利用 P87LPC762 单片机设计绕线计数器的方法

摘要： 本文详细介绍了利用 P87LPC762 单片机设计绕线计数器(DEMO 板)的方法。

该计数器具有如下特点：

(1) 可预置捲绕圈数(0～9999)；

(2) 可控制机器按预置的圈数捲绕，达到目标圈数停止捲绕并刹车；

(3) 预置圈数和已绕圈数均可掉电保存；

(4) 能够识别和控制电机转动方向，正反转处理数据。

该计数器可用于工业控制，适用于测控转速不太高的绕线机。

一、硬件电路设计

(一) 系统结构分析

该计数器是以 LPC762 单片机为核心的应用系统。整个系统包括如下几部分：处理器 LPC762、存储器 E^2PROM(24WC02)、电机、传感器、显示器等以及相应的驱动。系统结构如图 1。

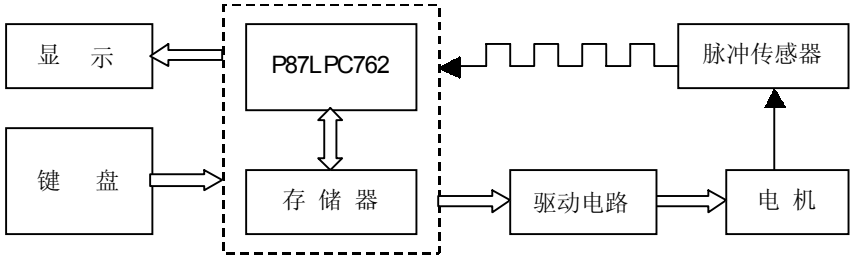

图 1　系统结构框图

(二) 电路原理分析设计

整个系统硬件大致可分为如下几部分：信号采集电路、信号处理部分、键盘显示以及电机驱动。现分别对这几部分作介绍。

1.　信号采集电路

这部分的主要器件是脉冲传感器，在设计中采用红外对管。电路图如图 2。

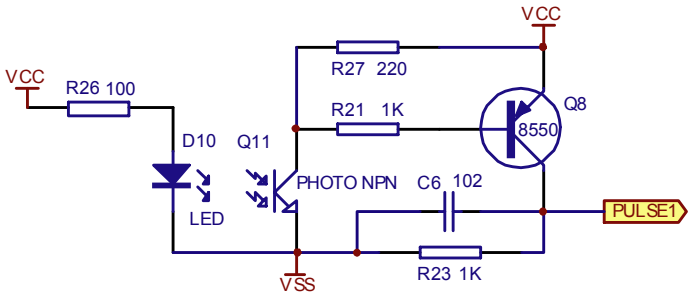

图 2　信号采集电路

为了识别转向，在实际电路中，有 2 路完全一样的信号采集电路，图 2 只是其中 1 路。其中电容 C6 起滤波作用，消除不稳定因素所带来的干扰。

红外对管实物连接如图 3 所示，在测试中得到的脉冲波形如图 4 所示。

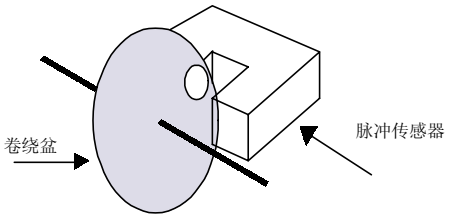

图 3　红外对管实物连接示意图

图 4　脉冲波形图

当电机上转盘的孔经过红外对管时，在 PULSE1 处得到高电平。

2．　信号处理部分

这部分是系统的核心部分，主要包括 CPU(P87LPC762)和存储器(24WC02)以及复位电路和外接晶振，CPU 和 E^2PROM 之间采用硬件 I^2C 总线(参见图 5)。

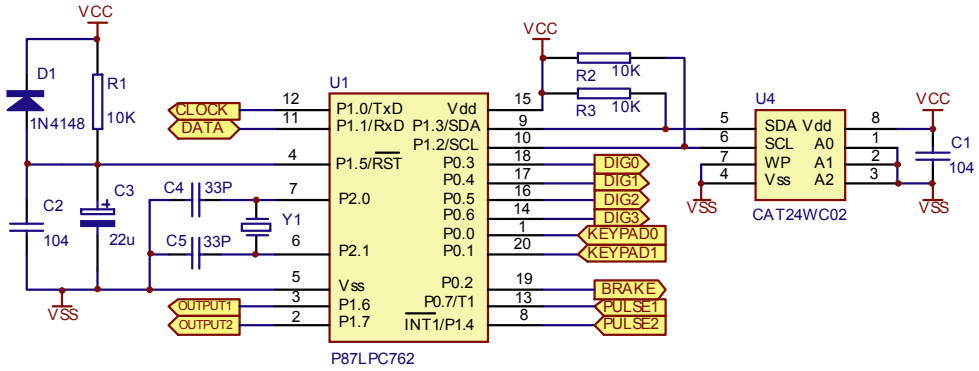

图 5　信号处理部分电路图

51LPC 系列单片机有内部复位功能，考虑到计数器有可能长时间工作和为了保险起见，我们还是采用了外部复位。

3．　键盘和显示部分电路

电路图如图 6 所示。在这部分扩展了 1 片 74HC164。拟设定的最大圈数为 9999，故安排了 4 个显示器。采用动态扫描方式显示。每个数码管公共端都加上了三极管，是作扫描信号的电流放大以驱动各数码管。

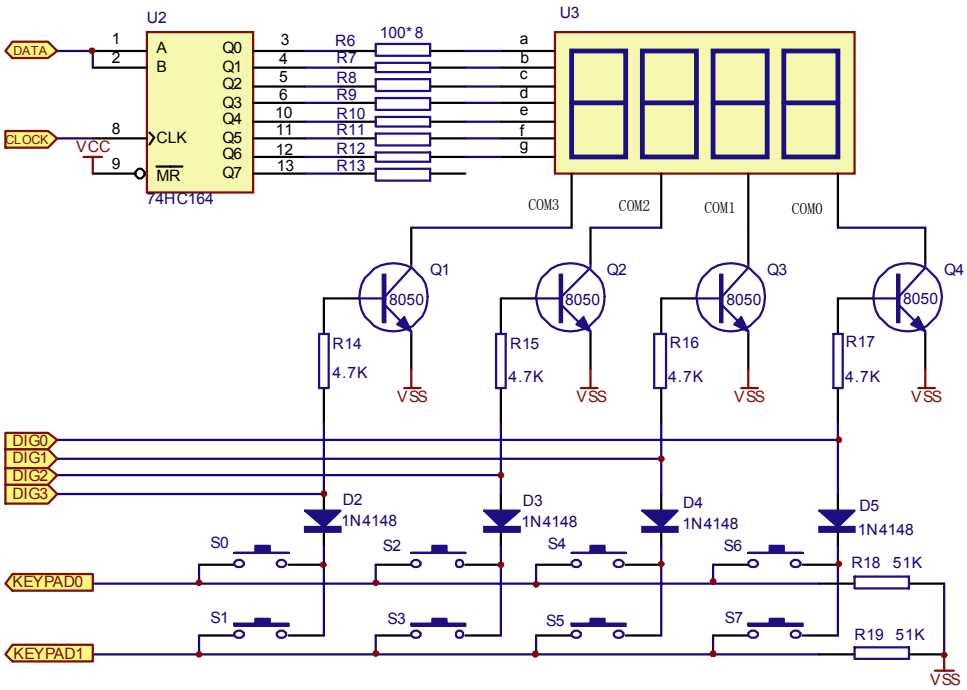

图 6　键盘和显示电路

采用矩阵键盘，一共安排了 8 个按键。4 根键盘扫描线和显示器位选线复用，这样只需要 2 根键盘回送线。显示器选用了共阴数码管，所以两键盘回送线接了低电平。选用 51K 电阻作为下拉电阻是为了减少按键对显示所产生的干扰。键盘扫描线上的二极管，是为了防止多根扫描线上的键被同时按下对显示产生干扰。

4．　电机驱动电路

利用继电器来控制电机(注意，是直流电机)。为了能够控制电机转动的方向，我们采用了 2 个继电器(参见图 7)。

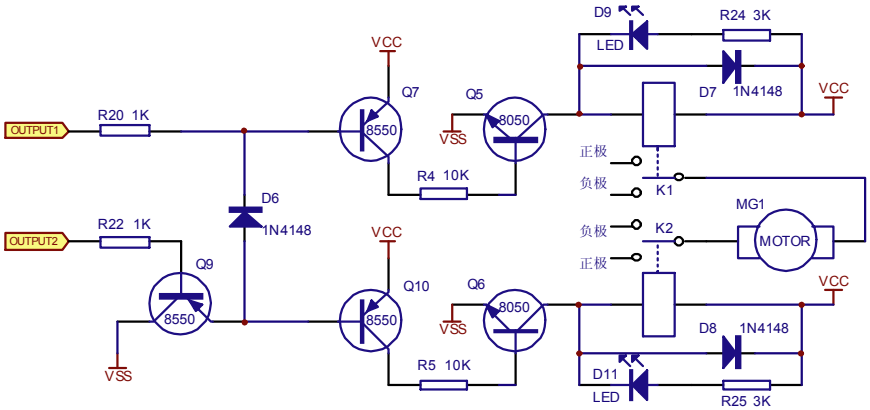

图 7　电机驱动电路

三极管 Q7、Q9、Q10 和二极管 D6 组成了互锁电路，这样能够确保不会出现 2 个继电器同时工作的情况。二极管 D7、D8 分别保护各自对应的继电器。当某路继电器工作时，相应的指示灯 D9、D11 点亮。

接入电机时，需遵循这样的规则：继电器的常开端接电机供电电源正极而常闭端接负极(参见图 7)。

二、软件设计

该计数器处理传感器送来的脉冲信号，而脉冲的到来是随机的，因此我们把脉冲信号当作中断来处理，结合硬件电路，我们把计数脉冲接到外部中断 1 上。对于键盘和显示，由于采用动态扫描，因此安排了定时中断 0 来处理。程序整体流程如图 8。

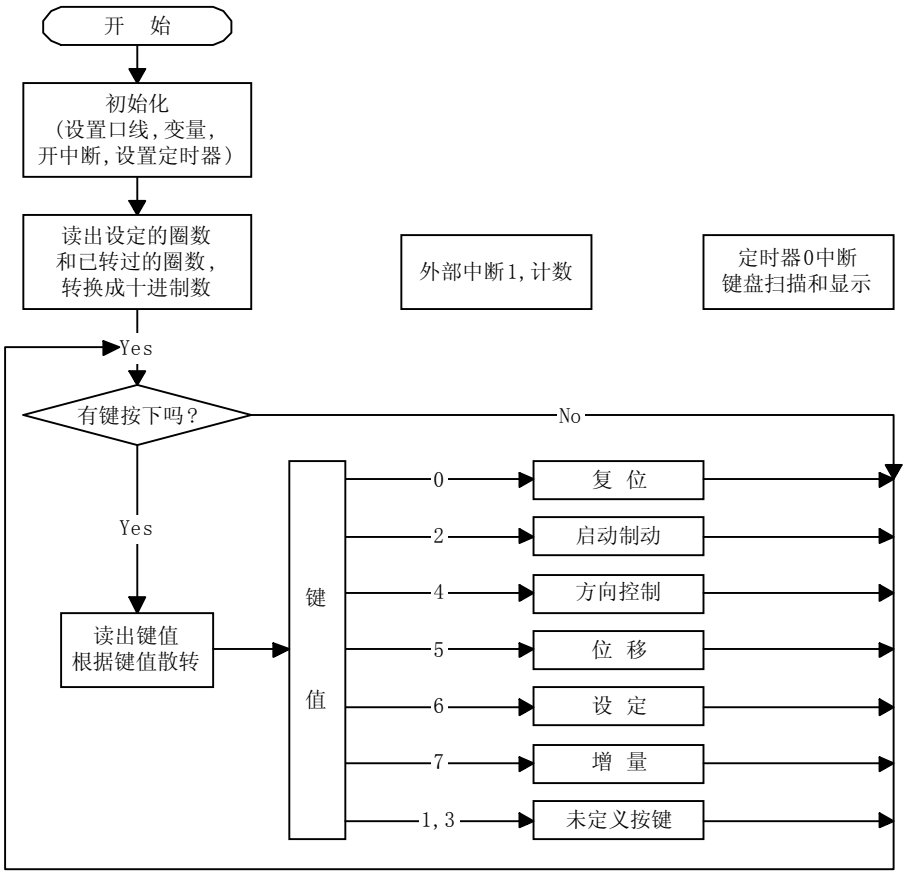

图 8 程序整体流程图

(一) 主程序

主程序结构很简单。从图 8 中可以看到，出程序主要完成对系统的初始化和处理按键，在没有按键的情况下，主程序不用做任何事情。现对初始化程序和按键服务程序作说明。

1．初始化程序

初始化程序完成对中断进行设置，单片机口线以及变量的初始化，具体操作参见源程序。

2．复位(键值为 0)

按下该键，系统复位，清除所有计数信息，包括设定圈数和已经转过的圈数。

3．启动制动(键值为 2)

按下该键，电机停止工作，同时启动刹车装置；再次按该键，电机继续工作。

4．方向控制(键值为 4)

系统上电按启动键后，电机按照默认的方向转动；制动后按该键再启动，电机转向。

5．位移(键值为 5)

在设置时按该键，可在 4 位键来回移动，选中位闪烁。

6．设定(键值为 6)

系统上电，显示已经转过的圈数，按该键，显示上次设定的圈数，再次按该键，则可以设定圈数，在这种状态下，会有 1 数码管闪烁，等待设置，设置好后，按该键确认并且保存，回到初态(显示当前转数)。

7．增量(键值为 7)

在设置状态下，按该键，选中位数字在 0~9 之间变化。

8．未定义键(键值为 1、3)

未定义，空操作；不同用户可用于需要的功能扩展。

(二)外部中断 1

1．外部中断 1 主要完成计数和转向识别，流程图如图 9。

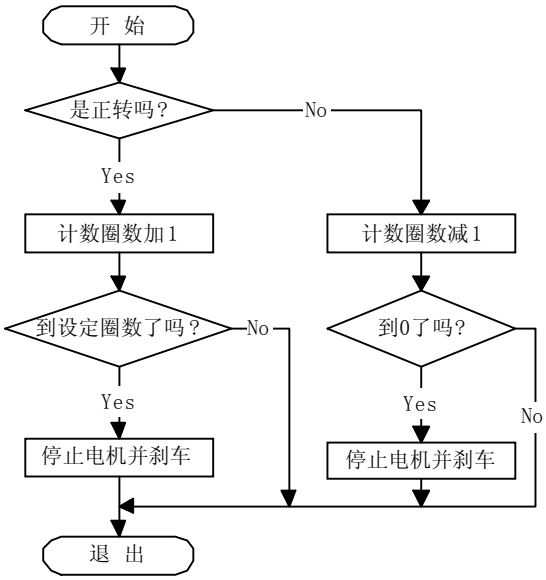

图 9　外部中断 1 流程图

2．转向识别

2 个传感器必须按照图 10 所示安装。

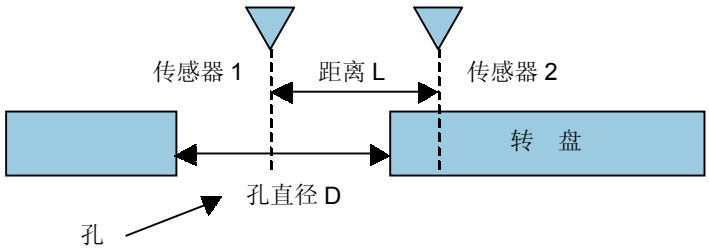

图 10　2 个传感器的安装示意图

为了识别转向，2 个传感器的距离 L 和捲绕盆上孔的直径 D 须满足如下的要求：D>L，当 D＝(N+1/2)L(N>0)时，2 个脉冲的相位刚好相差 90º，识别转向最可靠，这样将会得到如图 11 所示的波形。

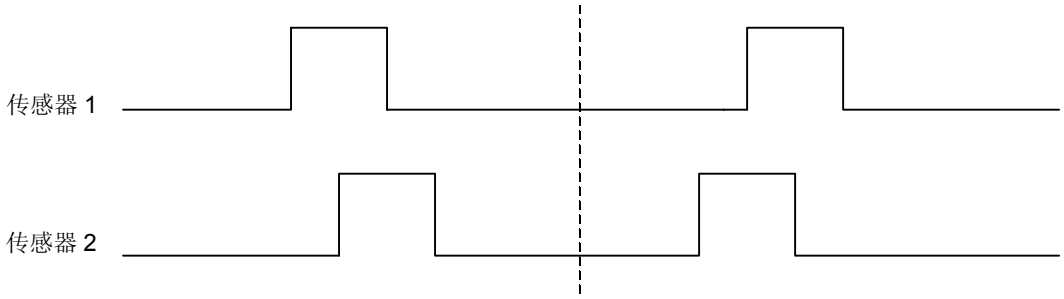

图 11 识别转向波形图

虚线两侧的波形各自代表了一种转向。传感器 2 接到外部中断 1。很显然，在虚线左侧，下降沿到来时传感器 1 的波形为低电平和在右侧则为高电平；据此就可以判断转向。

（二） 定时中断 0

定时器 0 中断则完成显示和键盘扫描的任务。流程图见图 12。

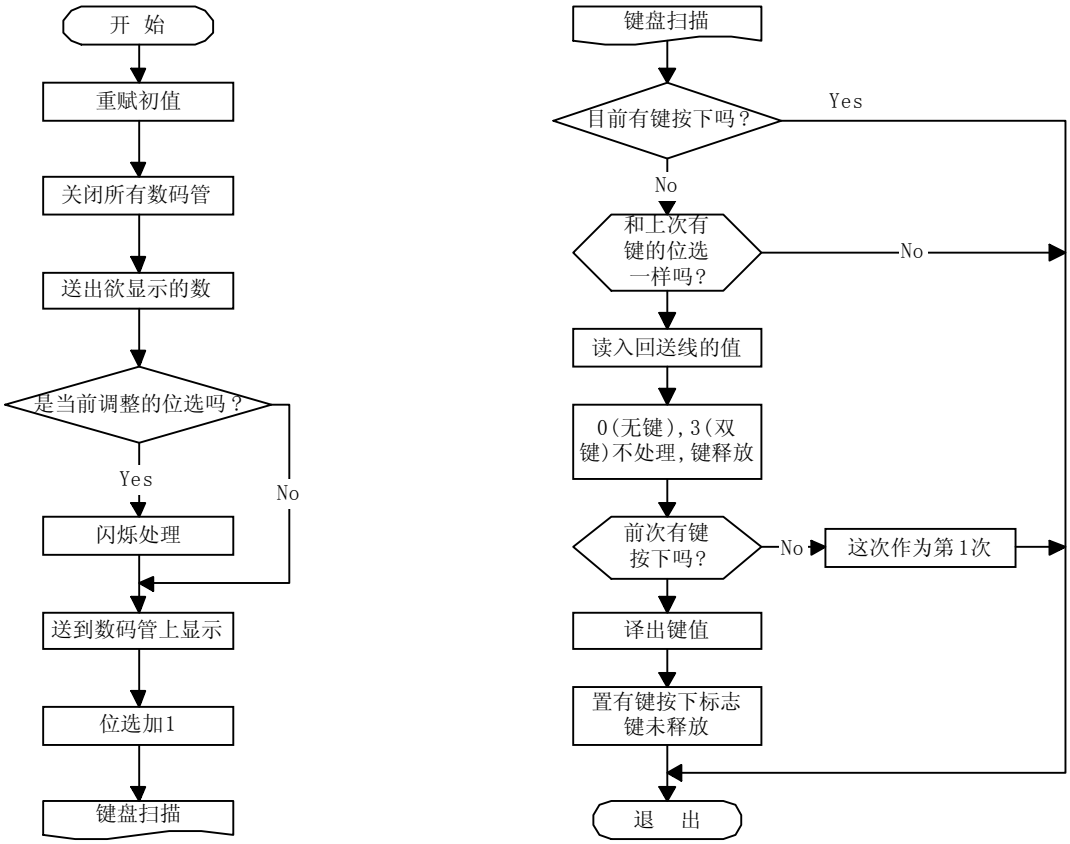

图 12 定时器 0 中断流程图

扩展了 74HC164，通过串口送数，只需要把欲显示的数据送到 SBUF 中去就可以了，再选通相应的数码管。为了显示的清晰，在每送数之前，需先关闭所有的数码管。

整个过程需要显示已经转过的圈数和所设定的圈数，程序根据状态标志变量来分别处理(参见显示部分程序)。

在中断中进行键盘扫描，很方便的进行了去抖动处理。具体操作请参考源程序。现对键值译码进行说明。为了键值的连续性和单一性，我们采取如下的方法译码：键值=位选值*2+回送线的值-1。位选有效值 0~3，回送线有效值 1、2，这样可以形成的键值就有 0~7，分别对应于图 6 的 S0~S7。

(三) 程序清单

注：因为采用了硬件 I²C 总线，故需要相应的驱动程序。本公司已经开发出了专门的驱动程序，封装成软件包，在需要的时候直接调用就可以了。

```
/*=============================================================*\
||键值和i，显示器的关系：                                      ||
||            显示器0     显示器1     显示器2     显示器3     ||
||i的值          0           1           2           3        ||
||键值及按键  复位键(0)   启动停止(2)  方向控制(4)  设定键(6)  ||
||            未定义(1)   未定义(3)    位移键(5)   增量键(7)  ||
||被调整的位可以闪烁；                                        ||
||可以保存已经转过的圈数和设定的圈数；                        ||
||可以识别转向和改变转向。                                    ||
||=============================================================*/

#include <REG764.H>
#include <HI2C_C51.h>                  //包含I²C软件包
#define uchar unsigned char            //宏定义
#define CLOCK_5MS (65536-5000)         //
#define CSI24WC02 0xA0                 //定义器件地址

sbit OutPut1=P1^6;                     //继电器控制口
sbit OutPut2=P1^7;
sbit Brake=P0^2;                       //制动控制口
sbit Int1=P1^4;                        //外部中断口
sbit Test=P0^7;                        //识别转向测试口

bit KeyDown;                           //有键按下标志,有键按下置1
bit PreKey;                            //上次有键按下置1
bit KeyEsc;                            //键释放清零
bit AddUp;                             //增量键标志
bit Out1;                              //控制继电器用
bit Out2;                              //控制继电器用

uchar Start;                           //启动制动信号
uchar PreDig;                          //PreDig为上次有键按下的位选
uchar KeyVal;                          //KeyVal键值寄存器
uchar pos;                             //pos用来保存位选,为键盘扫描用
uchar i=0;                             //扫描显示位选控制字i
uchar dpos;                            //位移控制字
uchar turning;                         //转向标志

uchar flashtime;                       //闪烁时间控制
uchar Sett;                            //设置键标志寄存器
uchar DispBuf[5];                      //显示缓冲区
```

```
uchar Decim[5];                         //十进制数存放区(设定圈数用)
uchar LapseBuf[5];                      //已经转过圈数缓冲
uchar RstBuf[]={0,0,0,0,0};             //供复位用

unsigned int TrapNumber;                //定义圈数(设定)
int LapseTrap;                          //上次已经转过的圈数

uchar code SegTable[]=
{   0xfc,0x60,0xda,0xf2,0x66,
    0xb6,0xbe,0xe0,0xfe,0xf6 };         //7段代码表

uchar code Loc[]={0x40,0x20,0x10,0x08};         //位选控制字

void Key2 ();                           //Key2()函数声明

//************************************************//
//初始化函数
//************************************************//
void Init()
{           //输入输出口初始化=====================
            //P0.7上拉高阻输入,P0.6~P0.3上拉,P0.7高阻输入,P0.0~P0.1仅输入
    P0M1=0x83;                          //P0M1=10000011B
    P0M2=0x7c;                          //P0M2=01111100B

            //P1.4高阻输入,P1.6,P1.7上拉输出
    P1M1=0x10;                          //P1M1=00010000B
    P1M2=0xc0;                          //P1M2=11000000B

            //中断设置=============================
    EA=0;
    TMOD=0x01;                          //定时器0，方式1，16位定时器

    TL0=CLOCK_5MS&0xff;                 //定时时间5ms
    TH0=CLOCK_5MS>>8;

    EA=1;                               //开总中断允许
    ET0=1;                              //开定时器0中断
    TR0=1;                              //开定时器0

    EX1=1;                              //开外部中断1
    IT1=1;                              //下降沿触发

    ETI=1;                              //开定时器I中断
```

```
    KeyDown=0;          //变量初始化==================
    PreKey=0;
    KeyEsc=0;
    Sett=0;
    dpos=1;
    Out1=1;             //默认转向
    Out2=0;

    Brake=1;            //口线初始化==================
    OutPut1=0;          //两个继电器都关闭
    OutPut2=0;
    Int1=0;             //INT1口线
}

/*****************************************************/

void Delayx10MS(uchar count)
{
    uchar j,k;
    while (count--!=0)
    {
        for(j=0;j<10;j++)
            for(k=0;k<72;k++)
                ;
    }
}

//*************************************************//
//键盘扫描函数,键值放在KeyVal中
//执行完相应的按键服务程序后,需要把KeyDown清0
//Key2()函数读入回送线的值,并根据不同情况处理
//*************************************************//

void ScanKey()
{
    if(!KeyDown)                //如果目前没有键按下
    {
        if(PreDig==pos)         //和上次有键按下时的位选一样吗?
        {
            PreKey=1;           //一样, 置PreKey为1
            Key2();             //调用Key2()函数
            return;
        }
        else                    //不一样
```

```
            { if(PreKey) {return; }           //PreKey=1，返回
                PreKey=0;Key2();return;       //否则，置PreKey为0
            }
     }
}

void Key2()
{
    uchar tempr;                              //回送线置暂存器
    tempr=P0&0x03;                            //读入回送线的值，并保存

    if(tempr==0)                              //回送线值为0，没有键按下
        { if(PreKey)
            { PreKey=0;KeyEsc=0; }
         else return;
        }
     else
        {
            if(tempr==3)                      //回送线值为3，双键，不处理
            {   KeyEsc=1;   }
                if(KeyEsc)   { return; } //键还没有释放

            if(PreKey==0)
            { PreDig=pos;return; }            //如果上次没有键按下，这次作为第一次

            KeyVal=2*pos+tempr-1;             //键值=2*位选值+回送线值-1
            KeyDown=1;KeyEsc=1;               //置有键按下标志KeyDown=1,键还没有释放
            PreDig=pos;return;                //
        }
}


//*************************************************//
//把已经转过的圈数(十六进制数)转换成十进制数,并送到LapseBuf[]中
//*************************************************//
void HexToDecimal(unsigned int Hexm)
{
    LapseBuf[0]=Hexm/10000;
    LapseBuf[1]=Hexm/1000%10;
    LapseBuf[2]=Hexm/100%10;
    LapseBuf[3]=Hexm/10%10;
    LapseBuf[4]=Hexm%10;
}
```

```
//*****************************************************//
//定时器0，键盘扫描和显示
//*****************************************************//

void Timer0Scan(void) interrupt 1
{
    TL0=CLOCK_5MS&0xff;              //定时时间5ms
    TH0=CLOCK_5MS>>8;
    TF0=0;

    P0=P0&0x87;                      //关闭4个数码管
    SBUF=SegTable[DispBuf[i+1]];     //取出显示缓冲区里的数，
                                     //转换成7段代码送到SBUF中
    pos=i;                           //保存当前所扫描的位选,供键盘扫描用
    if(Sett==2&&dpos==(i+1)&&flashtime<125)
        P0=P0&~Loc[i];               //进行闪烁处理,被调整的位闪烁
    else
        P0=P0|Loc[i];                //扫描第i个数码管
        flashtime++;
        flashtime++;

    i++;
    if(i>=4)                         //i超过3就恢复为0
    i=0;                             //

    while(TI==0);
    TI=0;

    ScanKey();                       //调用键盘扫描函数

    switch(Sett)
    {
        case 0:                      //显示已经转过的圈数,保存已经转过的圈数＝＝
            {
                uchar i;
                HexToDecimal(LapseTrap); //把已转过圈数转换成十进制数送到缓冲区中
                ISendStr(CSI24WC02,0x20,LapseBuf,5); //保存已经转过的圈数

    TrapNumber=Decim[0]*0+Decim[1]*1000+Decim[2]*100+Decim[3]*10+Decim[4];
                for(i=0;i<5;i++)
                DispBuf[i]=LapseBuf[i];      //读入已经转过的圈数到显示缓冲区,
                break;
            }
        case 1:                      //显示上次设定的圈数＝＝＝＝＝＝＝＝＝＝＝
```

```
            {
                uchar j;
                IRcvStr(CSI24WC02,0x10,Decim,5); //先读出设定的圈数
                for(j=0;j<5;j++)
                    DispBuf[j]=Decim[j];          //送到显示缓冲区
                break;
            }
        case 2:                         //显示正在设定的圈数==========
            {
                uchar j;
                for(j=0;j<5;j++)
                DispBuf[j]=Decim[j];
                ISendStr(CSI24WC02,0x10,Decim,5);//保存设定的圈数
                break;
            }
        default:
                break;
    }
}

//*************************************************//
//外部中断1,计数,判别方向
//*************************************************//
void ExternInt(void) interrupt 2
{
    if(Test)                            //设检测到高电平为正转
    {
        LapseTrap++;                    //已经转过的圈数
        if(LapseTrap>=TrapNumber)
        {
            LapseTrap=0;                //计数圈数恢复为0
            OutPut1=0;                  //关闭继电器
            OutPut2=0;                  //
            Brake=0;                    //启动制动信号
            Start=0;                    //启动制动信号
        }
    }
    else                                //检测到低电平为反转
    {
        LapseTrap--;
        if(LapseTrap<0)
        {
            LapseTrap=0;                //圈数恢复为0
            OutPut1=0;                  //关闭继电器
```

```
            OutPut2=0;                  //
            Brake=0;                    //启动制动信号
            Start=0;                    //启动制动信号
        }
    }
}

//0000000000000000000000000000000000000000000000000000//
//系统复位函数
//0000000000000000000000000000000000000000000000000000//
void ReSet(void)
{
    uchar i,j,k;
    LapseTrap=0;                            //清除已经转过的圈数
    TrapNumber=0;                           //清除设定圈数
    for(i=0;i<5;i++)
    Decim[i]=RstBuf[i];                     //清除缓冲区
    for(j=0;j<5;j++)
    LapseBuf[j]=RstBuf[j];                  //清除缓冲区
    for(k=0;k<5;k++)
    DispBuf[k]=RstBuf[k];

    ISendStr(CSI24WC02,0x20,RstBuf,5);      //清除E²PROM
    Delayx10MS(2);
    ISendStr(CSI24WC02,0x10,RstBuf,5);      //清除E²PROM
}

//*****************************************************//
//设定圈数
//*****************************************************//
void SetTrap()
{
    if(AddUp&&(Sett==2))
    {
        Decim[dpos]++;
        AddUp=0;
        if(Decim[dpos]>=10)
        Decim[dpos]=0;
    }
}

//*****************************************************//
//读键值,根据键值执行不同的任务
//*****************************************************//
```

```
void GetKeyVal()
{
    switch(KeyVal)
    {
        case 0:                         //系统复位=======
            KeyDown=0;
            ReSet();
            break;

        case 1:                         //未定义========
            KeyDown=0;
            break;

        case 2:                         //启动停止=======
            Start++;
            if(Start>=2)
                Start=0;
            if(Start)
            {   OutPut1=Out1;           //启动电机
                OutPut2=Out2;
                Brake=1; }              // 清除制动信号
            else
            {   OutPut1=0;
                OutPut2=0;
                Brake=0; }              //紧急制动
            KeyDown=0;
            break;

        case 3:
            KeyDown=0;                  //未定义========
            break;

        case 4:                         //改变转向=======
            turning++;
            if(turning>=2)
                turning=0;
            if(turning)
            {   Out1=1;                 //转向1
                Out2=0; }
            else
            {   Out1=0;                 //转向2
                Out2=1; }
            KeyDown=0;
            break;
```

```
        case 5:          //位移键////>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
            dpos++;
            if(dpos>=5)
                dpos=1;
            KeyDown=0;
            break;

        case 6:          //设定//!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
            Sett++;
            if(Sett>=3)
                Sett=0;
            KeyDown=0;
            SetTrap();
            break;

        case 7:          //增量键//++++++++++++++++++++++++++++++++
            AddUp=1;
            KeyDown=0;
            SetTrap();
            break;

        default:
            break;
    }
}


//*************************************************//
//主程序
//*************************************************//
void main()
{
    Init();                          //初始化

    IRcvStr(CSI24WC02,0x10,Decim,5); //先读出设定的圈数
    TrapNumber=Decim[0]*0+Decim[1]*1000+Decim[2]*100+Decim[3]*10+Decim[4];

    IRcvStr(CSI24WC02,0x20,LapseBuf,5);  //读入已经转过的圈数
    LapseTrap=LapseBuf[0]*0+LapseBuf[1]*1000+LapseBuf[2]*100+LapseBuf[3]*10+LapseBu
f[4];

    while(1)
    {
```

```
        if(KeyDown)
            GetKeyVal();                //有键按下,去读键值

        else                            //没有键按下
        {}
    }
}
```

三．测试及使用说明

(一)测试现象

系统上电后，显示已经转过的圈数，可以按照需要进行设置，设置好后按启动制动键，电机开始工作，达到目标圈数电机停止并且启动制动信号。

(二)使用说明

上电后，显示已经转过的圈数，按设置键还可以看到已经设置的圈数，如果不需要，可以按复位键清除。第一次按设置键，显示设置的圈数，再次按设置键，进入设置状态，将会看到有一位数码管在闪烁，这表示可以对该位进行设置，按位移键，可对所有数码管进行设置，设置好后，按设置键，保存所设定的圈数并且回到计数(显示已经转过的圈数)状态。

若需要改变转向，必须先停止电机，按方向控制键后再启动电机即可。