新唐 M451 无感 FOC 方案代码说明

www.nuvoton.com

nuvoTon

概要

- 代码总体架构
- PWM 输出极性和ADC 配置
- 电机参数配置
- 电流、电压测量电路参数的配置
- 电流PI参数、转速PI参数的调整方法
- 代码中的计算公式
- 数学函数简介

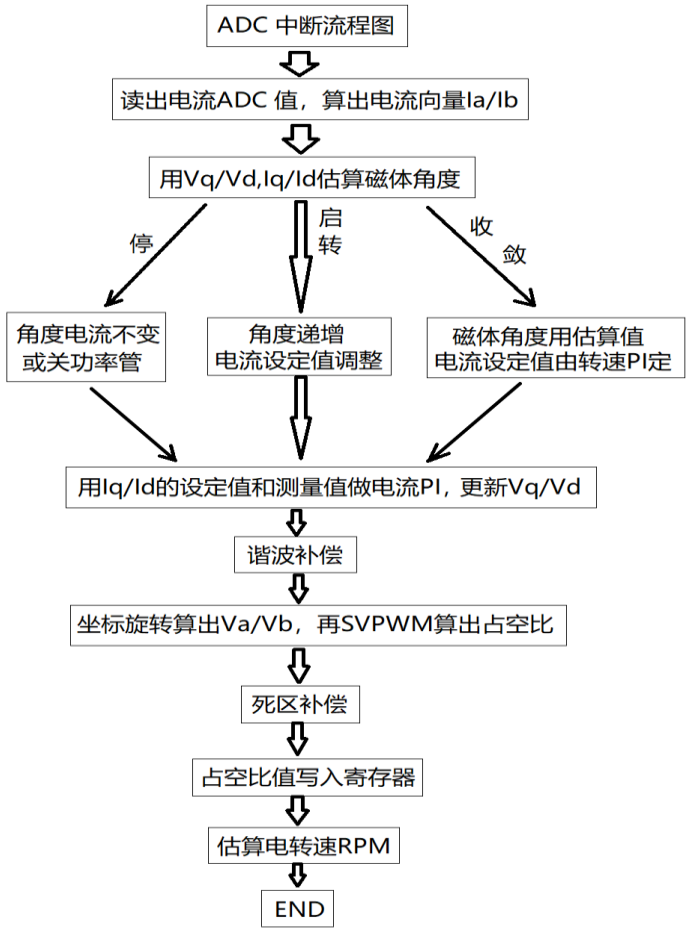
代码架构：main,ADC中断，SysTcik中断

nuvoTon 新唐科技

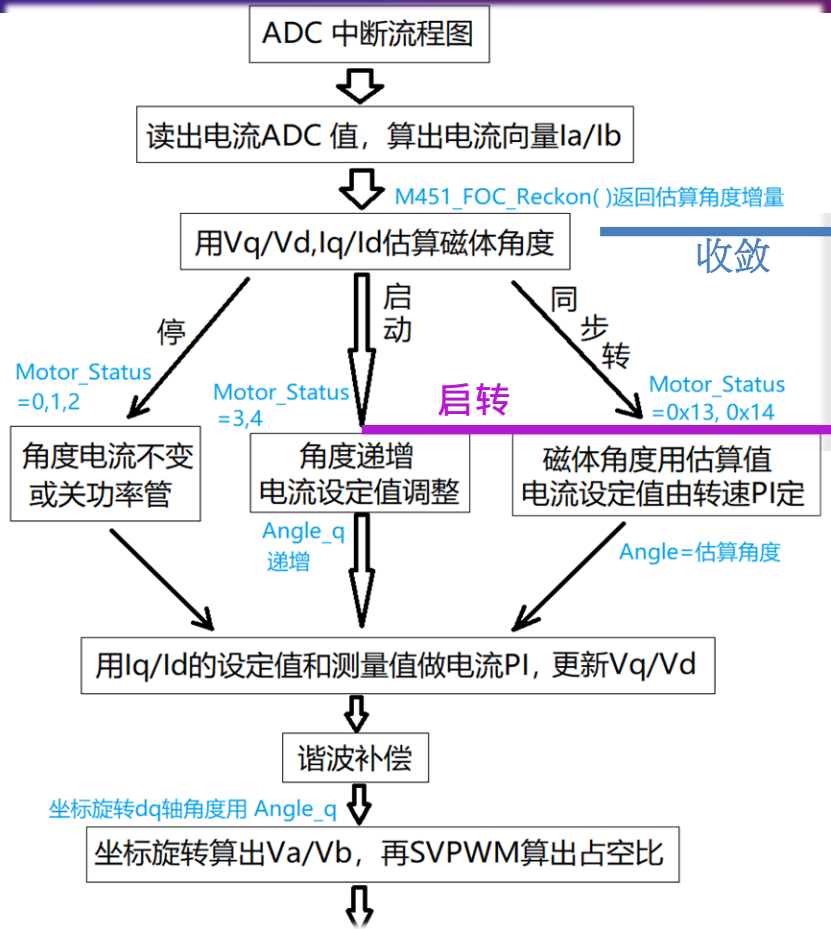
- 代码架构三大部分：main()+两个中断
- Main():控制启转、停转,故障监测及加速度和功率控制等
  - 判断是否欠压、过温、短路，以及处理转速值等
  - 启转：设定转速 RPM_Set 大于最小值,状态0变2再变5,然后启转
  - 停转：设定转速 RPM_Set <= 0,降速停转
- ADC中断：磁体位置估算,电流PI运算,PWM占空比计算
  - ADC中断代码一直在执行，停转时功率管关了，但计算没停止。
  - 坐标变换q轴角度用的是 Angle_q
  - 启转时,代码控制 Angle_q 递增，同步后 Angle_q 用估算值
- Systick中断：转速PI运算，故障显示等

ADC 中断代码流程

- 停转，起转，同步由以下状态机控制
  - 状态0，关功率管，不控制电机
  - 状态1，输出0电压，用于刹车
  - 状态2，下MOS导通50%，用于自举电容充电
  - 状态3，快速起转
  - 状态4，先拉到某角度稳定后再起转
  - 状态5，起转前的锁定，然后去状态3或4
  - 状态6，顺风起转准备，然后去状态4
  - 状态7，啥也不做，用于Debug测试电流值等
  - Bit4=1，同步转动状态（0x13，0x14）
  - 其它，变状态0



ADC 中断代码流程




```
260  if(Motor_Status & 0x10){
261    Angle_q += temp16 ;
262
263  }
264
265  //=== 状态 3 先转后等,  Vout_UpLimit_f17 赋值用于限流
266  else if(Motor_Status == 3){
```

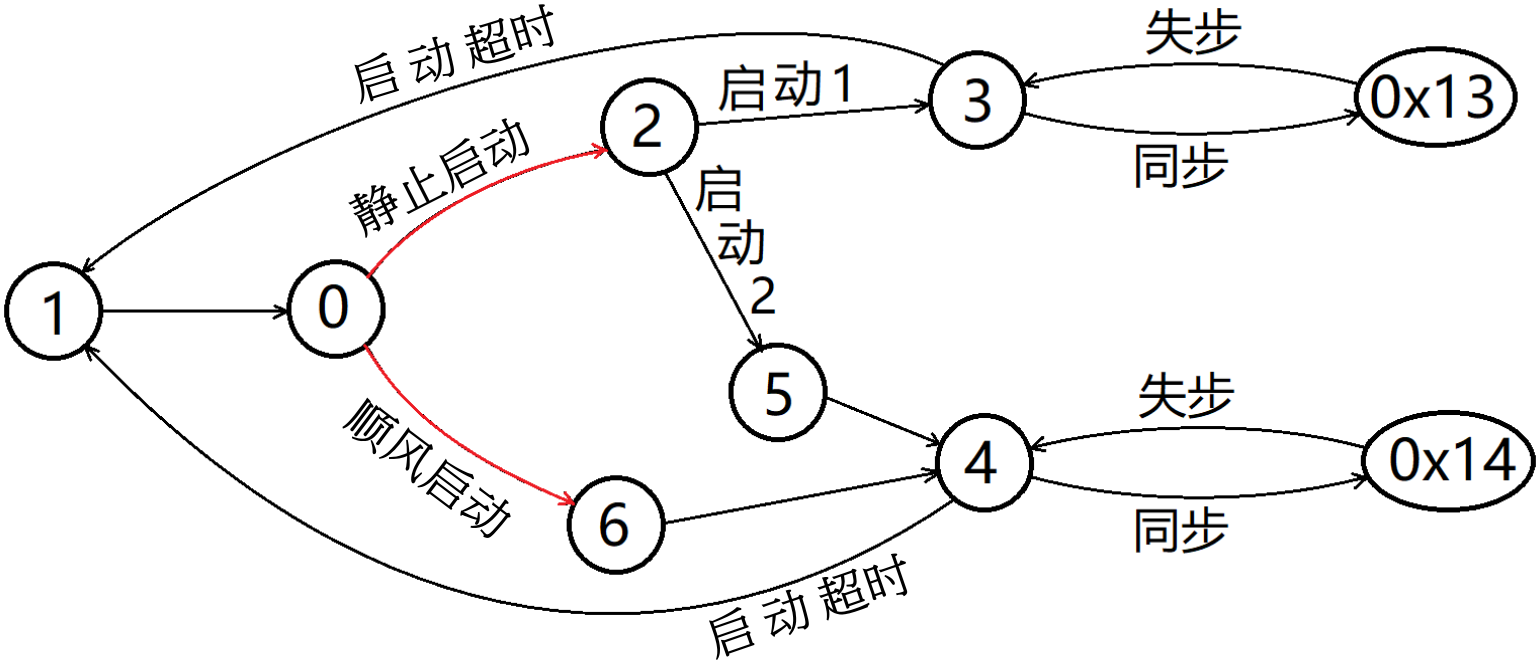
状态机 Motor_Status 控制代码流程

停转时,功率管关了

启动时,启动代码调整角度和电流

同步时,角度用估算值,电流用转速PI 计算结果

ADC中断里的状态机 Motor_Status

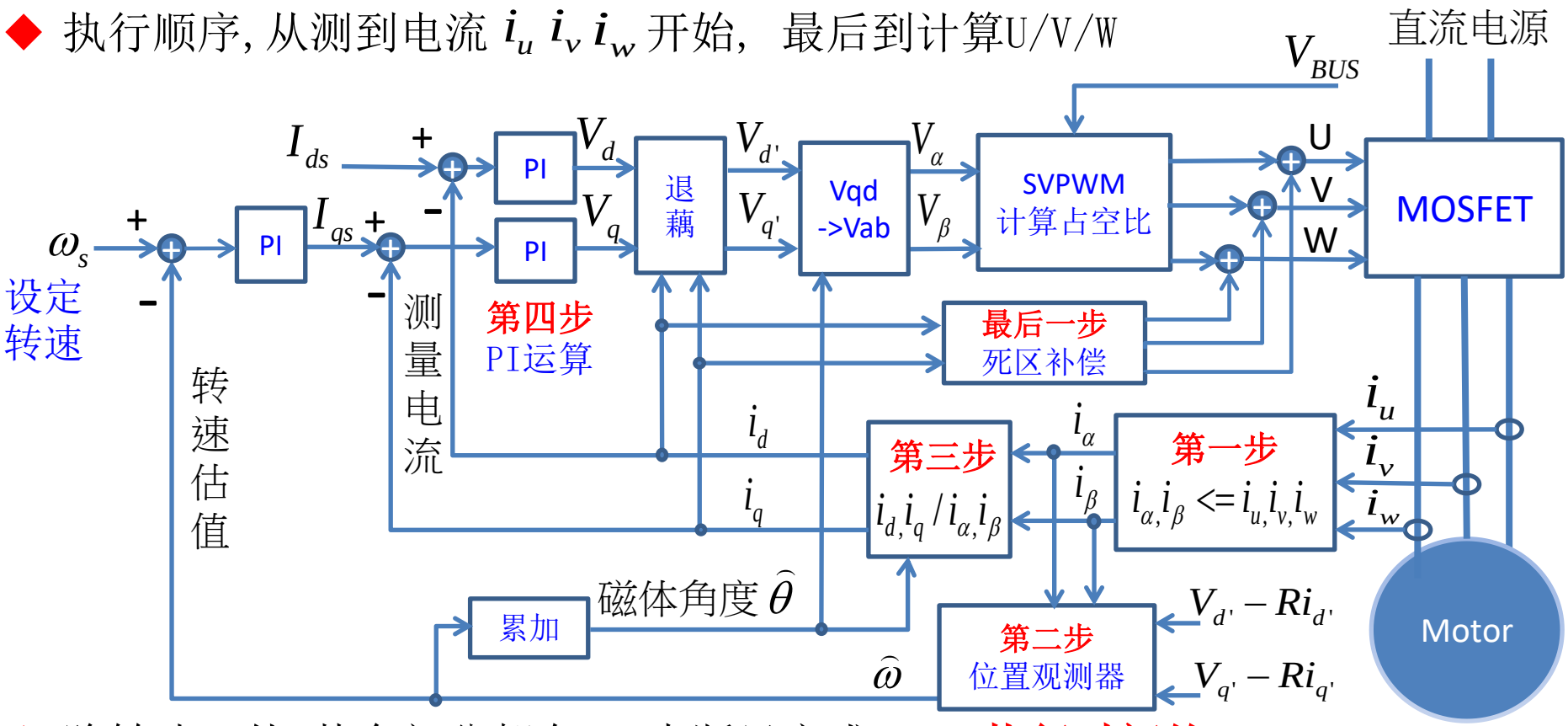


红线在Main()中启动时转变

启动1和启动2, 由宏定义二选一

总体框图

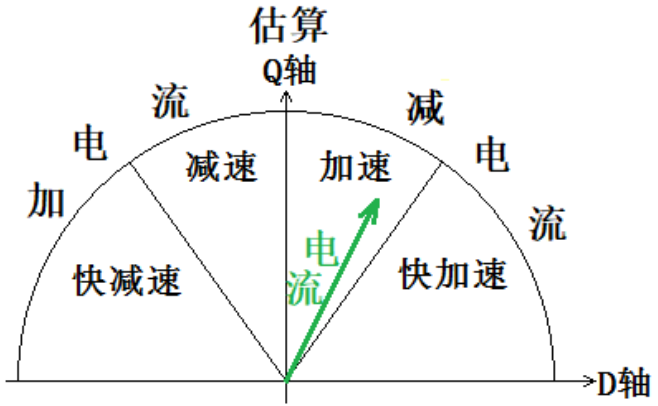
- 执行顺序，从测到电流 i_u i_v i_w 开始， 最后到计算U/V/W

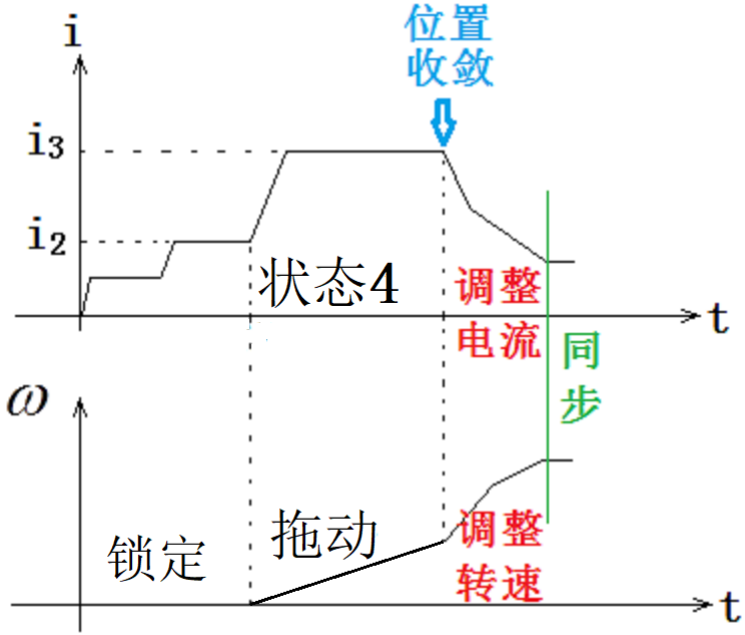


- 除转速PI外，其余部分都在ADC中断里完成，M451**执行时间约20us**

I/F 启转过程简介

- 状态4先匀加速托动，位置收敛后调整转速和电流
  - 锁定过程分俩阶段是考滤轻重载
  - 电流方向(绿)与估算Q轴重合时，变为同步





符号定义

- #Define定义的常量，全大写

```
7  #define  VIN_WORK_MIN_UP          (  13*10)        // 停转时的、最低起转电压，0.1V电压值,所以乘10
8  #define  VIN_WORK_MIN             (  12*10)        // 转动时的电压下限，再低就停转
9  #define  VIN_WORK_MAX             (1600*10)        // 最高工作电压，大于1638.3V 坐标旋转溢出
```

- const常量,首个词全大写,最后一词不能全大写,中间词随意

```
47  int32_t const PULL_Ommega_Min_f16 = ((RPM_ACTIVATE_MIN*65536/PWM_Frequency)<<16)/60;
48  int32_t const PULL_Ommega_Max_f16 = ((RPM_ACTIVATE_MAX*65536/PWM_Frequency)<<16)/60;
49  int32_t const PULL_Ommega_Inc_f16 = PULL_Ommega_Max_f16/(10*PWM_CYCLE_10th_sec);
```

- 某个变量的位变量，变量名 + 全大写位定义名

```
41  INT_EXT uint16_t  Stop_Runing ;                          // 非0就停转
42  #define  Stop_15V_LOW                0x0001              // 15V 电压低
43  #define  Stop_SHORT                  0x0002              // 短路了，发生了 Brake
44  #define  Stop_ACTIVE_FAIL            0x0004              // 多次启动失败
```

- 普通变量，每个词只有首字母大写,中间是否加“_”随意
  - 后缀_f4,_f12,_f16 表示小数位数
  - 电压标准值是0.1V,电流以ADC数据为基准,加小数后缀加 _fxx

PWM 输出极性配置

- 函数PWM0_Init()中，配置PWM引脚输出信号的极性
  - 若PWM024输出高时MOS导通，138行注释掉就可以了

```
137     PWM0->POLCTL = 0x002A ;                          // PWM135 反相, 输出0 MOS导通
138   //  PWM024_Out_0_MOS_On() ;
139     PWM135_Out_1_MOS_On() ;
140     PWM0->MSK    = ShutDown_AllMOS ;
141     PWM0->MSKEN  = 0x003F ;                          // 使能 MASK 功能, 先关断所有输出
142     PWM0->POEN   = 0x003F ;                          // 低6bit输出使能, 但引脚输出值 = ShutDown_AllMOS
```

ADC 配置

nuvoTon 新唐科技

◆ADC引脚转换的先后次序，可任意配置

- 三相电流ADC完成后产生中断,开始计算,计算时后续ADC就完成了

```
23  void ADC_Init(void)
24  {
25      EADC->CTL = EADC_CTL_ADRST_Msk                                  // ADC 电路复位(自动清零)
26                | 0x00070000                                        // 采样时间 8个时钟
27                | EADC_CTL_ADCIEN0_Msk | EADC_CTL_ADCIEN1_Msk       // 使能中断0 和中断1
28                | EADC_CTL_ADCEN_Msk ;                              // Enable EADC
29
30      EADC->SCTL[0] = EADC_PWM0TG0_TRIGGER | ADC_CHANNEL_PWM01 ;          // ADC_PWM01 电流, 零点触发,低4
31      EADC->SCTL[1] = EADC_PWM0TG0_TRIGGER | ADC_CHANNEL_PWM23 ;          // ADC_PWM23 电流
32      EADC->SCTL[2] = EADC_PWM0TG0_TRIGGER | ADC_CHANNEL_PWM45 ;          // ADC_PWM45 电流
33
34      EADC->SCTL[3] = EADC_PWM0TG0_TRIGGER | ADC_CHANNEL_SPEED ;          // Speed, 调速旋钮
35      EADC->SCTL[4] = EADC_PWM0TG0_TRIGGER | ADC_CHANNEL_VBUS ;           // 母线电压
36      EADC->SCTL[5] = EADC_PWM0TG0_TRIGGER | ADC_CHANNEL_TEMPERATURE ;    // 功率管温度
37
38      SYS->VREFCTL    = 0 ;                                   // Vref选择: 0 =外接Vref,0xF=4096mV, 注意与常量定义
39                                                             // ADC参考电压若选片内4096mV,Vref 引脚外接1uF电容,不
40      EADC->STATUS2   = 1 ;                                  // 开中断前先清中断标志
41      EADC->INTSRC[0] = 1ul << 2 ;                           // 模块0,1,2,测完三相电流就进中断开始计算
42
43      NVIC_SetPriority(ADC00_IRQn, 1) ;                      // 优先级=1, 最高优先级0保留急用
44      NVIC_EnableIRQ(ADC00_IRQn) ;                           // 使能ADC 中断
```

电机参数配置

- Main.c中给Ld,Lq赋值
  - 慢慢转电机测两相电感，最大值除2赋值给 Lq，最小值除2 = Ld
  - 或者三条线测两两电感,最大者是 Lq，最小者是 Ld
  - 必须 Lq >= Ld

```
247   Lq_Inductor   = 690 ;                    // 电机电感 uH,  风机
248   Ld_Inductor   = 665 ;                    // 必须 Lq >= Ld
249   Unit_Magnify = CURRENT_Adc1000*25*16 ;   // H=16, 表示电流ADC 值乘16后才用于角度估算
250
```

- 测两相电阻,毫欧值除2后在interrupt.c中赋值给RESISTER_Coil

```
26  #define       RESISTER_COIL         1000                              // 一相线圈电阻,毫欧值
27  int32_t const CURRENT_Adc1000  = 50*4096*1000/ADC_VREF_MV ;             // 1安电流ADC值乘1000的常数(10毫欧5倍=50毫欧)
28  int32_t const CURRENT_0A1_Value_f12 = ((CURRENT_Adc1000<<8)+62)/625; // 0.1A电流ADC值+12位小数(除10000乘4096)
```

电流、电压测量电路参数的配置

- Interrupt_Fun.c中，1A电流的ADC值，再乘1000的数值，赋值给CURRENT_Adc1000
  - DEMO代码按10毫欧运放5倍配置的，Vref=5000mV，所以1A的ADC值是：10毫欧*5倍 *4096/5000mV，再乘1000，算式如下

```
#define       RESISTER_COIL         1000                                  // 一相线圈电阻，毫欧值
int32_t const CURRENT_Adc1000     = 50*4096*1000/ADC_VREF_MV ;    // 1安电流ADC值乘1000的常数(10毫欧5倍=50毫欧)
int32_t const CURRENT_0A1_Value_f12 = ((CURRENT_Adc1000<<8)+62)/625; // 0.1A电流ADC值+12位小数(除10000乘4096)
```

- 母线电压Vbus_0v1是0.1V的电压数值

```
   //母线电压=(分压比(91+10)/10)*ADC_VREF_MV*ADC值/1000/4096，乘10变0.1V值=VREF_MV*(91+10)/10*ADC值/100/4096
  temp32 = (ADC_VREF_MV*(91+10)/10*(EADC->DAT[4]&0xFFFF)/100+2048) >>12;
  Vbus_0v1 = (Vbus_0v1 + temp32) >>1 ;                          // 若电压波动较大(用薄膜电容)，滤波会让电压数据滞后

                                  // 电压上限Vout_Max_ f17 也可等于固定值，比如 350V
  temp32 = Vbus_0v1*200 >>8 ;                                   // 调制比=1是0.577 =150/256, 过调制大于2/3 =172/256
  if(temp32 >16383)temp32 =16383 ;                              // 电压最大14位 = 16383 (1638.3V)
  Vout_Max_f17 = temp32 << 17 ;                                 // Vdq 限幅值加17位小数, 最大到31位
  if(Motor_Status & 0x10)Vout_UpLimit_f17 = Vout_Max_f17;   // 同步后电压上限随着变，注释此句可测启动电压转速
                          // PWM周期=4736，电压上限Vout_Max_ f17 不能超过母线5倍以上，详见SVPWM()头文件说明
```

电流PI参数的调整方法

- 电流P参数=ωL，L是电机电感，低通滤波频率转折点 ω 一般取PWM 频率的 15~100分之一
  - 启转时若偶有滋滋声，电流快速波动，就是PI参数过大了，取值小点再测。

```
// Kp = 低通带宽*L = (2*3.1415*F/N)*L,  F 是PWM频率, N 取值10~100, 为便于计算取值 10*3.1415 得  Kp = F*L/5,
// Ki <= 低通带宽*R, 即 Ki <= Kp*R/L, 为计算方便,把乘周期 t 先加上,得 Ki <= Kp*R/(L*F)

// 单位换算:
// 1A电流的ADC 值是 M =(A*r*4096)/5V, 电流 =I 时 ADC 值是 M*I, 做电流 PI 时再增5位小数, 电流数据是 32M*I
// 比例运算是 V=Kp*I,得17位小数的0.1V电压值, 公式两边乘(10*2^17)得 (10*2^17)*V = (10*2^17)*Kp/32M * (32M*I)
// 故电流PI时电流数据32M*I前面的系数 Kp_Current =10*2^17*Kp/32M=(10/M)*Kp*(2^12) 代入M值=Kp*50/(A*r),再代入Kp=F*L/5得

// Kp_Current  = 10*F*L / (A*r)                    F =PWM频率, L =电机电感, A =运放倍数, r =电流采样电阻
// Ki_Current <=   Kp*R / (L*F) = 10R/(A*r)

// 本例 L=0.7mH,F=15kHz,R=1欧,A*r=0.05欧, 算得 2100, 200, (实际取值可适当增大或减小,看实际效果)
int32_t volatile Kp_Current = 1000,   Ki_Current = 60;     // 电流5位小数做PI,输出17位小数电压,所以此值有12位小数
```

转速PI参数的调整方法

- 按转速 Ω 与转动惯量 J 与转矩 M 的关系，理论上比例系数应取值 ωJ 。积分系数一般取比例系数的1/20~1/100

$$J\frac{d\Omega}{dt}+B\Omega=M$$

- 实际应用转动惯量和转矩常常是变化的。简单点可以预估一个小一点的 PI 系数，测试轻重载效果。若觉得转速响应太慢，可适当增大后再测试，只要调速时转速不出现忽快忽慢的波动就可以

```
79  int32_t static volatile Kp_Speed =200,   Ki_Speed =2;      // 转速PI运算输出12位小数电流设定值,相当于此有12位小数
```

起转电流配置

- 起转时最大电流可先取正常转动时最大电流的一半左右，再按起转效果增减。
- 启转电压的上限，取值参考额定电压的15%左右

```
// 最大电流,最小电流设定值
#define       CURRENT_MIN           (30<<7)                                  // 做电流 PI 时会先右移7位，做小数对齐
int32_t const CURRENT_Min_f12 = -CURRENT_MIN;                                // 负值防0值下漂停不下来,负多了降速时充母线
int32_t const CURRENT_Max_f12 = 45*CURRENT_0A1_Value_f12+(30<<7); // 转动时电流峰值上限，加一点显示时高位不跳
int32_t const CURRENT_Active_f12  =  CURRENT_Max_f12*3/4 ;                   // 启动时最大电流，可比上限小一点
int32_t const CURRENT_Increase_f12 = CURRENT_Active_f12 >>4 ;                // 状态3启动时的电流增量,取值让状态5两倍设定值刚好转动
```

起转加速度配置

- 拖动转速下限PULL_Ommega_Min_16,比能同步的最低转速稍低
- 加速度PULL_Ommega_Inc_f16 先小点,磁铁能跟着转,后期再调整
- 拖动转速上限，可先取下限的 2~4倍
  - 一圈角度360 用 65536 表示
  - 这里转速是一个PWM 时间转过的角度,数值很小,所以又加了16位小数

```
43  #define  SECTION_RPM_AVERAGE        (6*POLE_PAIR)                  // 转速在这些扇区内求平均，取一个机械周期的扇区数
44  #define  RPM_ACTIVATE_MIN           (300*POLE_PAIR)                // 拖动转速下限,稍低于能稳定转的最低转速
45  #define  RPM_ACTIVATE_MAX           (RPM_ACTIVATE_MIN*3)           // 拖动转速上限,高于能稳定的最低转速,2~3倍下限值
46
47  int32_t const PULL_Ommega_Min_f16 = ((RPM_ACTIVATE_MIN*65536/PWM_Frequency)<<16)/60;  // 转速下限必须 >= (1<<17)
48  int32_t const PULL_Ommega_Max_f16 = ((RPM_ACTIVATE_MAX*65536/PWM_Frequency)<<16)/60;  // 每次PWM的角度增量上限
49  int32_t const PULL_Ommega_Inc_f16 = PULL_Ommega_Max_f16/(10*PWM_CYCLE_10th_sec);      // 启转加速度,转速增量,尽量慢
```

代码中的计算公式

- 电压、电流、电感计算单位
  - 若1A的ADC值是M，则电流i的ADC值是Mi
  - $V - Ri = Ldi/dt + \varepsilon$ 右端分子分母乘1000000M，得

$$V - Ri = \frac{1000000Ld(Mi)/dt + 1000000M\varepsilon}{1000M * 25 * 40}$$

  - 令电流ADC值Mi=i'，微亨值L'，电势变ε'，公式变为

$$\text{输入}(10V - 10Ri) * 4 = \frac{L'di'/dt + \varepsilon'}{1000M * 25} \text{ 估算}$$

电压0.1V的数值，再乘4用于运算，**电感用微亨值，电流用ADC值，**
常数1000M*25赋给全局变量 Unit_Magnify

母线电压不能超过 65535/40=1638.3V，否则坐标旋转时32位溢出

若LI超32位，电感值要除以10或100， 常量Unit_Magnify也除相应值即可

M451适于电机控制的特性

- 72MHz Cortex_M4内核
  - 5V工作电压,-40~105度工作温度范围
  - 40~256K FLASH 取指令0等待
  - RAM 16/32K，带硬件校验功能
- CAN 2.0 接口(M453型号)
- 96位唯一序列号用于代码加密
- 12位 DAC 实时输出中间变量观察数值变化
- 其它:WDT,UART,SPI,I2C,RTC,EBI,USB-OTG,PDMA 等

数学函数简介

- uint32_t Get_SquareRoot(uint32_t Data)开方函数。如果实参太小,可对实参左移2,4,6,8后再开方,再把结果右移1,2,3,4……
- int16_t Get_Arctan(int64_t Xx, int64_t Yy)求反正切，返回-32768~32767 [-180~180度)
- int64_t Updata_Theta_Value(int32_t V_q,int32_t V_d,int32_t I_q,int32_t I_d)
  做一次观测器迭代运算,对结果不做准确性判断。更新了偏差Theta_e和磁铁角度估值Estimate_Q_Position，返回值是反电势除Unit_Magnify后的值,返回值小数位与电压实参一样。
- Set_PWM_Frequency_LPF(uint32_t PWM_F,uint32_t LPF) 配置迭代运算频率和低通角频率。

用dq轴反电势求反正切前，先做一阶低通运算,减小数据波动

$$y_n = \frac{y_{n-1} f + x_n \omega}{f + \omega}$$

nuvoTon

Thank You !

更多资料见新唐论坛 www.nuvoton-mcu.com