog::OnInitDialog();
    // Add "About..." menu item to system menu.
    // IDM_ABOUTBOX must be in the system command range.
    ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
    ASSERT(IDM_ABOUTBOX < 0xF000);

    CMenu* pSysMenu = GetSystemMenu(FALSE);
    if (pSysMenu != NULL)
    {
        CString strAboutMenu;
        strAboutMenu.LoadString(IDS_ABOUTBOX);
        if (!strAboutMenu.IsEmpty())
        {
            pSysMenu->AppendMenu(MF_SEPARATOR);
            pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
        }
    }

    // Set the icon for this dialog.  The framework does this automatically
    //  when the application's main window is not a dialog
    SetIcon(m_hIcon, TRUE);            // Set big icon
    SetIcon(m_hIcon, FALSE);           // Set small icon

    // TODO: Add extra initialization here
    return TRUE;  // return TRUE  unless you set the focus to a control
}

void CAutoDetectDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
    if ((nID & 0xFFF0) == IDM_ABOUTBOX)
    {
        CAboutDlg dlgAbout;
        dlgAbout.DoModal();
    }
    else
    {
        CDialog::OnSysCommand(nID, lParam);
    }
}

// If you add a minimize button to your dialog, you will need the code below
//  to draw the icon.  For MFC applications using the document/view model,
//  this is automatically done for you by the framework.
```

```
void CAutoDetectDlg::OnPaint()
{
	if (IsIconic())
	{
		CPaintDC dc(this); // device context for painting

		SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);

		// Center icon in client rectangle
		int cxIcon = GetSystemMetrics(SM_CXICON);
		int cyIcon = GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
		int x = (rect.Width() - cxIcon + 1) / 2;
		int y = (rect.Height() - cyIcon + 1) / 2;

		// Draw the icon
		dc.DrawIcon(x, y, m_hIcon);
	}
	else
	{
		CDialog::OnPaint();
	}
}

// The system calls this to obtain the cursor to display while the user drags
//  the minimized window.
HCURSOR CAutoDetectDlg::OnQueryDragIcon()
{
	return (HCURSOR) m_hIcon;
}

BEGIN_EVENTSINK_MAP(CAutoDetectDlg, CDialog)
	//{{AFX_EVENTSINK_MAP(CAutoDetectDlg)
	ON_EVENT(CAutoDetectDlg, IDC_MSCOMM1, 1 /* OnComm */, OnOnCommMscomm1,
VTS_NONE)
	//}}AFX_EVENTSINK_MAP
END_EVENTSINK_MAP()

int flag = 0;
void CAutoDetectDlg::OnOnCommMscomm1()
{
	// TODO: Add your control notification handler code here
	VARIANT variant_inp;
```

```
    COleSafeArray safearray_inp;
    LONG len,k;
    BYTE rxdata[2048];
    CString strtemp;
    CString m_temp = _T("");
    double temp = 0;
    if(m_Comm1.GetCommEvent()==2)
    {
        variant_inp=m_Comm1.GetInput();
        safearray_inp=variant_inp;
        len=safearray_inp.GetOneDimSize();
        for(k=0;k<len;k++)
            safearray_inp.GetElement(&k,rxdata+k);
        for(k=0;k<len;k++)
        {
            BYTE bt=*(char*)(rxdata+k);
            strtemp.Format("%c",bt);
            m_temp+=strtemp;
        }
        temp = CStringtodouble(m_temp);
        temp = temp*1000;
        m_temp.Format(_T("%.4f"),temp);
        switch(flag)
        {
        case 1:
            m_period+=m_temp;
            break;
        case 2:
            m_amp+=m_temp;
            break;
        case 3:
            m_mean+=m_temp;
            break;
        case 4:
            m_rise+=m_temp;
            break;
        default:
            break;
        }
    }
    UpdateData(FALSE);
}
//示波器串口初始化函数
void CAutoDetectDlg::OnInit()
```

```
{
// TODO: Add your control notification handler code here
	if(m_Comm1.GetPortOpen())
		m_Comm1.SetPortOpen(FALSE);

	m_Comm1.SetCommPort(5);
	m_Comm1.SetInBufferSize(1024);
	m_Comm1.SetOutBufferSize(1024);
	m_Comm1.SetSettings("9600,n,8,1");
	m_Comm1.SetInputMode(1);
	m_Comm1.SetRThreshold(1);
	m_Comm1.SetInputLen(0);
	if( !m_Comm1.GetPortOpen())
		m_Comm1.SetPortOpen(TRUE);
	else
		AfxMessageBox("cannot open serial port");
	m_Comm1.GetInput();
}
//将示波器获取的 CString 转换为 double
double CAutoDetectDlg::CStringtodouble(CString str)
{
	double f;
	int au = (str[7]-48)*10+(str[8]-48);

	CString Strnum = str.Left(5);
	f = atof(Strnum);

	if(str[6]==45)
	{
		f = f/pow(10,au);
	}
	else
	{
		f = f*pow(10,au);
	}
	return f;
}
//获取振幅参数
void CAutoDetectDlg::OnGetAMP()
{
	// TODO: Add your control notification handler code here
	m_amp	=_T("");
	UpdateData(FALSE);
	flag = 2;
```

```
    m_Comm1.SetOutput(COleVariant(":MEASure:AMPlitude?\n"));
}
//获取平均电压参数
void CAutoDetectDlg::OnGetMEAN()
{
    // TODO: Add your control notification handler code here
    m_mean    =_T("");
    UpdateData(FALSE);
    flag = 3;
    m_Comm1.SetOutput(COleVariant(":MEASure:MEAN?\n"));
}
//获取周期参数
void CAutoDetectDlg::OnGetPeriod()
{
    // TODO: Add your control notification handler code here
    m_period=_T("");
    UpdateData(FALSE);
    flag = 1;
    m_Comm1.SetOutput(COleVariant(":MEASure:PERiod?\n"));
}
//获取上升沿参数
void CAutoDetectDlg::OnGetRISE()
{
    // TODO: Add your control notification handler code here
    m_rise    =_T("");
    UpdateData(FALSE);
    flag = 4;
    m_Comm1.SetOutput(COleVariant(":MEASure:RISe?\n"));
}
//Auto
void CAutoDetectDlg::OnAuto()
{
    // TODO: Add your control notification handler code here
    OnInit();
    flag=6;
    m_Comm1.SetOutput(COleVariant("AUTOSet\n"));
    OnGetAMP();
}
//通过串口发送 16 进制的起始位或终止位
void CAutoDetectDlg::SendHex(int t_flag)
{
    CByteArray hexdata;
    hexdata.RemoveAll();
    hexdata.SetSize(1);
```

```
    if(t_flag == 0)
        hexdata.SetAt(0,0x00);
    else
        hexdata.SetAt(0,0x01);
    m_Comm1.SetOutput(COleVariant(hexdata));
}
//信号发生器初始化
void CAutoDetectDlg::OnSignalInit()
{
    // TODO: Add your control notification handler code here
    if(m_Comm1.GetPortOpen())
        m_Comm1.SetPortOpen(FALSE);
    m_Comm1.SetCommPort(7);
    m_Comm1.SetInBufferSize(1024);
    m_Comm1.SetOutBufferSize(1024);
    m_Comm1.SetSettings("9600,n,8,1");
    m_Comm1.SetInputMode(1);
    m_Comm1.SetRThreshold(1);
    m_Comm1.SetInputLen(0);
    if( !m_Comm1.GetPortOpen())
        m_Comm1.SetPortOpen(TRUE);
    else
        AfxMessageBox("cannot open serial port");
    m_Comm1.GetInput();
}
//设定信号发生器的参数
void CAutoDetectDlg::OnSignalSet()
{
    OnSignalInit();
    UpdateData(TRUE);
    flag=6;
    SendHex(1);
    m_Comm1.SetOutput(COleVariant("APPL:SIN "+m_s_freq+" KHZ,"+m_s_amp+" Vpp,0.0
V\n"));
    SendHex(0);
}
//画图函数，绘制坐标系
void CAutoDetectDlg::OnFresh()
{
    // TODO: Add your control notification handler code here
    CWnd *pwnd = GetDlgItem(IDC_STATIC_DRAW);
    CDC *pdc = pwnd->GetDC();
    pwnd->Invalidate();
    pwnd->UpdateWindow();
```

```
pdc->Rectangle(0, 0, 380, 450);

CPen *ppen = new CPen;
ppen->CreatePen(PS_SOLID, 2, RGB(65,105,225));
CGdiObject *pOldpen = pdc->SelectObject(ppen);

pdc->MoveTo(30, 10);
pdc->LineTo(30, 430);
pdc->MoveTo(30, 10);
pdc->LineTo(25, 25);
pdc->MoveTo(30, 10);
pdc->LineTo(35, 25);
pdc->MoveTo(30, 430);
pdc->LineTo(350, 430);
pdc->LineTo(335, 425);
pdc->MoveTo(350, 430);
pdc->LineTo(335, 435);

CFont font;
VERIFY(font.CreatePointFont(100,"黑体",pdc));
pdc->SelectObject(&font);
pdc->TextOut(3,10,"AMP");
pdc->TextOut(18,427,"0");
pdc->TextOut(340,435,"Freq");

for(int i=20;i<300;i+=10)
{
        pdc->MoveTo(i+10,430);
        pdc->LineTo(i+10,433);
}
for(i=430;i>20;i-=10)
{
    if((i/10%2)==1)
    {
        pdc->MoveTo(30,i);
        pdc->LineTo(25,i);
    }
    else
    {
        pdc->MoveTo(30,i);
        pdc->LineTo(28,i);
    }
}
pdc->TextOut(2,24,"400");
```

```
}

//初始化程序
void CAutoDetectDlg::OnInitial()
{
	// TODO: Add your control notification handler code here
	m_s_amp = _T("1.0");
	m_s_freq= _T("10");
	UpdateData(FALSE);
	OnSignalSet();
	OnAuto();
}

void CAutoDetectDlg::OnDrawPic()
{
	// TODO: Add your control notification handler code here
	CWnd *pwnd = GetDlgItem(IDC_STATIC_DRAW);
	CDC *pdc = pwnd->GetDC();
//	pwnd->Invalidate();
//	pwnd->UpdateWindow();
	CPen *ppen = new CPen;
	ppen->CreatePen(PS_SOLID, 2, RGB(255,69,0));
	CGdiObject *pOldpen = pdc->SelectObject(ppen);

//	char sMsg[200];
	int x = _ttoi(m_s_freq);
	int y = _ttoi(m_amp);
	y = 430-y;
	pdc->Ellipse(x+27,y-3,x+33,y+3);
	pdc->MoveTo(x+30,430);
	pdc->LineTo(x+30,y);
//	sprintf(sMsg,"%d",x);
//	AfxMessageBox(sMsg);
}
```