高效利用 BLDC 电机控制资源奖

基于 dsPIC30F4011 的
全电动注塑机顶出伺服系统的设计

注册编号：MCHP16bitCDC0293

参赛队员：刘辉杰

摘　要：本设计以永磁无刷直流电动机（BLDCM）作为全电动注塑机顶出伺服系统的执行机构，以 dsPIC30F4011 为核心处理器驱动电路和控制电路，包括 IGBT 功率模块、键盘显示和通讯接口等部分。在系统的软件设计中，将所有的事件分成中速、低速事件两类，单片机通过定时器循环执行这两类事件。根据控制需求编写了各个模块程序，包括输出 PWM 信号、信号采集、换向控制、转速控制；并实现“速度－电流”双闭环控制的功能。
关键词：dsPIC 顶出伺服系统 永磁无刷直流电动机

1. 引言

随着近年来塑料制品应用领域的不断扩大，世界上对注塑机械的需求呈现了持续、大幅攀升的趋势。其中电动注塑机相比传统注塑机具有能耗低、各功能同步运行以及注塑准确性高等的优点，已成为近年各大公司研究的重点。

全电动式注塑机的所有运动机构都使用伺服电机驱动。所有动作和功能密切相关，一环接一环，每个环节都将影响整个注塑过程的质量。全电动式注塑机中总共有六个电机，本文对顶出机构的控制进行深入研究。在注射成型过程中，当塑件固化成型后就需要从模具中取出，一般采用顶出的方式让制品脱模。在电动注塑机中，顶出机构是由伺服电机通过同步带驱动滚珠丝杠来驱动的，如下图 1-1 所示。

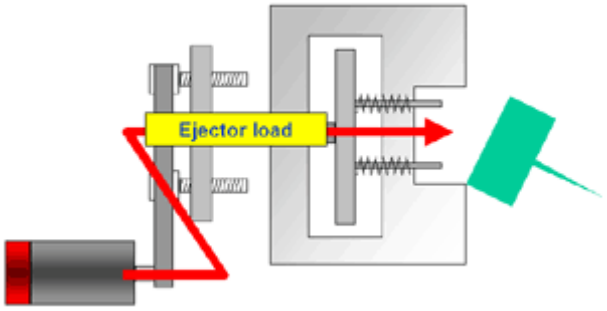

图1-1 顶出机构结构简图

由于注塑过程的工艺特点，需要精度要求高、容量较小的伺服系统的控制电机。因此在全电动注塑机中多采用永磁同步电动机。本研究将采用永磁无刷直流电动机(简称 BLDCM)作为顶出机构控制的电机。

无刷直流电机(BLDCM)作为一种电子驱动的无级变速电动机，具有结构简单、运行效率高、动态响应好等诸多优点，在家用电器和工业生产领域的应用日益广泛。对无刷直流电机控制主要通过数字信号处理器 DSP 实现。本文采用 dsPIC30F4011 作为控制器。

dsPIC30F4011 芯片是 Microchip 公司推出 16 位 dsPIC 数字信号控制器，其性能速度可以为 20MIPS，dsPIC30F CPU 模块采用 16 位（数据）改良的哈佛架构，并带有增强型指令集包含对 DSP 的有力支持。CPU 拥有 24 位指令字，指令字带有长度可变的操作码字段。程序计数器（PC）为 24 位宽，可以寻址高达 4M×24 位的程序存储器空间。

本文用 16 位 dsPIC 数字信号控制器控制全电动注塑机顶出系统的无刷直流电机，开发环境采用 MPLAB6.40，并用 Microchip C30 optimizing compiler (v1.10.02)进行编译，采用 MPLAB ICD2 进行调试和编程。采用无刷直流电机，

使控制系统的可靠性和实时性得到了提高，可以为企业提供技术上的借鉴,有利于系统的后续开发。

2.应用程序操作说明

设计中采用的各功能模块有定时器、I/O 口及 CN 模块、A/D 模块、PWM 模块和 I2C 模块。核心控制板如图 2-1 所示。在本设计中定时器用于脉冲计数和提供标准的实时时钟，其中 Timer1 用于电机的堵转保护和确定电机反转时机；I/O 口用于信号的输入输出，CN 模块是对电机反馈的位置信号进行判断；在无刷直流控制器中，A/D 模块用于对电机反馈的电流信号进行采集，也用于设定转速的可调电阻上的电压的采集；PWM 模块用于驱动 IGBT 模块,控制电机；I^2C 模块是用来与其他外设或单片机通信的串行接口，在这里与 ZLG7290 连接，用于显示控制参数和实时数据。

面板上有五个键和本子程序相关：MOVE、UP、DOWN、SET 和 FUN，它们的功能分别是：

MOVE 键：用来确定设定位，共有三位，个位、十位和百位。

UP 键和 DOWN 键：分别控制被控位的增与减，每次增 1 或减 1，在 0～9 之间循环。

SET 键：当用户确定其输入时可按 SET 键，将设定值存入对应的变量中。

FUN 键：参数切换键。在预定义的用户参数之间切换，方便参数的设置。

用于设定的电机参数和起动参数如附表 1 所示。电机参数的设置主要取决于所选择的电机。由于实现了两种不同的起动方法，同时还要对几个参数进行调整，以根据特定的应用调整起动。

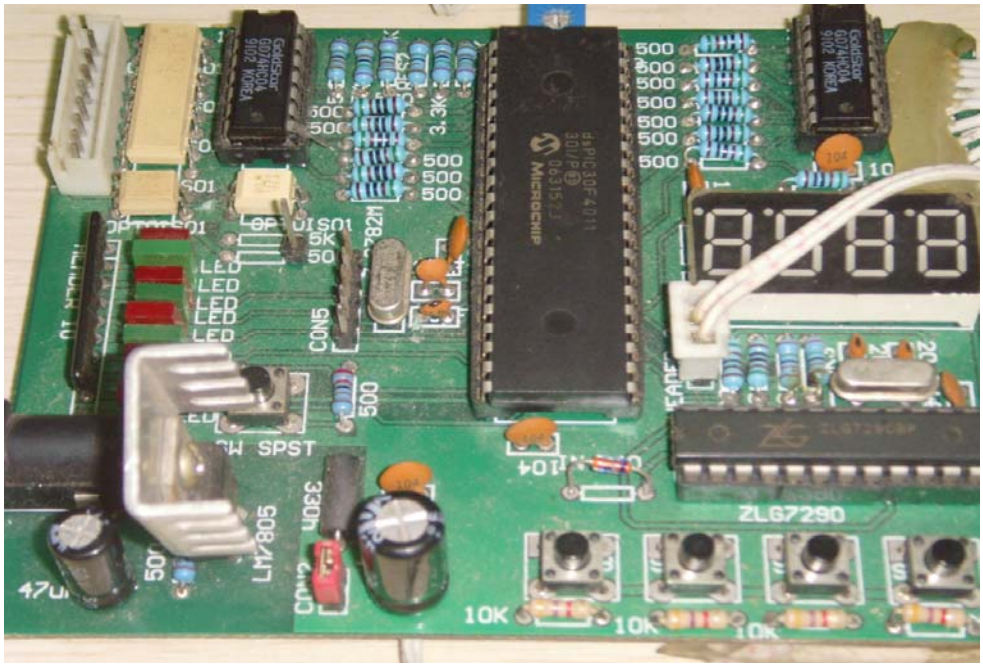

图 2-1 dsPIC 核心控制板实物图

3.原理图及主要应用电路说明

本节介绍驱动电路和控制电路的设计策略，以及以 dsPIC30F4011 为核心处理器的控制系统。硬件系统有三块电路板组成：开关电源电路板，IGBT 驱动电路板，dsPIC30F4011 核心控制板电路板。图 3-1 是板际间的电路连接示意图。

开关电源提供电机的霍尔传感器 15V 的电压；提供 IGBT 驱动板四组 20V 的光藕电源和 IGBT 开路的 300V 电压；提供控制板 12V 的电压。控制板采集电机霍尔传感器的反馈；提供 PWM 信号给 IGBT 开关器件。

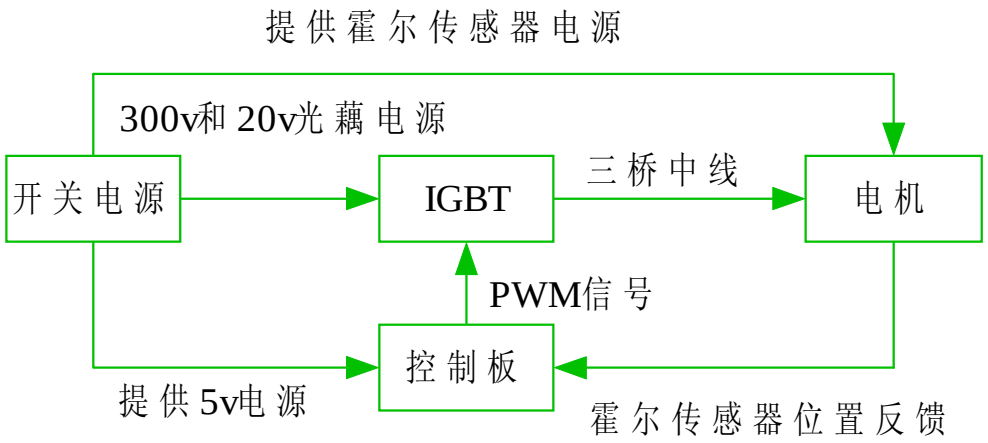

图3-1 板际间的电路连接示意图

3.1 电源电路

电源电路的输入为 380V 工频交流电，经整流，滤波，稳压后为稳定的直流电压，再经电感和电容滤波后作为逆变单元和开关电源单元的电源。电源开关电源电路输入为 220V 工频交流电。可以输出 380V、15V、20V、300V 和 12V 的电压。380V 电压供给电机，15V 电压供给霍尔传感器，20V 供给四组 IGBT 驱动板的光耦电源，300V 电压给 IGBT 驱动电路；12V 电压供给控制板。

3.2 IGBT 驱动电路

电机驱动器由 IGBT 分立元件组成，采用高速光耦对其驱动。IGBT 的驱动电路必须具备 2 个功能：一是实现控制电路与被驱动 IGBT 栅极的电隔离；二是提供合适的栅极驱动脉冲。实现电隔离可采用脉冲变压器、微分变压器及光电耦合器。本设计采用 TLP250 功率驱动模块。TLP250 内置光耦隔离可直接驱动 50A/1200V 以内的 IGBT。

3.3 电流反馈电路

采用基于磁补偿原理的HoneyWell霍尼韦尔公司的CSNE151-100这款多量程小体积电流传感器，它可以测量直流、交流或者脉动电流，原/副边电路之间电气绝缘性好，绝缘电压高达 5000V。传感器的供电为±15V 的直流电源，输出一个正比于测量电流的模拟电压，其以供电直流电源的地作为参考地的。需对两路定子电流进行 AD 转换，使调理后的电压限制在 0～5V 之间，满足 dsPIC 的 A/D 转换要求。

3.4 dsPIC30F4011 核心控制板电路

控制板电路包括以下几个部分：编程下载口电路、输入电路、键盘显示电路、电源电路和输出电路。其中电源电路输入 12V，经 7805 转换后为电路板提供 5V 电压。输出为 PWM，用于驱动 IGBT 模块。整个原理图参见图 3-2。

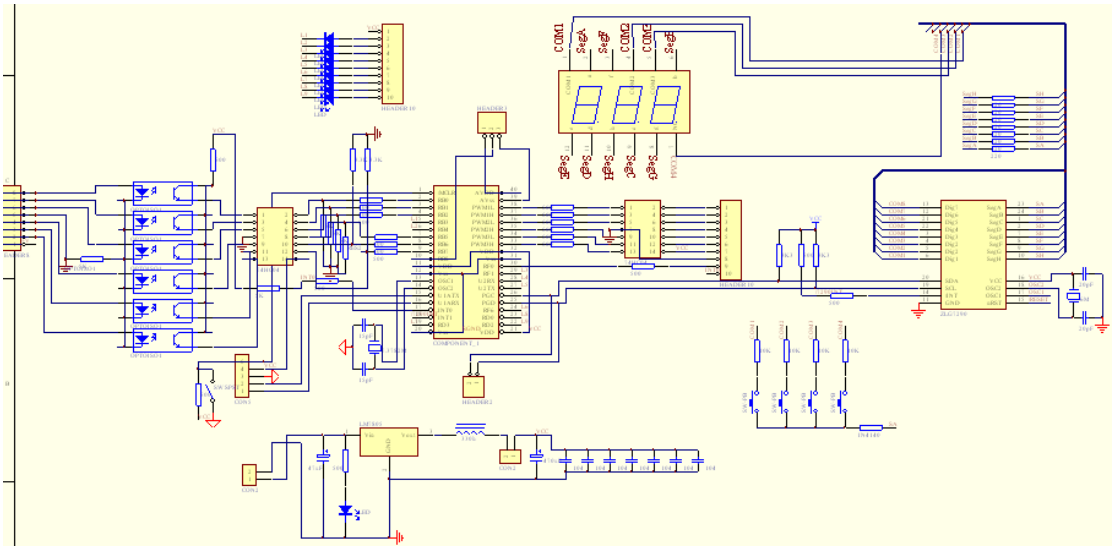

图 3-2 核心控制板电路原理图

i. 编程下载口电路

程序的烧写是 MPLAB ICD2 通过六芯的模块接口电缆与目标 PIC 单片机相连。ICD2 连接插座有六个引脚，但只使用了其中的五个引脚。目标 PCB 上 PICmicro 单片机与连接插座的连线。在 VPP/MCLR 线和 VDD 之间接一个上拉电阻（通常约为 10K 左右），这样 VPP/MCLR 线可置为低电平来复位 PIC 单片机。

3.4.2 霍尔位置传感器反馈输入电路及sp脉冲信号输入电路

如图 3-2 所示，HEADERS 是电机转 120°脉冲的 sp 信号及由电机反馈的三路霍尔的反馈信号，经过光藕隔离后再经 74HC04 后分别接单片机的电平变化中断(CN)捕捉口 RB0、RB1、RB2 和外部时钟计数器 T1 的 RC14。

3.4.3 键盘显示电路

控制板的键盘显示电路采用周立功的 ZLG7290，如图 3-2 所示。ZLG7290 的 I^2C 接口传输速率可达 32kbit/s 容易与处理器接口并提供键盘中断信号。为确保某个有效的按键动作，所有参数寄存器的同步性，建议利用 I^2C 通信的自动增址功能连续读 RepeatCnt FunctionKey 和 Key 寄存器。ZLG7290 的控制和状态查询全部都是通过读/写寄存器实现的，只需象读写 24C02 内的单元一样即可实现对 ZLG7290 的控制。

3.5 调速控制和驱动电路

图 3-3 是对三相无刷直流电动机用软件实现全数字双闭环控制的框图。给定转速与速度反馈量形成偏差，经速度调节后产生电流参考量，它与电流反馈量的偏差经电流调节后形成 PWM 占空比的控制量，实现电动机的速度控制。电流的反馈是通过测量两个桥臂上的电流感应电压来实现的。速度反馈则是通过霍尔位置传感器输出的位置量，经过计算得到的。位置传感器输出的位置量还用于控制换相。

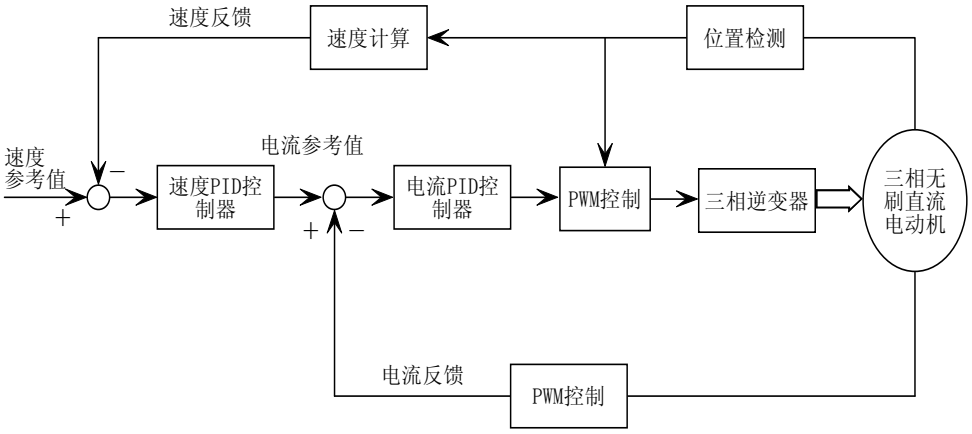

图3-3 三相无刷直流电动机调速控制框图

采用 dsPIC30F4011 芯片实现三相无刷直流电动机调速的控制和驱动电路的简图。在这个例子中，三个位置间隔 120°分布的霍尔传感器 H1、H2、H3 经整形隔离电路后分别与 dsPIC30F4011 的三个捕捉引脚 CAPl、CAP2、CAP3 相连，通过电平变化中断来给出换相时刻，同时给出位置、转速信息。

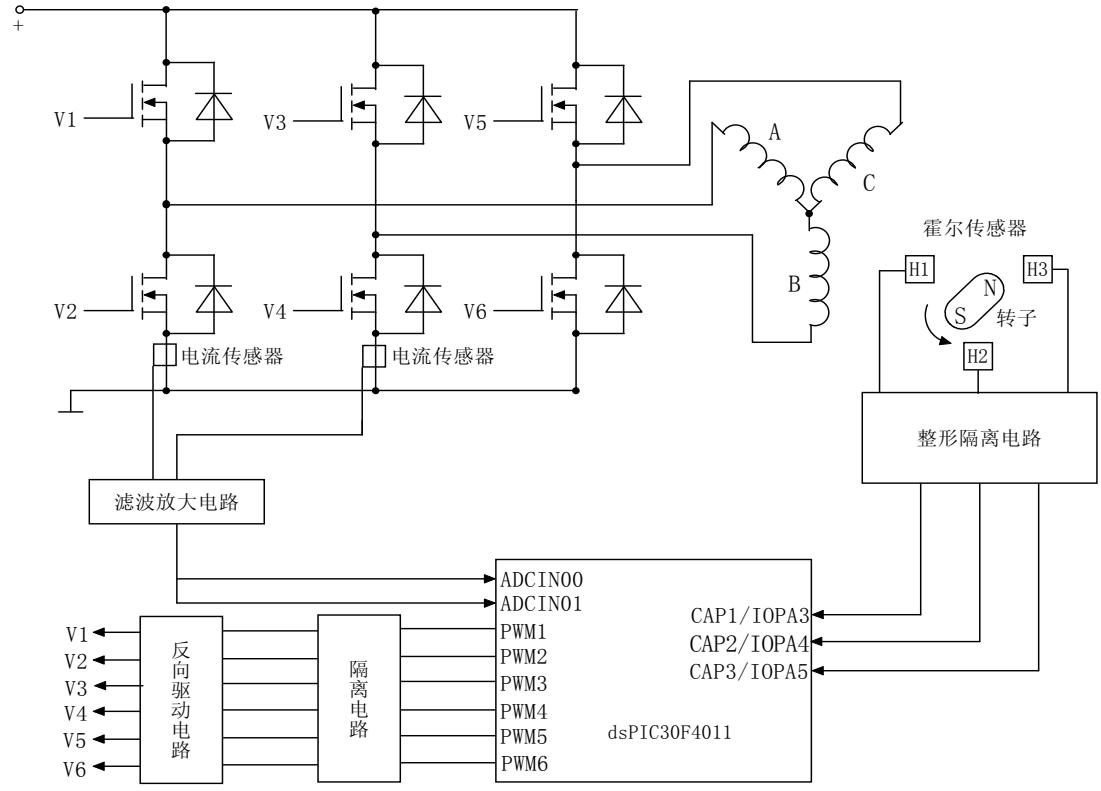

图3-4 DSP控制与驱动电路示意图

由于电动机每次只有两相通电，其中一相正向通电，另一相反向通电，形成一个回路，因此每次只需控制一个电流。两个电流传感器，将其安放在电源对地端，就可方便地实现电流反馈。电流反馈输出经滤波放大电路连接到 dsPIC30F4011 的 ADC 输入端口（图 3-4 中 DSP 控制和驱动电路 ADCIN0、ADCIN0），在每一个 PWM 周期都对电流进行一次采样，对速度(PWM 占空比)进行控制。

dsPIC30F4011 通过 PWMl～PWM6 引脚经一个反相驱动电路连接到六个开关管，实现定频 PWM 和换相控制。

4.流程框图及编程思想

程序开始运行之后，首先要对系统以及各变量进行初始化，然后再根据不同的运行模式进行相应的处理。

控制器软件的总体架构包括三部分：主程序、中断子程序、其它辅助子程序。软件结构是以主程序为主，通过函数调用和全局变量与子程序进行参数传递。下面先对主程序进行介绍，主程序是一个死循环结构，其功能是对系统进行必要的初始化：获得用户设定输入，包括运行模式设定以及预置数设定：然后，根据不同的运行模式调用相应的子程序，开相应的中断。对电机的换向控制主要是在中断子程序中进行的。部分源代码参见附录 2。

主程序流程框图如图表 4-1 所示。主要完成上电初始化，如锁相环、看门狗、系统控制寄存器、其他寄存器、ADC 模块及 CAN 通信接口，建立和分配各种初始化数据区等工作。然后查询以 PWM 时基定时 10ms 的中速事件是否到，如果时间到了就调用图表 4-2 的中速事件。然后查询以 PWM 时基定时 50ms 的低速事件是否到，如果时间到了就调用图表 4-3 的低速事件。

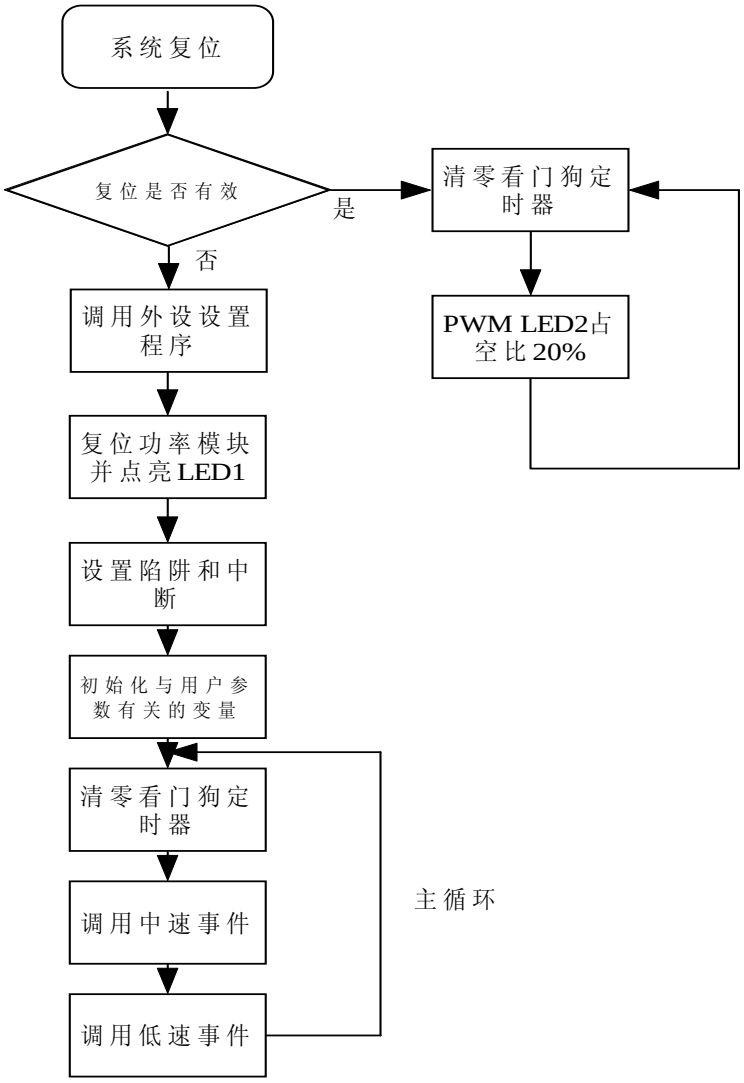

图4-1 主程序流程框图

图 4-2 是中速事件处理流程框图，程序首先判断中速事件的标志位 medium_event_count 是否等于 10ms，是则将其清零，处理接下来的程序。读取 Timer2 中断算出的电机转速，读取 A/D 中断处理程序得到的电机电流，分别采用增量式的 PID 算法，调节 PWM 输出的脉宽，实现对无刷直流电机的双闭环控制。

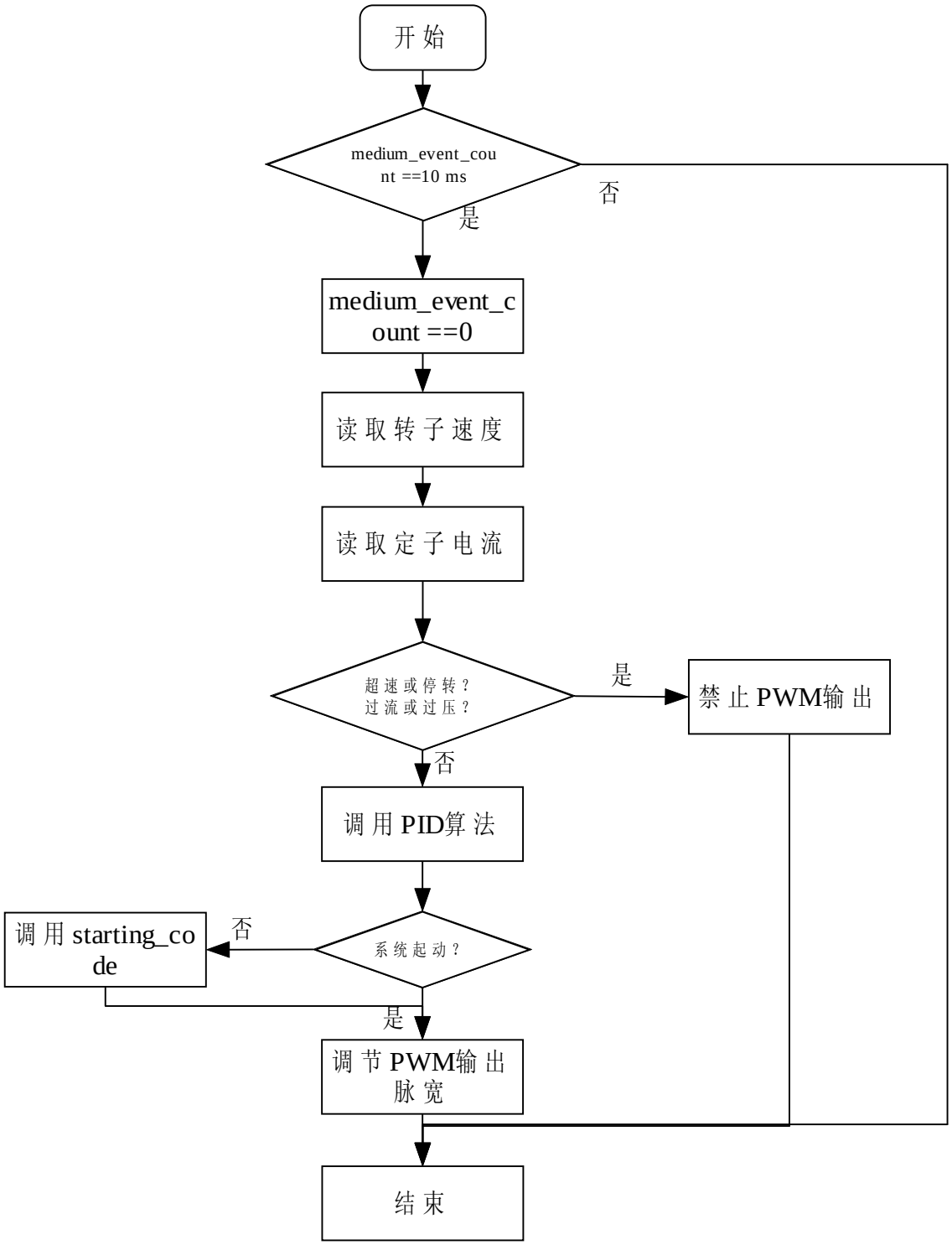

图4-2 中速事件处理流程框图

图 4-3 是低速事件处理流程框图，程序首先判断低速事件的标志位 slow_event_count 是否等于 50ms，是则将其清零，处理接下来的程序。首先显示当前显存中的内容，接着查询有无按键按下，如果有则进入按键处理程序。查询 I^2C 工作方式的串行口通讯标志位是否置位，如果置位再判断电机是否运转，如果运转则要禁止除换向外所有中断，再接受串行口通讯的数据。

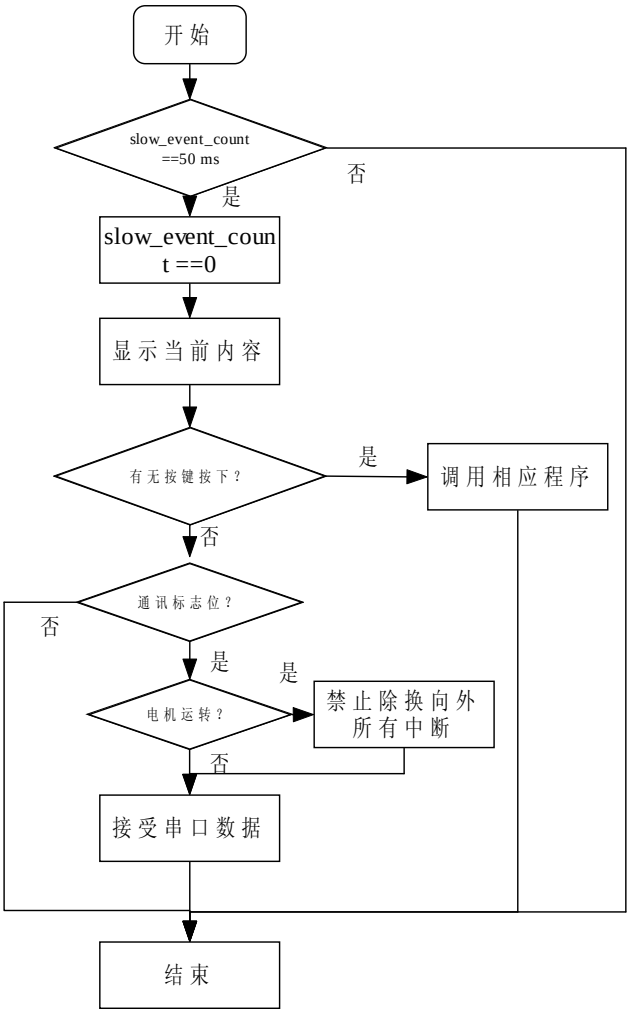

图4-3 低速事件处理流程框图

5.实验结果测试说明

文中从全电动注塑机顶出机构的实用要求和提高控制系统可靠性的角度出发，给出了以 dsPIC30F4011 处理器为核心的无刷直流电机硬件构建方案和无刷直流电机控制软件的整体框架，并在实验中验证了可行性。

实验用的测量系统结构框图如图 5-1 所示，由 dsPIC 控制器根据无刷直流电机反馈回来的位置信号，按照设定的速度施加电压信号给无刷直流电机相应绕组，力矩转速传感器用来测试无刷直流电机的转速和突变负载的大小，并将其相对应的模拟信号传递给 labview 测试系统，从而将实验结果测试出来。

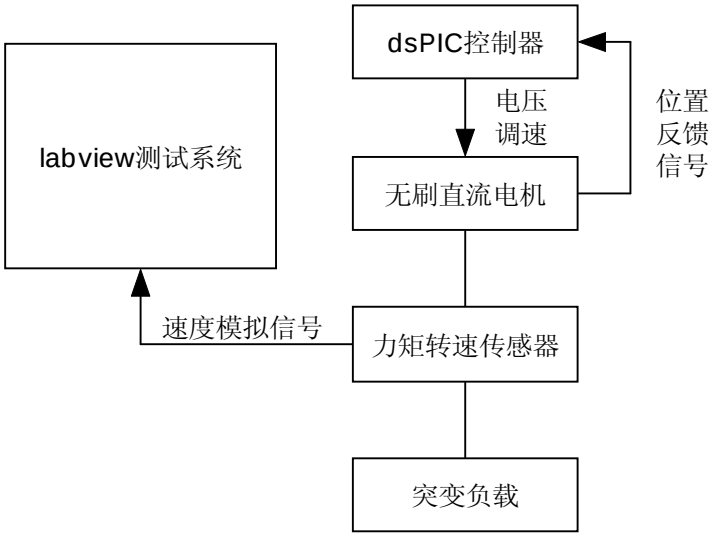

图5-1 实验测试系统框图

在制品脱模后，电机要反转，顶出电机的转速要经过减速－慢速－反转－慢速－加速的变化过程。电机的速度平滑的下降，在底部保持低速一段时间，使制品与顶杆之间产生一个速度差，有利于制品的脱模，然后顶出电机反转回复到顶出前的位置。

实验方法：在一定负载下，设置一个初始转速，观察电机的启动升速特性曲线。转速由 JN338 测量，它在后面板输出跟转速同频率的方波脉冲，使用 NI 的定时器 I/O 计数器采集卡 6602 进行脉冲计数，在 LabView 中显示变化过程曲线。

测量结果如图 5-2 所示，在 1.965Nm 的负载力矩下，给定 620 转/分的初始转速值，伺服电机在大约 600 毫秒左右的时间速度下降到 50 转/分左右，保持 350 毫秒左右，在次期间，速度很快下降到 0 转/分，然后反转又上升到 50 转/分左右。经过 1000 毫秒左右的时间上升到给定转速值，响应快速跟上设定的速度，速度有 1%左右的上下波动，取得了较好的效果。

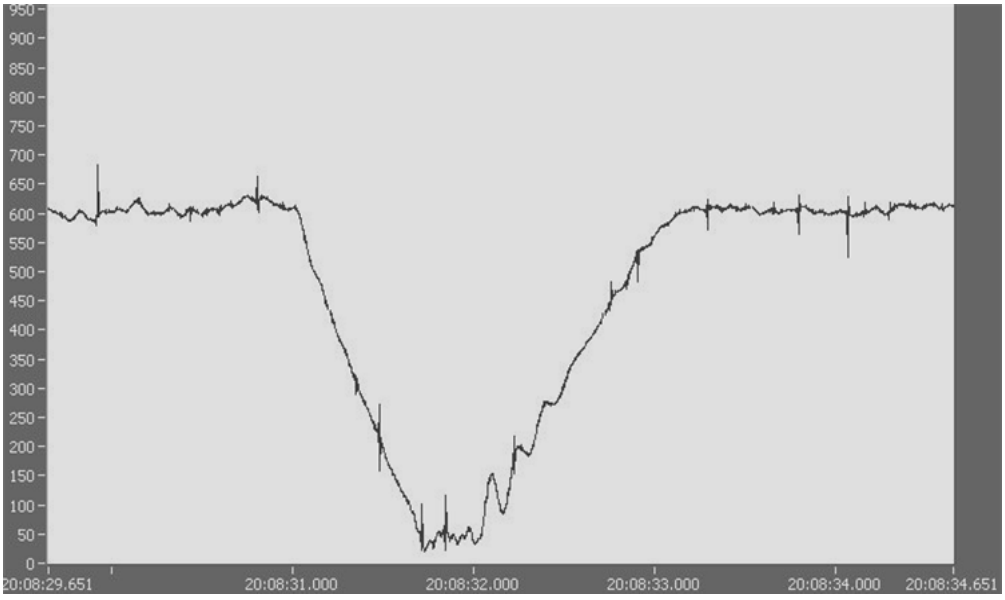

图5-2 顶出电机顶出到位反转过程

附录1 控制参数参考简表

参数编号	参数名称	描述	缺省
1	DIRECTION DEMAND	决定旋转方向	1
2	CONTROL MODE	决定速度控制方法： 0 －闭环速度控制，速度环输出直接控制PWM占空比。 1 －闭环速度控制，速度环输出作为电流环给定。 2 －开环速度控制，通过电位器调节的电压给定直接控制PWM占空比。 3 －开环速度控制，通过电位器调节的电流给定。 给定值由电位器VR2提供。	3
3	Lock Pos.1 Time	第一次开始锁定的时间，以10ms为增量	100
4	Lock Pos.2 Time	第二次开始锁定的时间，以10ms为增量	100
5	Lock Pos.1 Dem	第一次开始锁定的百分比给定	44
6	Lock Pos.2 Dem	第二次开始锁定的百分比给定	44
7	Ramp Start Speed	开始加速时的速度值（RPM）	20
8	Ramp End Speed	结束加速时的速度值（RPM）	250
9	Ramp Start Dem	开始加速的给定百分比%	49
10	Ramp End Dem.	结束加速的给定百分比%	50
11	Ramp Duration	加速过程持续时间，增量为10ms	200
26	No. Motor Poles	电机转子的极数，不是极对数。因此，极数应该是偶数。这个参数影响所显示RPM值的速度比例。	10
28	Current Scale		529

30	Voltage Scale	件电阻分压器的比值。	1035
33	Blanking Count	在换相后禁止PWM输出的死区周期数。这个延时也为非驱动相中电流的放电提供了时间。	1
40	ZeroX Enable Speed	采用采集方法1时，这个参数设置使能位置检测时的开环步进速度。	100
42	Starting Control	当这个参数设置为1时，使用电压控制起动。当设置为0时，使用电流控制起动。	1
43	Windmilling Dem.	这个参数适用于起动期间检测到旋转的情况（此时，转子运转方向与给定方向相反）。它设定用于将电机减速至静止状态的电流给定值。	20
44	Braking Ramp T	这个参数设置旋转时电机速度减为零所耗费的时间，增量为10ms。	200
45	Acquire Method	当此参数设置为0时，采用采集方法1起动。此参数设置为1时，采用采集方法2起动。	1

附录2 各个外围模块编程及部分源代码

```
#include "p30F4011.h"
#include "ZLG7290.h"
#define FCY    5000000                  //Fosc=20.0Mhz
#define MILLISEC FCY                    //1mSec delay constant
#define FPWM 10000              //Value used to simplify math on the Duty Cycle registers calculation
#define PDCSTART 80
#define PDCMAX 600
#define PDCMIN 80
#define SPEEDMAX 1500
#define SPEEDMIN 30
//This bit structure provides status flags for the software
        struct FLAGS{
                unsigned Block:1;
                unsigned Reverse:1;
                unsigned Button1:1;
                unsigned Button2:1;
                unsigned Button3:1;
                unsigned ADCEvent:1;
                unsigned DirChange:1;
                unsigned Accelerate:1;
                unsigned PWMFault:1;
                unsigned Restart:1;
                unsigned CCW:1;
                unsigned T3Event:1;
                unsigned CNEvent:1;
                unsigned PWMEvent:1;
                unsigned MediumEvent:1;
                unsigned SlowEvent:1;
```

```
        } Flags;
//counter-clockwise
const unsigned int StateLoTableCCW[]={0x0000, 0x2001, 0x0810, 0x0801,0x0204, 0x2004, 0x0210,
0x0000};
//clockwise
const unsigned int StateLoTableCW[]={ 0x0000, 0x0810, 0x0204, 0x0210, 0x2001, 0x0801,
0x2004,0x0000};
unsigned int HallValue;          // Variable containing the Hall Value from PORTB
unsigned int counter, counter2;
unsigned int countersave[2];
unsigned int SpeedSet, Speed, SpeedError;
signed int Error, SpeedAdd;
unsigned int PDCref, PDCTEMP;
unsigned int DispCounter;
unsigned int ProtectionON;
float Kp, Ki, Kd;
void InitVar();
void InitHall();
void InitADC10(void);                    //A/D Init Subroutine to Read POT.
void InitMCPWM(void);                    //Init PWM Module to drive the Inverter
void InitCN(void);                       //Change Notification for Hall Effect Sensors
void InitTMR1(void);
void InitTMR3(void);
void DelayNmSec(unsigned int N);   //General Delay subroutine
unsigned int ReadADC(unsigned int channel);
void Start();
//------------------------------------------------------------
//名称：CN中断函数
//功能：换相
//------------------------------------------------------------
```

```
void__attribute__((__interrupt__))_CNInterrupt (void)
{
    _CNIF = 0;
    HallValue = (unsigned int)(PORTB ) & 0x0007;
    OVDCON = 0;                 //换向保护
    if(Flags.CCW)
        OVDCON = StateLoTableCCW[HallValue];
    else
        OVDCON = StateLoTableCW[HallValue];
    counter++;
    counter2++;
}
//------------------------------------------------------------
//名称：Timer1中断函数
//功能：堵转保护；
//          确定反转时机
//------------------------------------------------------------
void __attribute__((__interrupt__)) _T1Interrupt(void)
{
    _T1IF = 0;
        _RF1 = 0;
        if(Flags.Reverse && (counter == 0))
    {
        Flags.Restart = 1;
    }
    else if(ProtectionON && (counter == 0))          //启动及堵转保护子程序
    {
        _CNIE = 0;
        OVDCON = 0;
        PDC1 = 0;
```

```
            PDC2 = PDC1;
            PDC3 = PDC1;
            InitTMR1();
            InitTMR3();
            Flags.Block = 1;
        }
        Flags.ADCEvent = 1;
        countersave[DispCounter] = counter;
        counter = 0;
        if(DispCounter < 2) DispCounter++;
        else DispCounter = 2;
}
//------------------------------------------------------------
//名称：主函数
//------------------------------------------------------------
int main(void)
{
        unsigned int SpeedDisp;
        InitHall();         //初始化霍尔传感器读取功能
        InitMCPWM();        //初始化PWM模块
        InitCN();           //初始化CN中断
        InitTMR1();         //初始化Timer1
        InitTMR3();         //初始化Timer3
        InitADC10();        //初始化AD模块
        InitINT2();         //初始化显示功能
        i2cint ();          //初始化I2C
        ProtectionON = 1;
        _TRISD2 = 0;
        _RD2 = 1;
        _TRISD3 = 0;
```

```
        _RD3 = 1;
        _TRISF1 = 0;
        _RF1 = 0;
        Flags.CCW = 0;
        InitVar();
        KEY = 0;
        PDCref = 0;
        PDCTEMP = 0;
        Flags.T3Event = 0;
        DelayNmSec(50);
        dp_data(SpeedSet);
while(1)
        {
                while(KEY != 1)
                {
                        if(KEY == 2)                    //转向选择按键
                        {
                                KEY = 0;

                                Flags.CCW = !Flags.CCW;
                                _RF1 = !_RF1;
                        }
                };
                KEY = 0;
                _RD3 = 0;
                Start();
        while(KEY != 1)
                {
                        if(Flags.Block)                 //堵转保护
                        {
```

```
        _RD2 = !_RD2;
        DelayNmSec(500);
    }
    else
    {
        if(_T3IF)                   //调节控制量
        {
            _T3IF = 0;
            Speed = (unsigned int)((float)(counter2) * 48.83);
            Error =     (signed int)(SpeedSet - Speed);
            Kp = 0.2;
            SpeedAdd = (signed int)((float)Error * Kp);
            if(abs(Error) > 3)
            {
                PDCTEMP = (unsigned int)((signed int)PDC1 + SpeedAdd);
            }
            if(PDCTEMP > PDCMAX) PDCTEMP = PDCMAX;
            else if(PDCTEMP < PDCMIN) PDCTEMP = PDCMIN;
            PDC1 = PDCTEMP;
            PDC2 = PDC1;
            PDC3 = PDC1;
            counter2 = 0;
        }
        if(Flags.ADCEvent)          //电位器
        {
            Flags.ADCEvent = 0;
            SpeedSet = ReadADC(8) + 30;
            if(SpeedSet > SPEEDMAX) SpeedSet = SPEEDMAX;
            else if(SpeedSet < SPEEDMIN) SpeedSet = SPEEDMIN;
        }
```

```
                if(DispCounter == 2) //显示速度
                {
                    DispCounter = 0;
                    SpeedDisp = (unsigned int)((float)(countersave[0] + countersave[1]) *
4.88);
                    dp_data(SpeedDisp);
                }
                if(KEY == 2)                //反转
                {
                    KEY = 0;
                    Flags.CCW = !Flags.CCW;
                    Flags.Reverse = 1;
                    _RF1 = !_RF1;
                    _CNIE = 0;
                    OVDCON = 0;
                    InitTMR3();
                }
                if(Flags.Restart)           //重新启动，用于反转
                {
                    Flags.Restart = 0;
                    Start();
                }
            }
        }
        KEY = 0;
        _RD3 = 1;
        dp_data(0);
        _CNIE = 0;
        OVDCON = 0;
        PDC1 = 0;
```

```
            PDC2 = PDC1;
            PDC3 = PDC1;
            InitVar();
            InitTMR1();
            InitTMR3();
            _RD2 = 1;
        }
}
//------------------------------------------------------------
//名称：电机启动函数
//功能：启动电机
//------------------------------------------------------------
void Start()
{
    SpeedSet = ReadADC(8) + 30;
    if(SpeedSet > SPEEDMAX) SpeedSet = SPEEDMAX;
    else if(SpeedSet < SPEEDMIN) SpeedSet = SPEEDMIN;
    PDC1 = PDCSTART;                // PDC1/1000*100%
    PDC2 = PDC1;
    PDC3 = PDC1;
    _CNIE = 1;      // enable CN interrupt
    HallValue = (unsigned int)(PORTB) & 0x0007;   // Read Hall Effect Sensors
    if(Flags.CCW)
        OVDCON = StateLoTableCCW[HallValue];        // Load Commutation Value
    else
        OVDCON = StateLoTableCW[HallValue];
    _T1IE = 1;
    T1CONbits.TON = 1;
    T3CONbits.TON = 1;
}
```

```
void InitVar()
{
	Flags.Block = 0;
	Flags.ADCEvent = 0;
	Flags.Reverse = 0;
	Flags.Restart = 0;
	counter = 0;
	countersave[0] = 0;
	countersave[1] = 0;
	DispCounter = 0;
	counter2 = 0;
	SpeedSet = ReadADC(8) + 30;
	if(SpeedSet > SPEEDMAX) SpeedSet = SPEEDMAX;
	else if(SpeedSet < SPEEDMIN) SpeedSet = SPEEDMIN;
}
void InitHall()
{
	_PCFG0 = 1;
	_PCFG1 = 1;
	_PCFG2 = 1;
	_TRISB0 = 1;
	_TRISB1 = 1;
	_TRISB2 = 1;
	_CNIF = 0;
	_CNIE = 0;
	HallValue = 0;
}
//-------------------------------------------------------
void InitADC10(void)
{
```

```
    _PCFG8 = 0;
    ADCSSL = 0;
    ADCON1 = 0x00E0;            //sample time automatic
    ADCON2 = 0x0000;
    ADCON3 = 0x0701;            //Tsam = 7Tad ,Tad = Tcy
    IFS0bits.ADIF = 0;
    IEC0bits.ADIE = 0;
    ADCON1bits.ADON = 1;   //enable AD
}
/*********************************************************************
InitMCPWM, intializes the PWM as follows:
1. FPWM = 19531 hz
2. Independant PWMs
3. Control outputs using OVDCON
4. Set Duty Cycle with the ADC value read from pot
5. Set ADC to be triggered by PWM special trigger
*********************************************************************/
void InitMCPWM(void)
{
    PTPER = 500;            // FCY/FPWM - 1 = 499
    PWMCON1 = 0x0777;           // enable PWM outputs
    OVDCON = 0x0000;        // allow control using OVD
    PDC1 = 0;               // init PWM 1, 2 and 3 to 0%
    PDC2 = PDC1;
    PDC3 = PDC1;
    PWMCON2 = 0x0700;           // 16 postscale values
    PTCON = 0x8000;         // start PWM
}
void InitCN(void)
{
```

```
	CNEN1 = 0x001c;		// Enable CN2, CN3 and CN4
	_CNIF = 0;	// clear CNIF
	_CNIE = 0;	// enable CN interrupt
}
void InitTMR1(void)
{
	T1CON = 0x0030; // internal Tcy/256 clock
	TMR1 = 0;
	PR1 = 10000;		//delay 0.5s
	_T1IF = 0;
	_T1IE = 0;
}
void InitTMR3(void)
{
	T3CON = 0x0030; // internal Tcy/256 clock
	TMR3 = 0;
	PR3 = 2000;		//delay 0.1s
	_T3IF = 0;
	_T3IE = 0;
}
void DelayNmSec(unsigned int N)
{
	unsigned int j;
	while(N--)
		for(j=0;j < MILLISEC;j++);
}
unsigned int ReadADC(unsigned int channel)
{
	if(channel > 0x000F) return(0);
	ADCHS = channel;
```

```
        ADCON1bits.SAMP = 1;
        while(!ADCON1bits.DONE);
        return(ADCBUF0);
    }
    //-------------------------------------------------------------
    // I2C通讯,用于显示控制参数和实时数据
void i2cint()
{
    I2CSTAT = 0x00;
    I2CCON = 0x8000;              //使能I2C模块并配置SDA、SCL引脚配置为串行端口引
脚
    _I2CEN = 1;
    I2CBRG = 181;                 //对20MHz振荡，设定I2C速率约为100kHz
    _MI2CIE = 0;                  //允许IIC中断
    }
void i2cidle()
{
    while(_TRSTAT);               //直到发送状态位为“0”，即发送结束
    while(I2CCON && 0x001f);//等待硬件将SDA、SCL引脚初始化，并清零
}
void i2ctrn(unsigned int subaddr，unsigned int data1，unsigned int data2)
{
    _SEN = 1;                     //启动I2C
    while(!_MI2CIF);
    _MI2CIF = 0;
    I2CTRN = 0x70    ;            //发送从地址
    while(!_MI2CIF);
    _MI2CIF = 0;
    I2CTRN = subaddr;             //发送子地址
    while(!_MI2CIF);
```

```
    _MI2CIF = 0;
    I2CTRN = data1;                 //发送数据
    while(!_MI2CIF);
    _MI2CIF = 0;
    if(data2 != 0xff)
    {
        I2CTRN = data2;             //发送数据
        while(!_MI2CIF);
        _MI2CIF = 0;
    }
    _PEN = 1;                       //停止I2C
    while(!_MI2CIF);
    _MI2CIF = 0;
}
    //-------------------------------------------------------
```

参考文献:

1. 钟汉如．注塑机控制系统[M]．化学工业出版社，2004
2. microchip technology inc．dsPIC30F系列参考手册[M]．内部参考，2005年
3. 赵德申，胡雪梅. 开关电源高频变换电路结构与分析[J]. 装备制造技术.2007，6
4. 陈永军，翁惠辉，李俊杰.TLP250功率驱动模块在IRF840MOSFET中的应用[J].今日电子. 2005
5. 王晓明，王玲．电动机的DSP控制[M]．北京航空航天大学出版社，2004
6. 王宏华．新型交流电动机及控制技术系列讲座(3)永磁无刷电动机[J]．机械制造与自动化，2004
7. 张深．直流无刷电动机原理及应用[M]．机械工业出版社，1999: 1-120

[1] 原理及应用[M]．机械工业出版社，1999: 1-120