第十七届“冯如杯”学生课外学术科技作品竞赛项目论文

基于 AVR 和 PC 的一体化测控仪

项目编号　　

院（系）名称　宇航学院　

专业名称　　飞行器设计与工程（航天工程）

学　　号　　34151117

学生姓名　　段传辉

2007 年　**4** 月　**14** 日

基于 AVR 和 pc 的一体化测控仪

摘要

段传辉　34151117　　马　俊　34151112

王　楷　34151110　　邵昀明　34151114

本文系统的讲解一个由 pc 和 AVR 的 8 位单片机 mega16 为主体所搭建起来的测量和控制平台，构成一个实用仪器，以用来控制常用电动执行元件：直流电机、步进电机、舵机，测量常用物理量，如温度、频率、电压、电容，电感，转速，另外还可作为信号发生器产生可调频率的波形。本项目采用了 AVR 的 mega16 高速低功耗单片机。采用了电脑和单片机结合起来的策略，用电脑 RS232 串行口和单片机串口相连，电脑端用 VC 编写了界面友好的软件，可视化的窗口操作非常简便。下位机方面既用到了电脑上常用的 PS/2 键盘和 4×4 矩阵键盘作为输入设备，又用 16*2 字符液晶显示器作输出，另外，电路采用设计的印刷电路板搭接，避免了手工引线的杂乱和不可靠因素，工作稳定可靠。，本项目配有完备的使用说明和详细代码注释，既实用又有广阔的覆盖面。非常适合进行科技制作的同学，尤其是没有时间系统学习控制方法、电路设计存在困难、编程调试不通过的同学。此外，它也很适合用作教学仪器，演示测量原理、控制原理、通讯原理。总的来说，本产品质量可靠，功能强大，应用广泛。

关键词：MEGA16,串口，仪器，Visual C++

目录

基于 AVR 和 pc 的一体化测控仪

Chapter 1． 绪论

课外科技实践是大学学习生活的重要组成部分。然而，制作一个复杂的产品，尤其是机电综合产品，需要投入很大的精力，比如在电路设计、执行元件驱动控制、接口和程序的设计方面，要学习许多相关的知识，费时费力。本项目从科技实践的角度，为学生的课外科技制作和工科专业的教学量身定制了一套工具，软硬件俱全。目前大学生在机电控制系统的设计上，喜欢使用技术成熟、价格便宜的单片机。一块功能丰富的单片机，加上一些电动执行元件和一套传动设备，这就构成了我们产品的主要部分。现在的电子市场上，单片机的种类有很多，譬如：Intel 51 系列，ATEML 的 51 和 mega 系列，Motorola 公司 MCH68 系列等等，根据个人的使用习惯采用不同的系列。和 16 位 32 位相比，8 位机因实用价格低廉，至今仍是全球销量最大的机种。在测量和控制这些方面，是 8 位机的典型应用，ATMEL 公司生产的 AVR 系列，因具有 ISP 在线烧写，片内集成了 AD， PWM ，I2C 等常用电路，而且采用了哈佛总线，实现了单周期的指令执行，速度远超传统 51 系列。所以本项目采用了 AVR 的 mega16 高速低功耗单片机。

我们注意到，在单片机的开发应用上，市场里有一些单片机开发板、学习板出售，但大多价格不菲，也不一定适用于学生的学习和科技制作，很多的功能不实用，我们所需要的功能却没有，需要花很多的时间来自己开发和调试，这令许多入门者无所适从。本项目从控制直流电机，步进电机和舵机出发，让初学者快速掌握常用电动执行元件的控制，轻松的把此平台移植到自己的项目上作为控制源。另外，本产品还可测量电压、频率、温度、转速，做到了测量和控制兼顾，基本上可以满足课外科技制作的控制与测量需求，也提供给老师很好的教学演示教具，可以演示很多关于控制、传感、通讯和编程的实例。设计时采用了常用的输入输出方式，既有窗口操作软件，字符型液晶，又有普通的 PS/2 键盘，操作简便。模块化的设计让使用者根据需要进行拆装，把电路植入自己的机电产品，提高效率。我们衷心希望能够通过这个平台使同学们能够更好的学习、实践、掌握，投身更深入的科技实践中去。

Chapter 2. 硬件及原理

2.1 总体概念

在介绍具体原理之前，先认识一下我们的设计成果，了解其各个组成部分。

硬件组成：

舵机，步进电机（2 相或 4 相），直流电机（6~12V），　16*2 标准 LCD,PS/2 键盘，串口电缆，USB 转 RS232 转换器，mega16 单片机，DS18B20 温度传感器，L298N（2 片），红外发射和接收管，mega8 单片机，电阻，电容，开关若干

我们简要的介绍一下我们所用到的元件：

1 .mega16 简介

40 脚 DIP 封装，16KB FLASH，512Byte，EEPROM。片内集成 I2C,SPI,UART 总线控制器，8 路 10 位 AD，3 个定时器，3 路 PWM

2　ICL8038　Intersil 公司的函数发生器芯片，可以产生方波，正弦波和三角波，频率可调。

3　PS/2 键盘 PC 机用的普通键盘

4　16*2 标准字符液晶

采用日立公司 HD4478D 控制器的标准液晶模块，共 16 线。可用 4bit 或 8bit 数据线。本项目为节省 IO，采用 4bit 数据线接法

5　DS18B20 3 脚温度传感器

采用 DALASS 公司的 1-wire 总线，接口简单，直接用软件读出温度值，无须转换。

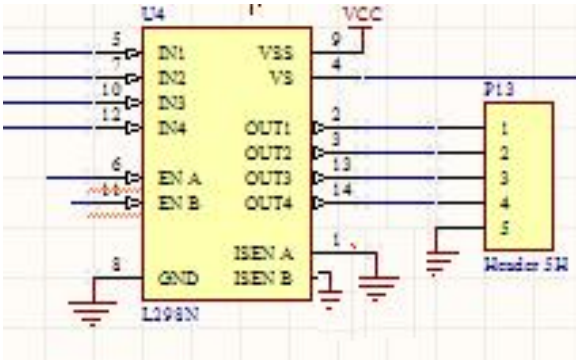
6 电压芯片 78 系列和 137 系列的稳压芯片 提供给仪器元件用，还可以对外输出。

2.2 电路原理

2.2.1 电动机部分原理

采用 L298N 功率驱动芯片，来驱动电动机。

L298N 是 ST 公司生产的 H 桥功率驱动驱动器，有两路输出，电压 5V~40V 均可工作，电流可达 2A。用四根输入线和两个使能端控制，分成两组，可以接两组线圈（可以改变电流方向），也可以接四个线圈（电流方向不变）。INPUT 1~4 用来改变线圈电流方向，



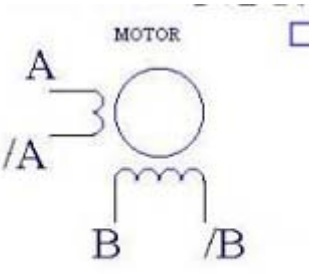
ENABLE 1 和 2 用来控制两组线圈是否工作。电路如左下所示。OUT1，OUT2 作为一组接电动机。电动机的速度通过 PWM 即脉冲宽度调制来调节，PWM 在电动机的控制中有广泛的应用。本项目通过单片机 MEGA16 的 PMM 模块在 OCR1A 和 OCR1B 引脚来产生占空比可以调节的矩形波，接到 L298N 的 EN A 和 B，周期性导通。占空比由键盘或软件发送给单片机，达到调速目的。具体实现过程可以参考我们的 PWM 源程序。

2.2.2 步进电动机的工作原理：

步进电机是可以精确控制转动角度和速度的电动机，这里选用了两相永磁步进电机，步距角是 1.8 度。一共四根线，电路和直流电机电路一样，把 L298N 的 OUT 1~4 接到步进电机 A，/A，B，/B 即可。通电时，采用两相四拍，每次通电两个线圈。时序是 AB（AB 表示 A 和 B 是高电压，下同），B /A，/A/B，A/B，这样不断循环，即可转动，每通电一次即可转动 1.8 度，改变通电延时时间即可改变速度。把以上通电顺序反过来即可以实现反方向转动。

2.2.3 舵机的控制原理：

舵机在控制系统有广泛的应用，它可以输出大力矩，通过输入周期 18~20MS，高电平宽 1~2MS 的脉冲即可控制角度，角度和高电平宽度成正比，宽度可在 0 到 180 度调节。这里采用 MEGA16 的 T imer2 定时器溢出中断产生脉冲，输出到 PORTB 的 PB0~4，一共可以控制 4 个舵机。命令由电脑端软件或者 PS/2 键盘给单片机。



步进电机接线示意图

2.2.4 频率测量：

频率测量一般有两种方法，其一是测量单位时间的脉冲个数，另一种直接测量一个脉冲的时间。本项目采用后者，这得益于强大的 MEGA16 的片内集成输入捕捉单元，把输入脉冲接到 ICP1 引脚，每当上升（或下降）沿时，进入中断程序，记下 Timer1 的计数值，下一次再记一个。两次差值即得出周期，取倒数即得频率。

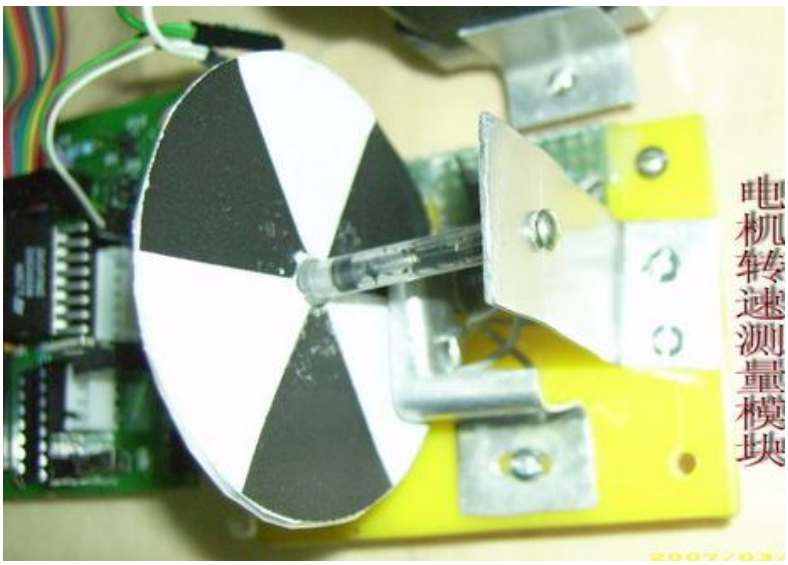
2.2.5 电压测量：

MEGA16 提供 8 路 10 位片内 AD 转换器，用户不必再接 AD 芯片。

本项目把电压引到 PA0 脚，用中断方式测量电压。一共两档，5V 档和 20V 档。MEGA16 支持直接测量 5V 以内的电压值，20V 超量程，但是可以用一个简单的分压电路来实现电压按比例下降。20V 是用电阻分压得出 0~5V 然后让单片机机测量。

2.2.6 转速测量：

本项目的实现使用自制的圆形光电编码盘，加上发射式红外发射、接收一体管。我们知道，黑色和白色的纸对红外线的反射率不同，接收管在没有接收到红外线时电阻约为几十千欧，有红外线时电阻只有几十欧，这是很大的电阻跳跃，自然我们想到把它接到电路中，把电阻的变化转化为电压的变化，跟踪电路中电压的变化，这样，随轴转动，就有脉冲输出，采用测量电压变化频率的方法，即可测出转速。



2.2.7 计数器：

几乎所有的单片机都有计数器/定时器，我们所选用的 MEGA16 也不例外。利用 MEGA16 的 T1 计数器，设定工作在外部脉冲计数模式，每来一个脉冲记数加一。这就实现了计数的过程。这是单片机最基本的应用之一。

2.2.8 信号发生器：

在电路制作，调试方面，信号发生器是最常用的仪器之一，但是通常的信号发生器

价格昂贵，远超一般学生承受能力，这里，我们用 ICL8038 做了一个输出正弦，方波，三角波的信号发生器，输出幅度可调，频率 1~15KHZ 可调信号，失真小，信号稳定，还通过单片机检测输出的频率，

2.2.9 逻辑分析仪：

逻辑分析仪是电路调试中的常用仪器，有了它，可以测出不同时刻某个点的电平（逻辑 1 或是 0），但是一个专业的逻辑分析仪，价格昂贵，远非学生可以承受，这里，我们用 MEGA16 的 PORTA 端口查询引脚输入的 8 路信号的逻辑，记录电平变化情况，时间值存入 MEGA 的 SRAM，最后发送到电脑显示出来，实现了简易的 8 路逻辑分析仪。

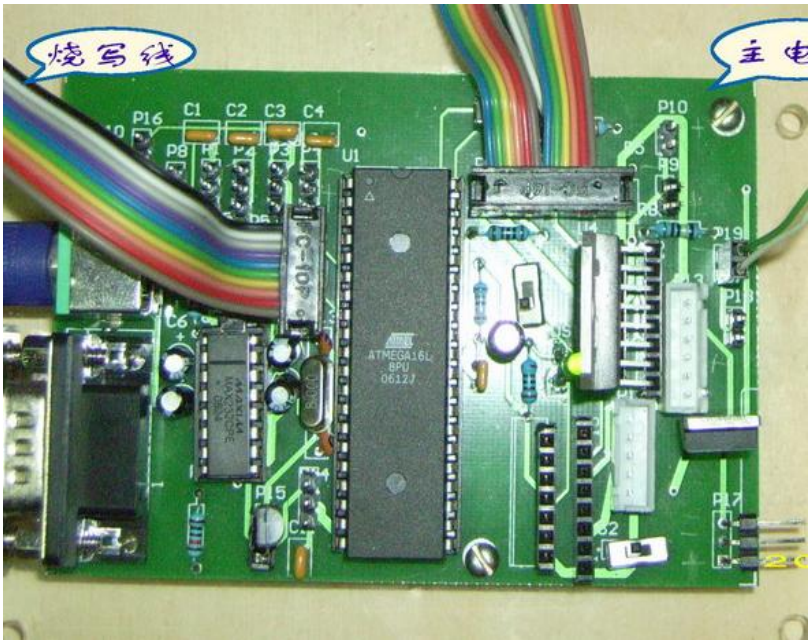
2.2.1 LC 测量

用 555 电路把 C 转化为方波信号的频率值，用电感三点式电路把电感转化为正弦波再由施密特触发整形得到方波，以上方波送到单片机，再由单片机运算出 L 和 C 值。

注：用到了两块 MEGA16，一个为主控，一个为 4X4 距阵键盘，逻辑分析仪 8 路输入

，两单片机用 SPI 通讯，主控单片机用串口连到电脑。

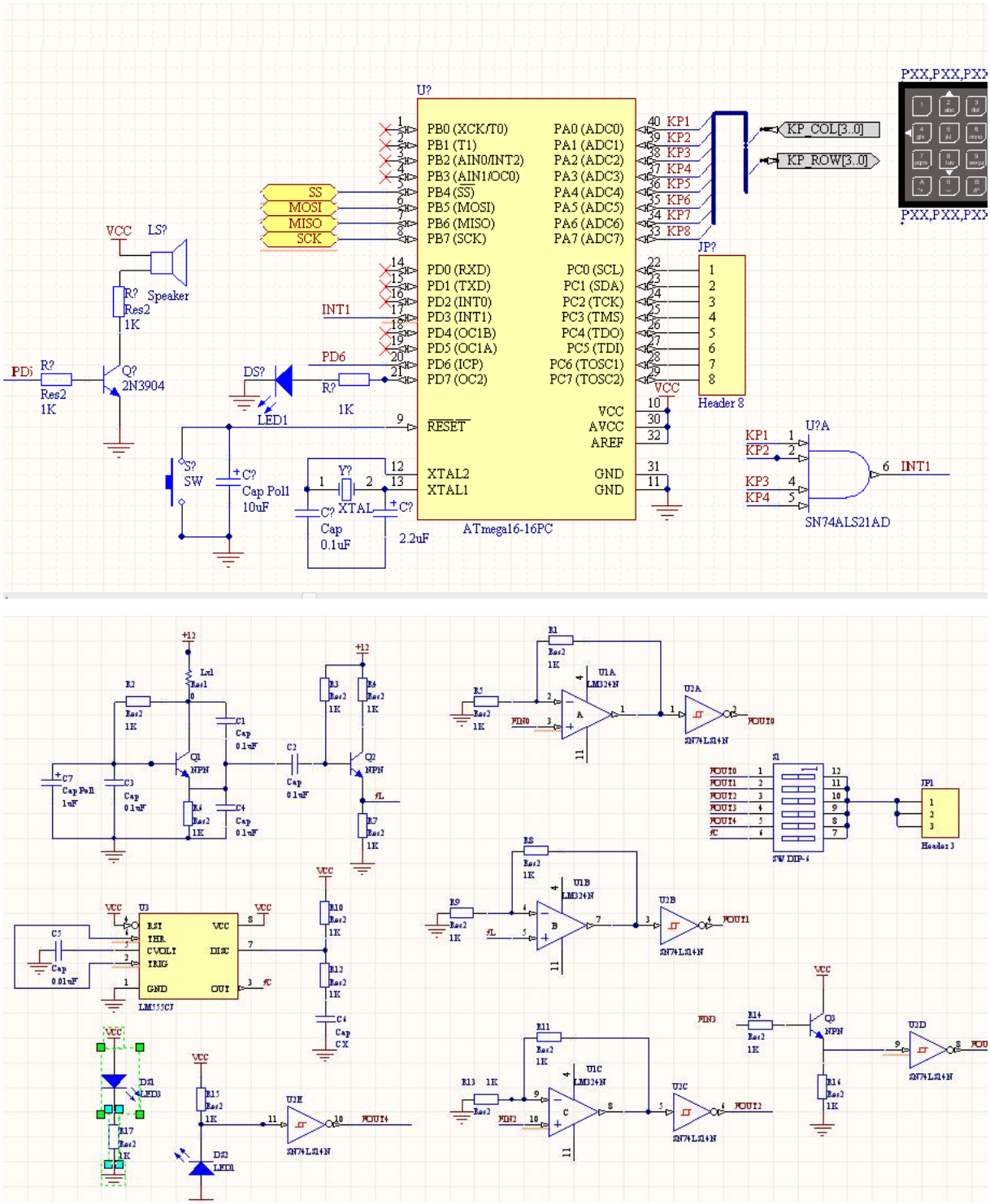
2.3　主芯片电路的组成



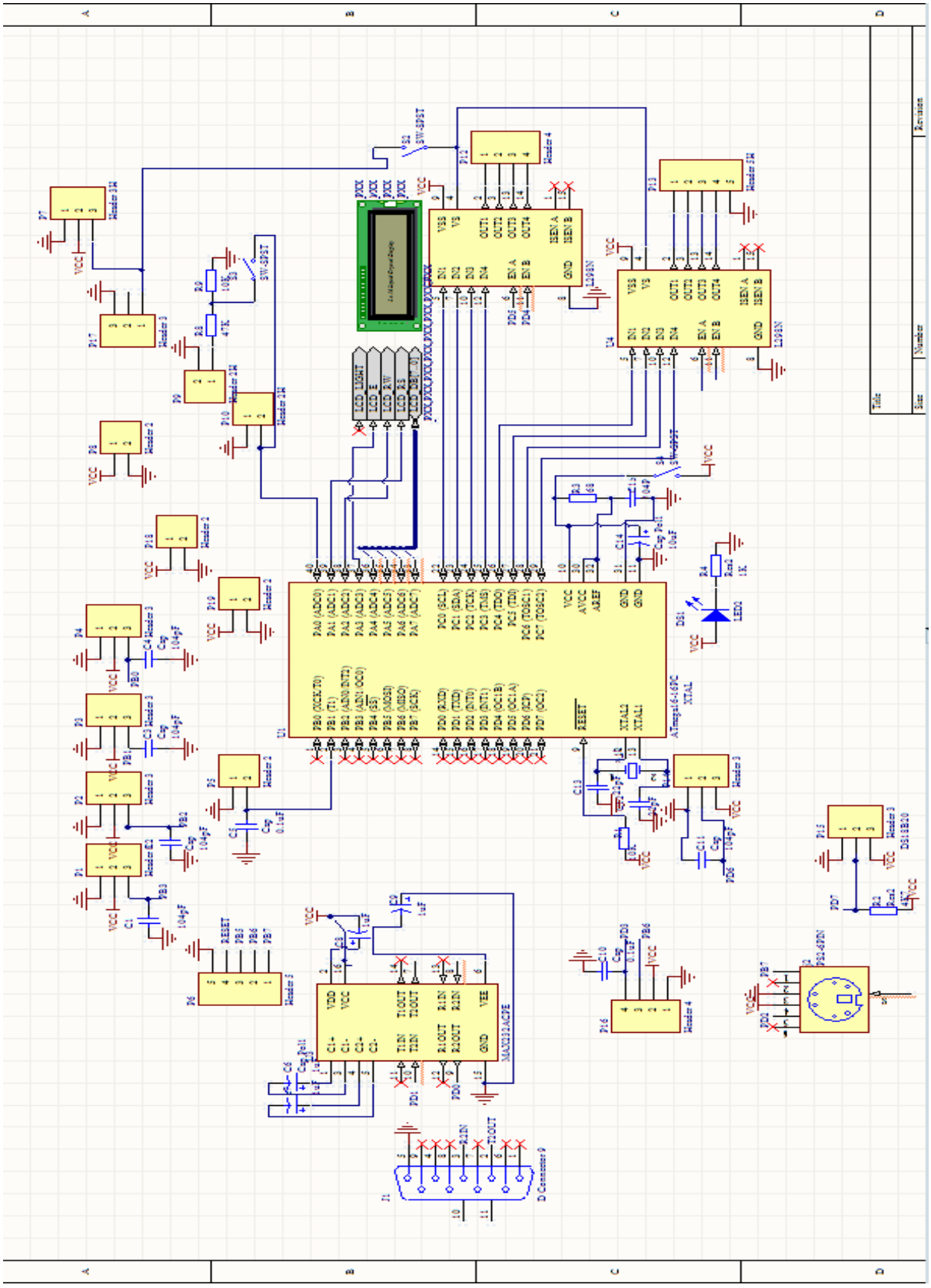
电路主控板的主要组成是：主单片机、驱动芯片、串口通信接口，PS/2键盘接口、矩阵键盘接口，液晶显示，信号发生器电路等。

下面是测量整形和键盘输入模块电路图：

。



外来频率要通过整形变成方波才能输入单片机 ICP 引脚。



2.4 通讯接口

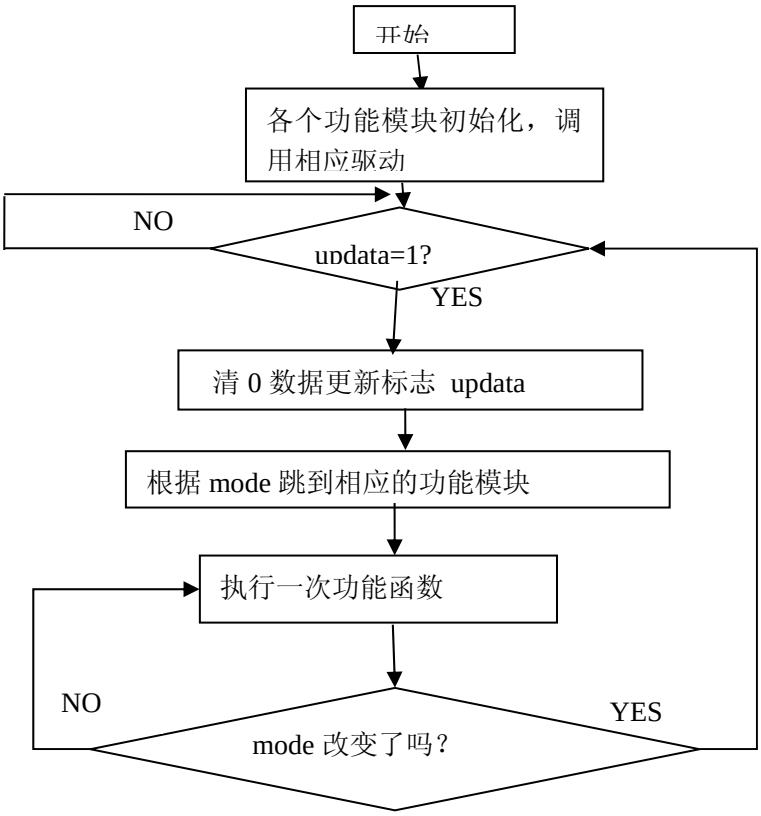
单片机的串口 TRD\RXD 通过 Max232（Maxim 公司生产的接口芯片），把 T T L 电平转换为 RS232 电平，然后接到电脑的九针串口上。完成单片机和电脑之间的全双工串行通讯。PC 端软件和单片机之间可以通过这个连接交换数据。另外，可以通过一个 PS/2 键盘的 DATA 线接到单片机的 PB7 上，CLK 时钟线接到单片机的 INT0（外部中断），单片机通过软件解码，识别按键，完成相应的动作。本项目的输出是 16*2 的标准 LCD 显示，通过单片机上的 PA2~PA7 口控制 LCD 的显示。

Chapter 3.软件

3.1 下位机的工作流程

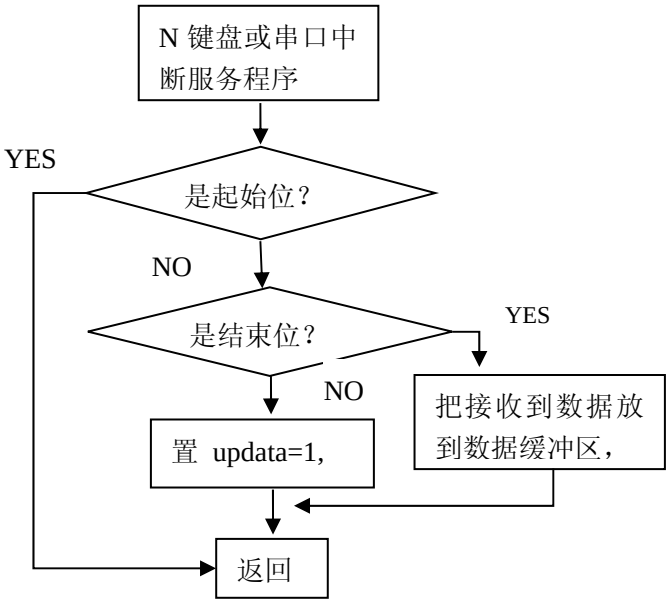
主函数程序流程图:

主函数是一个 死循环，不断的检测数据更新标志 update，当它为 1 时，表明有数据块传送到单片机，然后根据 mode 的值跳到相关功能模块，执行相应的功能，直到接收新数据。

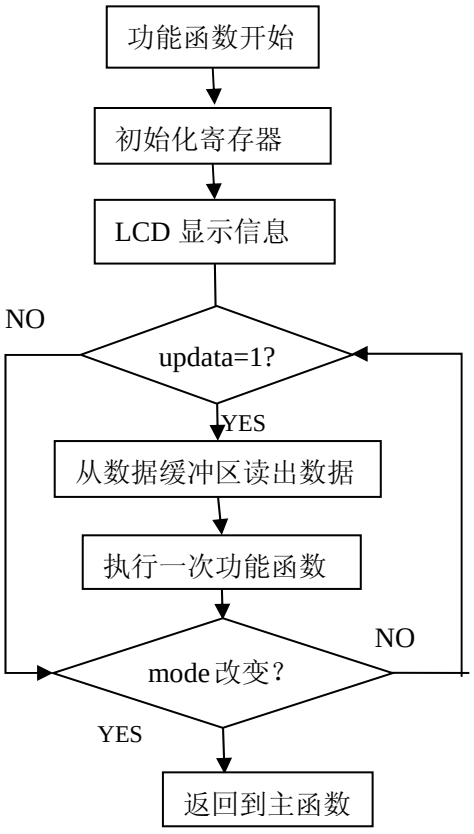


3.2 输入输出流程图:

输入有 PS/2 键盘输入，也有串口中断传来电脑端的输出，两个输入的功能是相似的，都是通过中断函数来接收数据，检测开始和结束标志，当检测到结束标志时置 update=1，否则把接收到



的数据存入数据缓冲区之中。接收完毕之后返回。



3.3　各个功能模块的流程图，

每一个功能模块也是一个循环，先初始化本功能所要用的寄存器，然后在 LCD 显示提示信息。从数据缓冲区中读入所要用到的数据，再把接收到的数据格式转化一下，作为参数传入函数，系统就开始执行相应功能。每执行一次就判断一次是否有数据更新，无则不再调用功能函数；再判断模式 mode 是否改变，若改变则返回主循环。

3.4 上位机软件介绍

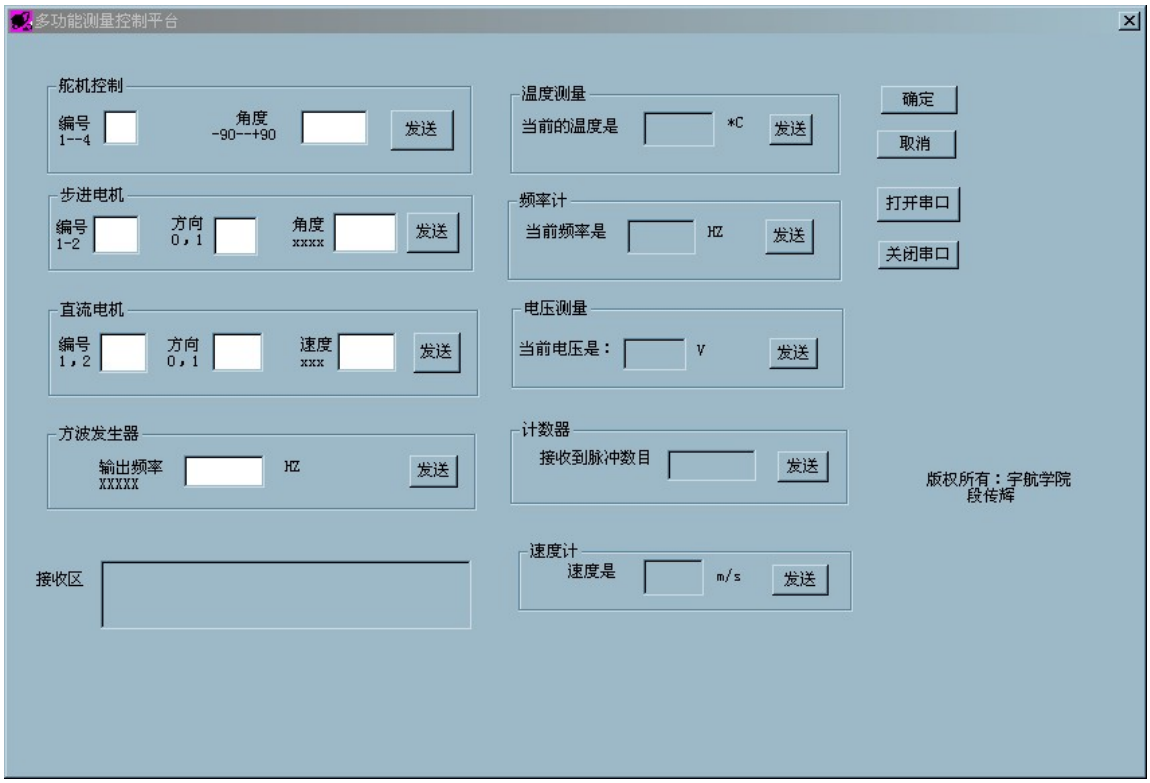
上位机也就是电脑终端的软件是用 Visual　C++6.0 编写的。

Visual　C++是微软公司开发的可视化 C++，特别适合编写 Windows 应用程序，它的界面友好，有很多代码是自动 返回到主函数 用按纽，编辑框等也很方便，本软件用 Visual　C++编写一个基于对话框的应用软件，在工程里导入 MSComm 控件，该控件是 Visual　C++6.0 自带的控件，封装了串口通讯所需要的类，处理串口通讯比较方便。这里选择基于事件驱动，

如果有按钮按下，发送消息，软件跳入相应函数，把编辑框的数据前面加上起始标志，结尾加上结束标志，（本软件以%开始，以@表示结尾）组成一个数据帧，以 ASCII 码格式从电脑 RS232 口发送给单片机。

一旦有数据从单片机传到电脑，就向系统发送消息，执行接收程序，根据所选功能，把接收的数据在适当的编辑框中显示出来。

软件的界面如下：



控件 ID	成员变量	变量类型	成员函数
IDC_DUOJI_NUM	m_duoji_num	CString	
IDC_DUOJI_J	m_duoji_j	CString	
IDC_DUOJI_SEND	----------	CString	OnDuojiSend()
IDC_STEP_NUM	m_step_num	CString	
IDC_STEP_DIRECTION	m_step_direction	CString	
IDC_STEP_J	m_step_j	CString	
IDC_STEP_SEND	----------	----------	OnStepSend()
IDC_MORTOR	m_mortor_direction	CString	

IDC_MORTOR_NUM	m_mortor_num	CString	
IDC_MORTOR_V	m_mortor_v	CString	
IDC_SEND_MORTOR	----------	----------	OnSendMortor()
IDC_FREQU	m_frequ	CString	
IDC_FREQU_SEND	----------	--------	OnFrequSend()
IDC_FREQUENCY	m_frequency		
IDC_FREQ_SEND	----------	--------	OnFreqSend()
IDC_TEMPERATURE	m_temperature	CString	
IDC_TEMPRATURE_SEND	----------		OnTenmperatureSend()
IDC_VOLTAGE		CString	
IDC_VOLTAGE_SEND	--------		OnVoltageSend()
IDC_MAICHONG	m_maichong_num	CString	
IDC_MAICH	---------	---------	OnMaiChSend()
IDC_V	m_v	CString	
IDC_SEND_V	----------	----------	OnSendV()
IDC_MSCOMM1	m_MSComm	CMSComm	
IDC_OPEN	----------		OnOpen()
IDC_CLOSE	----------		ONClose()

用到的控件和添加的变量，函数如下：

具体的函数编写见附录程序。

Chapter4.性能参数和使用说明

1．总体性能参数

输入电压　12V 直流

功能　1 控制 4 路舵机

2 最多控制 4 路直流电机（《=12V）,可调速双转向

3 最多可控制 2 路步进电动机（2 相或 4 相）

4 测量温度-20~110°C(误差+-0.5°C)

5 测量电压，两档，0~+5V,误差+-0.3V,0~20V,误差+-0.8V

6 测量频率，20~20000HZ，误差<2%

7 测量转速 600~60000r/min

8 计数器 最大脉冲数 60000 个

9 测量 L，C

10 输出电压 5V,10V,0~10V 三种

2．插上串口电缆，PS/2 键盘，以及舵机、步进电机、直流电机的插头（电路板上相应的排针）；

3．如果要测量频率、电压和转速或是计数，则要把相应的插头插到电路板上相应排针上；

4．插上电源，打开开关，LCD 显示“Select The Mode”,打开 PC 端软件，既可以在电脑端使用软件控制（填入相应的参数，点击发送即可），也可以拔掉串口，脱离电脑直接使用 PS/2 键盘或矩阵键盘控制。

致谢

在本项目的制造过程中，遇到了很多的困难，尤其是在程序的编写和硬件接口的设计方面，可谓是问题连连，幸好这里有许多受帮组的途径，首先当然是要感谢北航图书馆，北航出版社是全国嵌入式出版物的领军者，图书馆大量单片机和电子类参考书给另外我们极大的帮助，当然现代化的工具如网络也给了我们很多的帮助。比如 www.ouravr.com.cn 在这个网页里，我学到了很多的东西，对于这些帮助一并做出感谢。在制作电路板时，遇到了印刷电路板的一些问题，幸好有大四的师兄帮助，及时地完成了电路板的设计制作。同时我们还有一个出色的团队，在工作时彼此分工协作，加快了制作的进程，互帮互助，互相鼓励，就连我的父亲也在底板的加工中帮了我一把。这个项目并不能说是特别的复杂，但是凝聚了许多人的智慧，感谢他们提供的所有帮助。

参考文献

Atmel 《mega 单片机数据手册》

张军《AVR 单片机应用系统典型实例》 北京 中国电力出版社 2005

MgGraw 公司《机器人创意与制作系列丛书》北京 科学出版社 2003

《mega128/2560 高档单片机原理与应用》北京 希望电子出版社

www.ouravr.com.cn

www.sl.com.cn

附录 A. 源程序代码

为了方便使用者进行深入的学习、了解和改进，我们将程序的源代码公布如下，此代码仅用于学习交流，未经授权不可用于商业目的。

```
*                作者:段传辉                */
/*                2007 年 3 月                */
/*    目标 MCU:MEGA16     晶振:8MHZ    */
```

程序 P. 17　　总工程代码　　总工程.c

```
#include <iom16v.h>
#include"LCD_new.h"
#include"uart_m16_zhongduan.h"
#include"pwm.c"
#include"AD.c"
#include"ps2 键盘.c"
#include"舵机.c"
#include"步进电机.c"
#include"温度.c"
#include"方波发生器.c"
#include"频率测量.c"
#include"电动机.c"
#include"计数器.c"
#include"测速.c"
void Temprequere()
{unsigned char Date[]={0,0,0,0,'\0'};
//puts("温度测量准备完成\n");
 LCD_write_command(0x01);
if(LCD)
 LCD_write_string(1,0,"The temperature");
do{
      if(updata==1)
          {updata=0; Temper();}
   }
while(mode==1);
}

void Fangbo()
{   Fangbo_Init();
    //puts("方波工作模式，在 OCR1A 引脚输出方波");
    //puts("Give %2xxxxx@");
```

```
    LCD_write_command(0x01);
 if(LCD)
 LCD_write_string(0,0,"the frequency is");
    do
      Fangbo_Work();
  while(mode==2) ;
  // TCCR1A&=~(1<<COM1A0);//退出方波模式时停止 PWM，
 }
void Frequency()
{ Frequency_Init();
   SEI();
 // puts("测量频率");
   LCD_write_command(0x01);
   if(LCD)
 LCD_write_string(0,0,"the frequency is");
 do{
      Frequency_Work();
   }
    while(mode==3);
}
void Voltage()
{
AD_Init();
 SEI();
   LCD_write_command(0x01);
 if(LCD)
   LCD_write_string(0,0,"the Voltage is");
do AD();
while(mode==4);
}
void Duoji()
{
 LCD_write_command(0x01);
 if(LCD)
 {
 LCD_write_string(0,0,"duoji is wroking");
 LCD_write_string(0,1,"num :");
 }
duoji_init();
   do duoji_work();
   while(mode==5);
}
void Mortor()
```

```
{uint     pwm_wide;//脉宽
uchar mortor_num;
DDRC=0xff;
pwm_init();
 LCD_write_command(0x01);
if(LCD)
LCD_write_string(0,0,"mortor wroking");
SEI();
do
  {     if(updata==1)
                {    updata=0;
                 mortor_num=uart_buffer[1]-48;
                 pwm_wide=(uart_buffer[3]-48)*100+(uart_buffer[4]-48)*10
                 +uart_buffer[5]-48;
                 Mortor_Work(mortor_num,uart_buffer[2]-48,pwm_wide);
                  if(LCD)
                 {
                 LCD_write_string(0,1,"num:");
                 LCD_write_char(5,1,uart_buffer[1]);
                 if(uart_buffer[2]=='1')
                           LCD_write_char(7,1,'+');
                    else     LCD_write_char(7,1,'-');
                 LCD_write_char(8,1,uart_buffer[3]);
                 LCD_write_char(9,1,uart_buffer[4]);
                 LCD_write_char(10,1,uart_buffer[5]);
                 LCD_write_string(11,1,"/1000") ;
                //以下测频率
                    }
            }
    }
while(mode==6);
TCCR1A&=~(1<<COM1A1)|(1<<COM1B1);//退出时去掉 PWM
}
void Steportor()
{uchar direction,step_num;
uint num; // 角度
Init_Step();
 LCD_write_command(0x01);
if(LCD)
      {
      LCD_write_string(0,0,"stepmortor wrok");
  LCD_write_string(2,1,"num:");
  }
do{
```

```
      if(updata==1)
          {updata=0;
        step_num=uart_buffer[1]-48;//步进电机编号
        direction=uart_buffer[2]-48;//方向 0 逆 1 顺
           num=(uart_buffer[3]-48)*1000+(uart_buffer[4]-48)*100+
        (uart_buffer[5]-48)*10+uart_buffer[6]-48;//角度
           Step( step_num,direction,num);//步进电机运行
        }
   }
while(mode==7);
    }
 void T1jishu()
{uint num_old=0,num_new=0;
 uchar N[6]={'0','0','0','0','0','\0'} ;
   LCD_write_command(0x01);
   if(LCD)
     {
     LCD_write_string(0,0,"jishuqi wrok");
   LCD_write_string(2,1,"num:");
   }
while(updata==0)       {;}
T1jishu_Iint();
       do {num_new=TCNT1;
        if(num_old!=num_new)
              {num_old=num_new;
               Int_to_Char_5(num_old,N);
               puts(N);
               if(LCD) LCD_write_string(8,1,N);
               Delay_nms(20);
              }
        }
    while(mode==8) ;
   }
 void Cesu()
{uint v=0;
 uchar N[4]={'0','0','0','\0'} ;
    LCD_write_command(0x01);
 if(LCD)
     {
     LCD_write_string(0,0,"su du ji wrok");
  LCD_write_string(2,1,"sudu:");
   }
while(updata==0)     {;    }
       do {v=Cesu_work();
```

```
            Delay_nms(100);
            Int_to_Char(v,N);
            puts(N);
            if(LCD)
              {
                LCD_write_string(7,1,N);
                LCD_write_string(11,1,"r/min");
              }
           }
       while(mode==9) ;
   }
void main(void)
        { SEI();
          uart_init();
          LCD_Init();
          ps2_init();
       /* puts("基于 PC 和单片机的测控平台");
        puts("版权所有：宇航学院段传辉");
        putchar(0x0d);
          puts("请选择功能：1 温度测量  2 方波发生器 3 频率计  4 电压表 ") ;
        puts("5 舵机控制  6 电动机控制  7 步进电机控制")          ;
        puts("%MXXXXX@");*/
        LCD_write_string(0,0,"Select the mode");
         while(1)
         if(updata==1)
         {
           switch(mode)
                {case 0:break;
                 case 1:Temprequere();break;
                 case 2:Fangbo();   break;
                 case 3:Frequency();   break;
                 case 4:Voltage() ;   break;
                 case 5:Duoji();     break;
                 case 6:Mortor();     break;
                    case 7:Steportor();break;
                 case 8:T1jishu();break;
                 case 9:Cesu();break;
                 default :break;
                }
           }
         }
```

P2. LCD 驱动 LCD_new.h

```
#define LCD_EN_PORT      PORTA     //以下 2 个要设为同一个口
#define LCD_EN_DDR       DDRA
#define LCD_RS_PORT      PORTA     //以下 2 个要设为同一个口
#define LCD_RS_DDR       DDRA
#define LCD_DATA_PORT    PORTA     //以下 3 个要设为同一个口
#define LCD_DATA_DDR     DDRA      //一定要用高 4 位
#define LCD_DATA_PIN     PINA
#define LCD_RS           (1<<PA1) //0x02    portA1          out
#define LCD_EN           (1<<PA3) //0x08     portA3          out
#define LCD_DATA         ((1<<PA4)|(1<<PA5)|(1<<PA6)|(1<<PA7)) //0xf0     portA4/5/6/7 out
uchar   LCD   ; //是否让 LCD 显示     0，不显示;;;1 显示
/*-------------------------------------------------------------------------------------------
函数说明
-------------------------------------------------------------------------------------------*/
void LCD_init(void);
void LCD_en_write(void);
void LCD_write_command(unsigned   char command) ;
void LCD_write_data(unsigned char data);
void LCD_set_xy (unsigned char x, unsigned char y);
void LCD_write_string(unsigned char X,unsigned char Y,unsigned char *s);
void LCD_write_char(unsigned char X,unsigned char Y,unsigned char data);

void LCD_Init(void)             //液晶初始化
{
 LCD_DATA_DDR|=LCD_DATA;     //数据口方向为输出
  LCD_EN_DDR|=LCD_EN;          //设置 EN 方向为输出
  LCD_RS_DDR|=LCD_RS;          //设置 RS 方向为输出

  CLI();
 PORTA&=~0x04;//RW=0
 DDRA=0xff;
 LCD_write_command(0x30);
 Delay_nms(10);
 LCD_write_command(0x30);
 Delay_nms(5);
 LCD_write_command(0x30);
 Delay_nms(5);
  LCD_write_command(0x28);//写指令， 4 位数据线， 5 * 1 0 字体
  LCD_en_write();
```

```
    Delay_nms(4);
    LCD_write_command(0x28);   //4 位显示
    LCD_en_write();
    LCD_write_command(0x0c);   //显示开   光标关，字符不闪烁
    LCD_en_write();
    SEI();
    Delay_nms(2);
}

void LCD_en_write(void)   //液晶使能
{
    LCD_EN_PORT|=LCD_EN;      //
  Delay_nus(1);
    LCD_EN_PORT&=~LCD_EN;   //   给出下降沿
}

void LCD_write_command(unsigned char command) //写指令     E 下降，RS=0,RW=0
{
  Delay_nms(6);
    LCD_RS_PORT&=~LCD_RS;             //RS=0
    LCD_DATA_PORT&=0X0f;              //清高四位
    LCD_DATA_PORT|=command&0xf0; //写高四位
    LCD_en_write();                   //E 下降
    command=command<<4;               //低四位移到高四位
    LCD_DATA_PORT&=0x0f;              //清高四位
    LCD_DATA_PORT|=command&0xf0; //写这次低四位到数据线高 4 位
    LCD_en_write();

}

void LCD_write_data(unsigned char data) //写数据        E 下降，RS=1,RW=0
{
    Delay_nms(6);
    LCD_RS_PORT|=LCD_RS;            //RS=1
    LCD_DATA_PORT&=0X0f;            //清高四位
    LCD_DATA_PORT|=data&0xf0;   //写高四位
    LCD_en_write();
    data=data<<4;                   //低四位移到高四位
    LCD_DATA_PORT&=0X0f;              //清高四位
    LCD_DATA_PORT|=data&0xf0;     //写低四位
    LCD_en_write();
}
```

```
void LCD_set_xy( unsigned char x, unsigned char y )   //写地址函数
{
      unsigned char address;
      if (y == 0) address = 0x80 + x;   // 最高位是 1，另七位才是地址   故有+0x80
      else      address = 0xc0 + x;
      LCD_write_command( address);
}

void LCD_write_string(unsigned char X,unsigned char Y,unsigned char *s) //列 x=0~15,行 y=0,1
{
      LCD_set_xy( X, Y ); //写地址
      while (*s)   //  写显示字符
      {
         LCD_write_data( *s );
         s ++;
      }

}

void LCD_write_char(unsigned char X,unsigned char Y,unsigned char data) //列 x=0~15,行 y=0,1
{
   LCD_set_xy( X, Y ); //写地址
   LCD_write_data( data);

}
```

P2　　串口输入输出文件　　uart_m16_zhongduan.h

```
#define fosc 8000000 //晶振 8MHZ
#define baud 9600     //波特率

unsigned char uart_buffer[16]={'0'};//不算起始位'%'
unsigned char uart_buffer_p=0;
unsigned char Date[4]={0,0,0,0};
unsigned char updata=0;
uchar    mode=0;
void Int_to_Char(unsigned int i,char *p)
 {p[0]=i/1000+48;
   p[1]=(i/100)%10+48;
 p[2]=(i%100)/10+48;
 p[3]=(i%100)%10+48;
 }
```

```
/*              字符输出函数              */
void putchar(unsigned char c)
      {
       while (!(UCSRA&(1<<UDRE)));
       UDR=c;
      }
/*              字符输入函数              */
unsigned char getchar(void)
      {
       while(!(UCSRA& (1<<RXC)));
       return UDR;
      }
/*              字符串输出函数            */
int puts(char *s)
      {
      while (*s)
            {
            putchar(*s);
            s++;
            }
      putchar(0x0a);//回车换行
      putchar(0x0d);
      return 1;
      }
/*            不含回车换行的字符串输出函数          */
void putstr(char *s)
      {
      while (*s)
            {
            putchar(*s);
            s++;
            }
      }
/*              UART 初始化                   */
void uart_init(void)
{
 OSCCAL=0xc4;
 UCSRB=(1<<RXEN)|(1<<TXEN)|(1<<RXCIE);//允许发送和接收,及中断接收
 UBRRL=(fosc/16/(baud+1))%256;
 UBRRH=(fosc/16/(baud+1))/256;
 UCSRC=(1<<URSEL)|(1<<UCSZ1)|(1<<UCSZ0);//8 位数据+1 位 STOP 位
}
#pragma interrupt_handler uart_receive:12
void    uart_receive()
```

```
{uchar i;
 i=UDR;

if(i=='%')        return;//接收 bffer 从[1]开始，习惯
else if(i=='@') {uart_buffer_p=0;updata=1; return;}//接收指针清零，数据更新标志
else if(uart_buffer_p==0&&(i>='0'&&i<='9'))    {mode=i-48    ;uart_buffer_p++;}//接收 mode
   else    { uart_buffer[uart_buffer_p]=i; uart_buffer_p++;} //接收其他数据


}          //uart_buffer[1]开始数据， mode 独立
```

P3 脉冲宽度调节 PWM.c

```
void pwm_init(void)
{
OCR1A=512;//初始输出方波 50%占空比
OCR1B=512;
DDRD|=(1<<PORTD4)|(1<<PORTD5);//OCR1A，OCR1B 作输出
TCCR1A|=(1<<COM1A1)|(1<<COM1B1)|(1<<WGM11)|(1<<WGM10);//AB 皆比较匹配,
TCCR1B|=(1<<CS11);//相位可调 PWM,8 分频
}
```

P4 电压测量 AD.c

```
#define vref    508      //参考电压
unsigned int ad_con;//存放 PA0 的 AD 转换结果
void AD_Init(void)
{
DDRA&=~(1<<PA0);
MCUCR=0;
 ADMUX=0x00;//0 通道
 ADCSRA=(1<<ADEN)|(1<<ADSC)|(1<<ADIF)|(1<<ADIE)|0x07;
 //中断方式，64 分拼频
 SEI();
 }
void AD()
{ unsigned int temp;
 unsigned char i;
        {
    temp=(unsigned int)(((unsigned long)((unsigned long)ad_con*vref))/1024);
    temp-=9*(1024-ad_con)/1024;
      Int_to_Char(temp,Date);
       for(i=1;i<4;i++)
```

```
        {putchar(Date[i]);
            if(i==1) putchar('.');
         }
            if(LCD)
           {
             LCD_write_char(5,1,Date[0]);
            LCD_write_char(6,1,Date[1]);
           LCD_write_char(7,1,'.');
           LCD_write_char(8,1,Date[2]);
           LCD_write_char(9,1,Date[3]);
           LCD_write_char(11,1,'V');
           }

    Delay_nms(400);
      }
}
/*    AD 转换中断程序        */
#pragma interrupt_handler adc_isr:15
void adc_isr(void)
   {
     unsigned int temp=0;
     while(temp<6)
          temp++;
     ad_con=(uint)ADCL;
     temp=(uint)ADCH;
     ad_con=ad_con+(temp<<8);
     ADCSRA|=(1<<ADSC);//启动转换
   }
```

程序 P5 ps/2 键盘　ps2 键盘.c

```
#include "pincode.h"

#define BUFF_SIZE 64
#define PIN_KB    PINB
#define PORT_KB PORTD
#define CLOCK     2
#define DATAPIN 7
unsigned char bitcount;//PC 键盘数据长度计数
unsigned char kb_buffer[BUFF_SIZE];//键盘缓冲区
unsigned char input=0;//缓冲区读指针
unsigned char output=0;//缓冲区写指针
```

```
//送键盘按键 ASCII 码到键盘缓冲区

void put_kbbuff(unsigned char c)
 {
  kb_buffer[input]=c;
  if (input<(BUFF_SIZE-1))
      input++;
  else
      input=0;
 }
//从键盘缓冲区读取按键的 ASCII 码
unsigned char get_char(void)
  {
   unsigned char temp;
   if(output==input)
        return 0;
   else
       {
        temp=kb_buffer[output];
        if(output<(BUFF_SIZE-1))
          {
           output++;
          }
        else
          output=0;
        return temp;
       }
  }
  unsigned char getkey(void)
  {uchar key;
   while(1)
       { key=get_char();
         if(key!=0)    break;}
       return    key;
  }
//为运行读取 PC 键盘程序进行初始化
void ps2_init(void)
     {DDRD|=0x02;
       PORTD|=0xf0;
     DDRB&=~(1<<DATAPIN);
      MCUCR=0x02;//设置 8535 的 INT0 为下降沿触发中断
      GICR|=(1<<INT0);//使能 INT0 中断
      SEI(); //开中断
      bitcount = 11;
```

```
    }
//对 PC 键盘的扫描码进行解码
void decode(unsigned char sc)
    {
      static unsigned char shift,up,shiftup,start_flag=0;
      unsigned char i,tempkey;
      if (sc==0xf0)//按键释放
          {
           up=1;
           return;
          }
      if (up==1)//SHIF 键开关
          {
           up=0;
           if ((sc==0x12)|(sc==0x59)) shift=0;
           return;
          }
      switch (sc)
        {
          case 0x12:{//检测左 SHIF 键
                      shift=1;
                      shiftup=1;
                     }
          case 0x59:{//检测右 SHIF 键
                      shift=1;
                      shiftup=1;
                     }
          default:{
                   if (shift==0)
                    {
                      for(i = 0;unshifted[i][0]!=sc && unshifted[i][0]; i++);
                      if (unshifted[i][0] == sc)
                          { put_kbbuff(unshifted[i][1]);
                        tempkey=unshifted[i][1];//putchar(tempkey);
                         }
                    }
                   else
                      {
                      for(i = 0;shifted[i][0]!=sc && shifted[i][0]; i++);
                      if (shifted[i][0] == sc)
                          {
                            put_kbbuff(shifted[i][1]);
                          tempkey=shifted[i][1];    // putchar(tempkey);
                          }
```

```
                    }
                  }
          }
 if(tempkey==' ')    {start_flag=1;   return;}//接收 bffer 从[1]开始，习惯
else if(uart_buffer_p==0&&tempkey>='0'&&tempkey<='9'&&start_flag==1)
                  {mode=tempkey-48    ;      uart_buffer_p++;return;}//接收 mode
else if(tempkey==0x0d&&start_flag==1)
                 {uart_buffer_p=0;updata=1;start_flag=0; return;}
                          //接收指针清零，数据更新标志
  else { uart_buffer[uart_buffer_p]=tempkey; uart_buffer_p++;return;} //接收其他数据
}
#pragma interrupt_handler int0_isr:2//键盘数据读取程序

void int0_isr(void)
   {
    static unsigned char data;
    switch (bitcount)
      {
       case 11:{
                 if ((PIN_KB&(1<<DATAPIN))!=0)
                       return;          //START 位 ，不要送 DATA
                 else
                      bitcount--;
                 break;}
       case 2:{                    //奇偶校验，不管
                 bitcount--;
                 break;}
       case 1:{                    //停止位
                 bitcount--;
                 if ((PIN_KB&(1<<DATAPIN))==0)
                      {
                       bitcount=11;
                       return;
                     }
                 else
                     {
                      bitcount=11;
                      decode(data);//STOP 位正确，确认数据有效并解码
                     }
                 break;}
       default:{   data = (data >> 1);
                 if((PIN_KB&(1<<DATAPIN))!=0)
                      data|=0x80;
                 bitcount--;
```

```
                    //处理 8 位数据
                }
            }
        }
```

程序 P6 舵机代码：

```
unsigned char width=90;
unsigned char duoji_num=0;
unsigned int   count2=0;

void duoji_init()
{
 DDRB|=0x0f;//低 4 位接舵机
 PORTB|=0x0f;
 TCCR2|=(1<<CS20);//不分频，普通溢出模式
TCNT2=156;    //100 个数
SREG|=0x80;
TIMSK|=0x40;//开 T2 溢出中断

}
void duoji_work()
{unsigned char width_temp;
uchar a[4]={'1','+','0','0'};
  if(updata==1)
      {//for(i=1;i<7;putchar(uart_buffer[i]),i++)
         updata=0;
         a[2]=uart_buffer[2];
         if((a[2]=='+')||(a[2]=='-'))
          { a[3]=uart_buffer[3];      //角度十位
            a[4]=uart_buffer[4];       //角度个位
            duoji_num=uart_buffer[1]-49; //舵机编号
            width_temp=(a[3]-48)*10+(a[4]-48);
            if((width_temp>=0&&width_temp<91)&&(a[2]=='+'))   width=90-width_temp*2/3;
            else if((width_temp>=0&&width_temp<91)&&(a[2]=='-'))   width=90+width_temp*2/3;
            if(LCD)
            {

            LCD_write_char(7,1,uart_buffer[1]);
            LCD_write_char(11,1,a[2]);
            LCD_write_char(11,1,a[3]);
            LCD_write_char(11,1,a[4]);
            }
         }
```

```
        }
}
```

程序 P7 步进电机代码； 步进电机.c

```
#define  E1  PORTD=0X10          //上边作 E1
#define  E2  PORTD=0x20        //
void Init_Step(void)
{DDRC=0xff;
 DDRD|=(1<<PORTD5)||(1<<PORTD4);
}
void Step(uchar stepnum ,uchar direction,uint num)
{uchar q;
uchar m=0;//1~4 细分
num=num*10/18;//角度/1.8 得步数
if(LCD)
  {    LCD_write_char(8,1,stepnum+48);
       if(direction)
                LCD_write_char(11,1,'+');
       else     LCD_write_char(11,1,'-');
       for(q=0;q<4;q++)
       LCD_write_char(12+q,1,uart_buffer[q+3]);
  }
 if(direction==0&&stepnum==1)
              for(;num>0;num--)
             switch(m)
              {case 0:PORTC=0b00001001;
                        E2;E1;
                      Delay_nms(3);
                      m=1;     break;
                case 1: PORTC=0b00001010;
                        E1; E2       ;
                      Delay_nms(3);
                      m=2;     break;
                case 2:PORTC=0b00000110;
                      E2;E1;
                      Delay_nms(3);
                      m=3;     break;
                case 3: PORTC=0b00000101;
                        E1;E2;
                        Delay_nms(3);
                      m=0;     break;
              }
else if(direction==1&&stepnum==1)
              for(;num>0;num--)
```

```
                switch(m)
                 {
                 case 0:PORTC=0b00000101;
                            E2;E1;
                           Delay_nms(3);
                           m=1;     break;
                  case 1:PORTC=0b00000110;
                            E1; E2       ;
                           Delay_nms(3);
                           m=2;     break;
                  case 2:PORTC=0b00001010;
                           E2;E1;
                           Delay_nms(3);
                           m=3;     break;
                  case 3: PORTC=0b00001001;
                            E1;E2;
                            Delay_nms(3);
                           m=0;     break;

                  }
else if(direction==0&&stepnum==2)
                 for(;num>0;num--)
                switch(m)
                 {case 0:PORTC=0b10010000;
                            E2;E1;
                           Delay_nms(3);
                           m=1;     break;
                  case 1: PORTC=0b10100000;
                            E1; E2       ;
                           Delay_nms(3);
                           m=2;     break;
                  case 2:PORTC=0b01100000;
                           E2;E1;
                           Delay_nms(3);
                           m=3;     break;
                  case 3: PORTC=0b01010000;
                            E1;E2;
                            Delay_nms(3);
                           m=0;     break;
                 }
      else if(direction==1&&stepnum==2)
           for(;num>0;num--)
           switch(m)
               {case 0:PORTC=0b01010000;
```

```
                    E2;E1;
                    Delay_nms(3);
                    m=1;    break;
            case 1:PORTC=0b01100000;
                    E1; E2     ;
                    Delay_nms(3);
                    m=2;    break;
            case 2:PORTC=0b10100000;
                    E2;E1;
                    Delay_nms(3);
                    m=3;    break;
            case 3: PORTC=0b10010000;
                    E1;E2;
                    Delay_nms(3);
                    m=0;    break;
            }
}
```

程序 P8　　温度测试代码：温度.c

```
//温度测量通过串口传到电脑;8MHZ
#define CL_DQ PORTD&=~(1<<PD7)
#define SET_DQ PORTD|=(1<<PD7)
#define SET_OUT DDRD|=(1<<PD7)
#define SET_IN   DDRD&=~(1<<PD7)
#define IN_DQ   PIND&(1<<PD7)

void write_1820(uchar x)
{   uchar m;
    CLI();
    for(m=0;m<8;m++)
    {
      CL_DQ;
       if(x&(1<<m))      //写数据，从低位开始
       {Delay_nus(2);SET_DQ; }
       else
      CL_DQ;
       Delay_nus(60);     //15~60us
      SET_DQ;
       Delay_nus(10);
    }
```

```
      SET_DQ;
      SEI();
}
uchar read_1820()
{     uchar temp,k,n;
      CLI();
   temp=0;
      for(n=0;n<8;n++)
          {
          CL_DQ;
        //Delay_nus(2);
          SET_DQ;
          //delay(3);
          SET_IN;
          Delay_nus(1);
          k=IN_DQ;       //读数据,从低位开始
          if(k)
          temp|=(1<<n);
          else
          temp&=~(1<<n);
          Delay_nus(70); //60~120us
          SET_OUT;
     }
     SEI();
     return (temp);
}
void init_1820(void)
 {CLI();
 SET_OUT;
   SET_DQ;//输出 1
   CL_DQ;
   Delay_nus(550);//拉低一段时间
   SET_DQ;//释放
   SET_IN;//输入
   Delay_nus(60);
   while(IN_DQ)   {;}//等待回复
   Delay_nus(240);//回复的低电平在 60 到 240US
   SET_OUT;
   SET_DQ;//回到初始 DQ=1；
   SEI();
 }

void Temper(void) //主函数
{
```

```
// uart_init();
//Delay_nus(2000);
puts("Ready!!!");
uchar Date[]={0,0,0,0,'\0'};
    while(1)
   {
      uint j,tem;
      uchar i,temh,teml;
      init_1820();  //复位 18b20
       write_1820(0xcc);    // 发出转换命令  搜索器件
       write_1820(0x44);       //启动
       for(j=1000;j>1;j--)
              Delay_nus(1000);//转换一次 12 位的  最多 750MS
        init_1820();
       write_1820(0xcc);  //发出读命令
       write_1820(0xbe);

       teml=read_1820();  //读数据
       temh=read_1820();
       tem=(teml>>4)+(temh<<4);
       Int_to_Char(tem,Date);
      // puts("现在的温度是：");
      puts(Date);
     // puts("℃          ");
      putchar(0x0a);

      //每次转换需要延时 200ms 以上
      for(j=2000;j>1;j--)
              Delay_nus(1000);
         }
  }
```

程序 P9　　方波发生器代码：方波发生器.c

```
void Fangbo_Init(void)
{
DDRD|=(1<<PD5);//PD5(OCR1A)作为方波输出
PORTD|=(1<<PD5);
TCCR1A=(1<<COM1A0);//A 比较匹配,
TCCR1B=(1<<CS10)|(1<<WGM12);//CTC PWM,不分频
OCR1A=4000;  //顶值  在 OCR1A   频率 4000000/OCR1A;
}
void Fangbo_Work(void)
```

```
{uchar      x;
uint f=0;   //频率 100---10K   HZ
 if(updata==1)
         {updata=0;
          f=(uart_buffer[1]-48)*10000+(uart_buffer[2]-48)*1000+
          (uart_buffer[3]-48)*100+(uart_buffer[4]-48)*10+(uart_buffer[5]-48);
         OCR1A=4000000/f;
         if(LCD)
          {for(x=1;x<6;x++) LCD_write_char(x+4,1,uart_buffer[x]);
          LCD_write_string(11,1,"HZ");
          }
          }
}
```

程序 P10　　频率测量代码 ：频率测量.c

```
unsigned int oldcount=0;//捕获时新旧 T1 值
unsigned int newcount=0;

void Frequency_Init()
  {
  DDRD&=~(1<<PD6);  //捕捉引脚  处为输入
  TIMSK|=0X20;//使能 ICP 捕捉中断;
  //TCCR0=0X02;//8 分频，溢出中断 T0
  TCCR1B=0XC2;//8 分频;捕捉最小单位 1us
  TCNT0=0;
  ICR1=0;
  TCNT1=0;
  }
void Frequency_Work()
     {uint freq=8888;
     uchar f[7]={'0','0','0','0','\0'};
     if((newcount&0xfff8)!=(oldcount&0xfff8))   //低三位不一样，频率
                  { oldcount=newcount;      //改变较大，要接受新值
                    freq=4000000/oldcount;   //count 每个数代表 1us
                    Int_to_Char(freq,f);
                    puts(f);
                    if(LCD)
                     {LCD_write_string(4,1,f);
                      LCD_write_string(9,1,"HZ");
                      }
                    Delay_nms(100);     //不必太快
                  }
         }
```

```
//捕捉中断处理程序
#pragma interrupt_handler Icp_timer1:6//ICP 中断入口地址
void Icp_timer1()
  {//putchar('a');
  newcount=ICR1;
  ICR1=0;
  TCNT1=0;
  TCCR1B=0XC2;//8 分频，噪声抑制，使能捕获
  }
```

程序 P11 电机控制代码 电动机.c

```
void Mortor_Work(mor_num,direction,v)    //PORTC 高四位输出，OCR1A OCR1B 输出 PWM
{          if(direction==0&&mor_num==1)   {PORTC=0b10000000;OCR1B=v;}
     else if(direction==1&&mor_num==1)    {PORTC=0b01000000;OCR1B=v;}
     else if(direction==0&&mor_num==2)    {PORTC=0b00100000;OCR1A=v;}
     else if(direction==1&&mor_num==2)    {PORTC=0b00010000;OCR1A=v;}
     else if(direction==0&&mor_num==3)    {PORTC=0b00001000;OCR1B=v;}
     else if(direction==1&&mor_num==3)    {PORTC=0b00000100;OCR1B=v;}
     else if(direction==0&&mor_num==4)    {PORTC=0b00000010;OCR1A=v;}
     else if(direction==1&&mor_num==4)    {PORTC=0b00000001;OCR1A=v;}

}
```

程序 P12 计数器代码：计数器.c

```
void T1jishu_Iint()
{TCCR1B=6;
 DDRB&=~(1<<PORTB1);
}
```

程序 P. 11 测转速：

```
void Cesu_Init()
{    uint freq=8888;
Frequency_Init();
     if((newcount&0xfff8)!=(oldcount&0xfff8))  //低三位不一样，频率
                  { oldcount=newcount;      //改变较大，要接受新值
                    freq=4000000/oldcount;  //count 每个数代表 1us
                v=freq/3; //光电编码盘一圈有 3 次黑白交替      }
                 return     v;
}
```