



新唐 M451 无感 FOC 方案代码说明

www.nuvoton.com



nuvoTon

- ◆ 代码总体架构
- ◆ 起转、停转的代码流程
- ◆ 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

◆代码架构三大部分：main()+两个中断

◆Main()：故障监测, 控制启转、停转等

- 判断是否欠压、过温, 调速旋钮值是多少等等
- 启转：设定转速 RPM_Set 大于最小值, 状态0变2
- 停转：设定转速 RPM_Set ≤ 0 , 启动时又不想启动了就变状态1

◆ADC中断：磁体位置估算, 电流PI运算, PWM占空比计算、输出

- 坐标变换q轴角度总是用 Angle_q
- 启转时代码控制 Angle_q 递增, 控制电流设定值
- 同步后 Angle_q 用估算值, 电流设定值用转速 PI 运算的结果

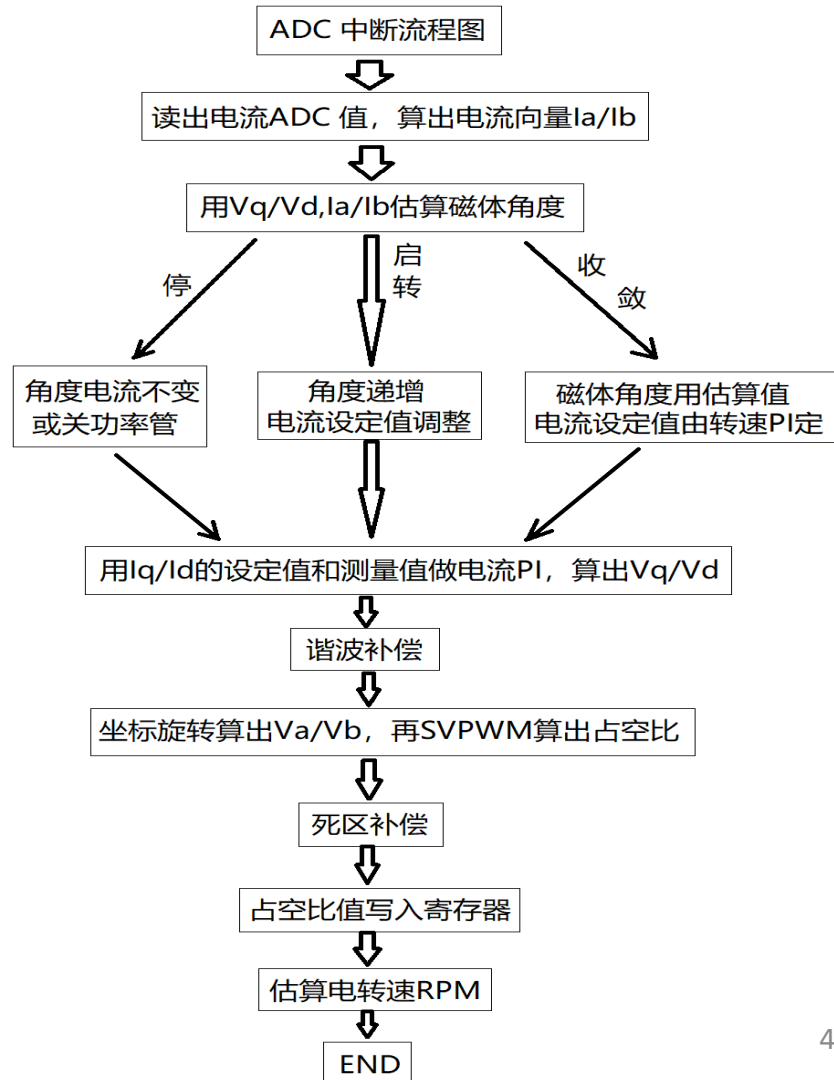
◆Sysstick中断：转速PI运算, 功率计算, 故障显示等

- 按转速计算出电流设定值, 提供给 ADC 中断里电流 PI 计算

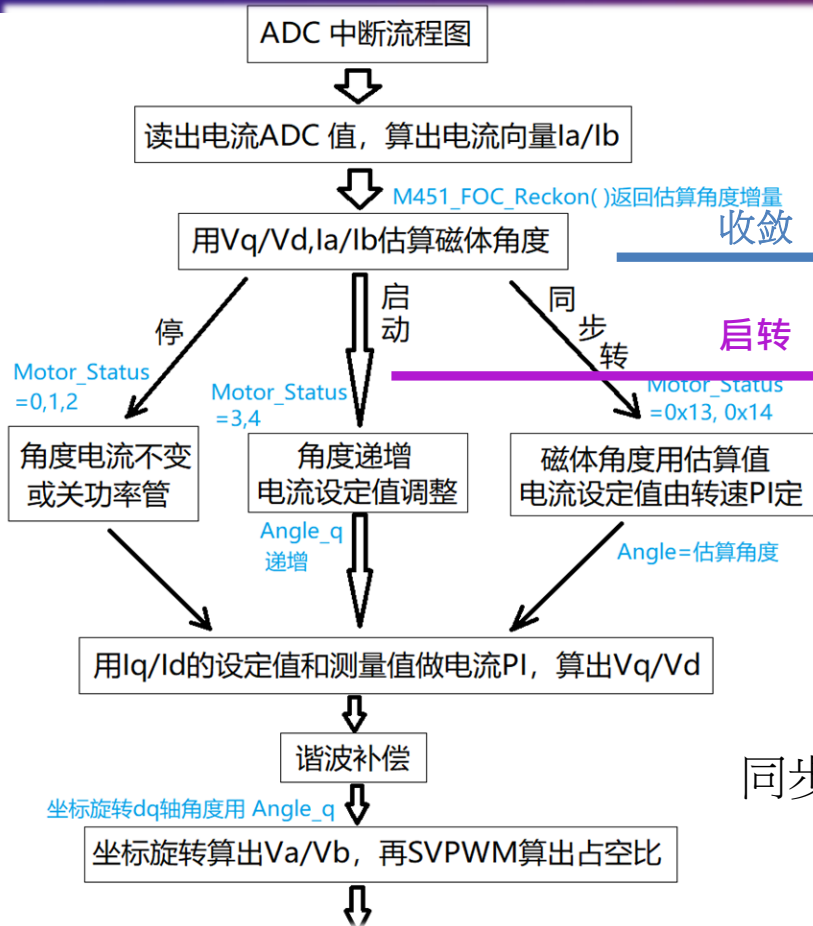
ADC 中断代码流程

◆ 起转, 停转, 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, 只下MOS输出, 用于自举电容充电
- 状态3, V/F 起转
- 状态4, I/F 起转
- 状态5, 起转前的锁定, 然后去状态4
- 状态6, 顺风起转准备, 然后去状态4
- 状态7, 啥也不做, 用于调试
- Bit4=1, 同步转动状态 (0x13, 0x14)
- 其它, 变状态0



ADC 中断代码流程



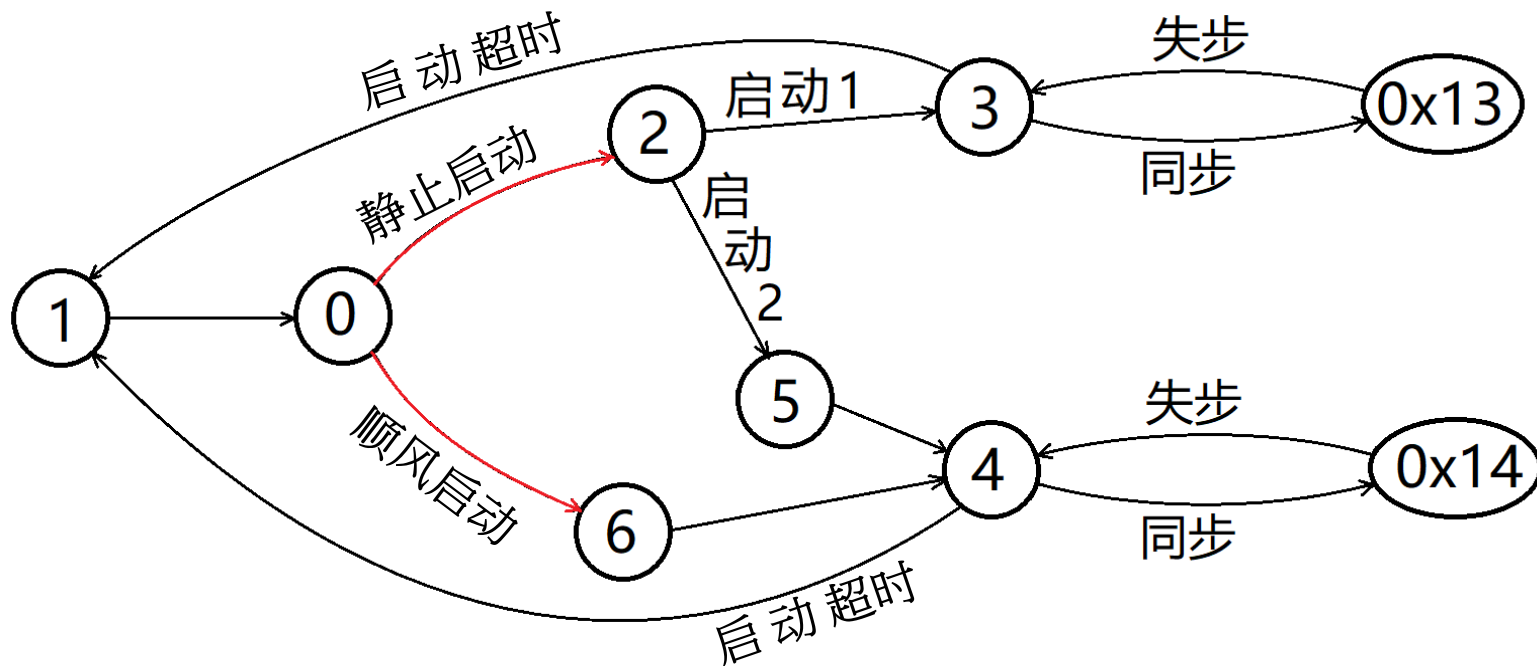
```
218 //=====
218 // 接下来, 按 Motor_Status 不同的值, 执行不同代码
219 //=====
220
221 //Pwm_Duty_Uplimit = gPWM_Period -500 ;
222
223 if(Motor_Status & 0x10){ Angle_q += temp16 ;
224 }
225
226 //==== 状态3 电压拖动起转: 每转一扇区限压值Emf_ADC_Max :
227 else if(Motor_Status == 3){
228
229
```

状态机 Motor_Status 控制代码流程

停转时,功率管关了

启动时,启动代码调速角度、电流和电压

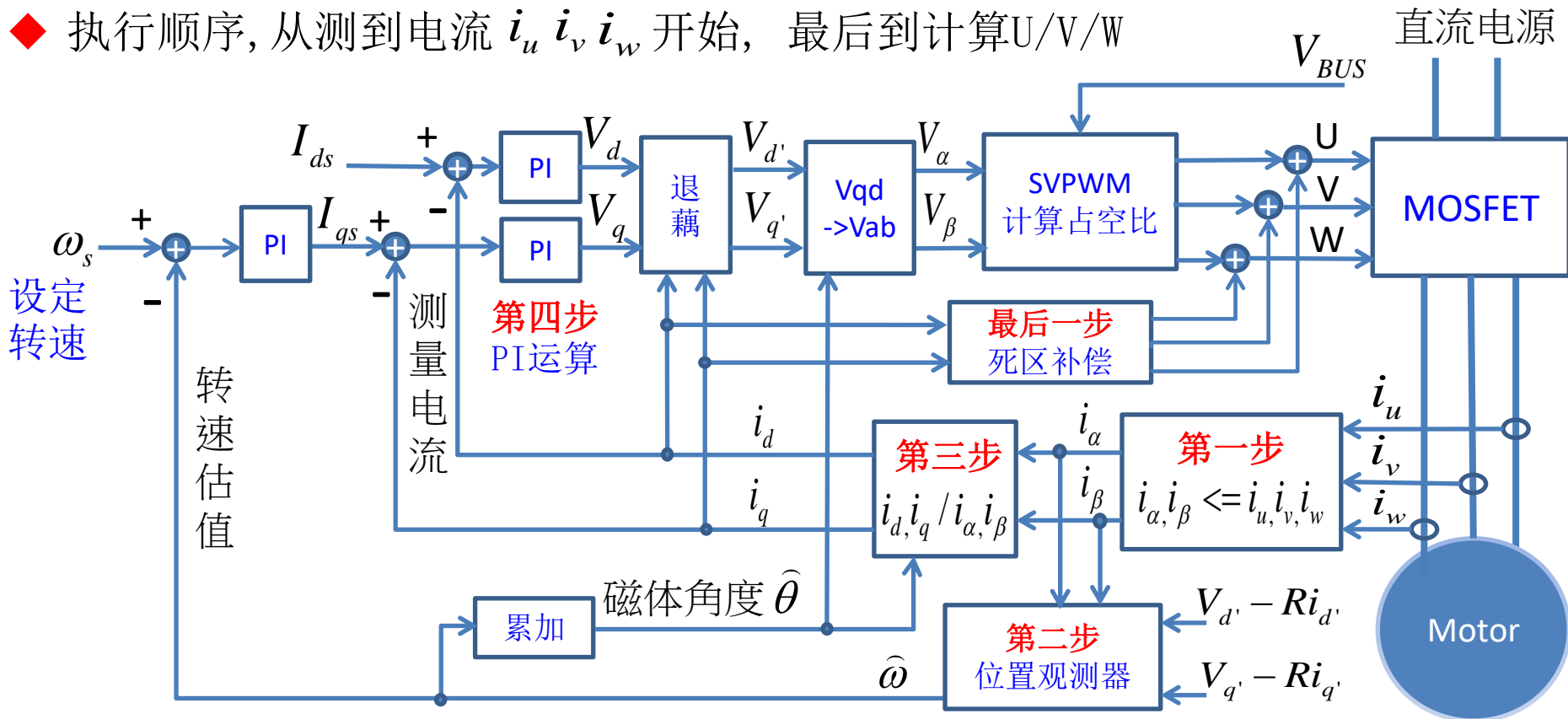
同步转时,角度用估算值,电流用SysTick中断里PI 计算值



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**

起转、停转的代码流程

- ◆ 1, main() 中读调速旋钮ADC值, 算出转速并限定在最大最小值之间后, 赋值给转速设定值变量RPM_Set, 此值大于转速下限, 就会起转

```
395 //== 调速旋钮, 获得转速设定值 =====
396 temp32 = ((uint16_t)EADC->DAT[3]);
397 Speed_Adc = (temp32 + Speed_Adc*31 + 16) >>5; // 低通, 可稳定旋钮 ADC 数据
398 temp32 = Speed_Adc;
399
400 RpmMechanical_Set = temp32; // 机械转速设定值, 注释此句仿真可实时赋值
401
402 temp32 = RpmMechanical_Set * POLE_PAIR; // 电转速 = 机械转速*极对数
403
404 //=== 把电转速设定新值temp32, 限定在上下限之内 =====
405 if(Rotation_Direction != Expect_Direction) temp32 = RPM_STOP_VALUE; // 设定转向与实际转向不同, 停转
406 else if( Stop_Runing ) temp32 = RPM_STOP_VALUE; // "停转变量"非0, 转速 = 停转值
407 else if(temp32 > RPM_SET_MAX) temp32 = RPM_SET_MAX; // 设定转速超上限, 算上限
408
409 if(temp32 < RPM_SET_MIN){
410     if(RPM_Set <= 0){ // 未转时
411         if(temp32 < (RPM_SET_MIN*3 >>2)) temp32 = RPM_STOP_VALUE; // 大于下限值的 3/4 才起转
412         else temp32 = RPM_SET_MIN;
413     }
414     else if(temp32 < (RPM_SET_MIN >>1)) temp32 = RPM_STOP_VALUE; // 在转动时, 小于下限值1/2, 就停转
415     else temp32 = RPM_SET_MIN;
416 }
417
418 // 若限最大功率(或平均电流), 转速设定值RPM_Set不能直接赋值, 要加个PI 运算, 或慢慢加减
419
420 RPM_Set = temp32; // 电转速设定值 = 最小~最大间, 停转时=负值
```


起转、停转的代码流程

- ◆ 2, 若在同步转, 第1步仅是更新转速设定值RPM_Set (318行)。
- ◆ 3, 若不是在同步转, 状态2, 3, 4, 5, 6是已在起动, main() 中不必做什么 (361行), 若是在状态0, 就要在状态0执行次数=0时, 开始起动(362行)。
- ◆ 4, 起动前先测是否在顺风转, 然后就跳转到状态2给自举电容充电后起动

```
317 //===== 在同步转 =====
318 if((Motor_Status & 0x10) == 0x10){
319
320     if((RPM_Set <=0)
321         &&((RPM_Measure*65536/60/PWM_Frequency) <(PULL_Ommege_Min_f16>>16))) {
322         Exec_Counter = 3*PWM_10th_sec; Motor_Status = 1; // 转速低了就停吧, 防1
323     }
324
325 }
326 //===== 未同步(停转或在起转), 且设定转速 RPM_Set 小于0, 不需要启转 ==
327 else if(RPM_Set <= 0){
328
329     //===== 未同步, 且设定转速 RPM_Set 大于0, 需要启转 =====
330     // 状态2, 3, 4, 5, 6是在起转, 只有状态0不自动变其它状态
331     else{
332         if(Motor_Status + Exec_Counter ==0){
333
334             #if 1
335                 temp32 = Test_Period(); // 若在顺风转, 此函数在 Angle_q ==-16384 时刻返回转速
336
337             #else
338                 temp32=0; Emf_ADC_Max=0; // 水泵无顺风转, 不执行Test_Period()可早点起转
339
340             #endif
341         }
342     }
343 }
```

起转、停转的代码流程

- ◆ 5, 在ADC中断里, 状态2执行次数=0后, 因RPM_Set>0, 状态变为3(386行)。当然在变状态3之前, 要做起动前的准备工作(371行开始)。

```
366 if(Exec_Counter) --Exec_Counter ; // 执行计数非0 就递减
367 else if(Motor_Status != 0){
368     if(RPM_Set < RPM_STOP_VALUE) Motor_Status = 0 ; // 转速设定值比停转值还负, 是在给自举电容充电
369     else if(RPM_Set <=0) Motor_Status =MOTOR_STOP_STATUS; // 这个停转状态可以是0, 也可以是5
370     else{ // LED_LIGHT_0() ;
371         #ifdef __IF_ACTIVE_MODE // 定义了IF 起转方式
372             Exec_Counter = TIME_MotorLock ;
373             Motor_Status = 5 ; // 状态5 会跳转到状态4 I/F起转
374
375             #else
376             Iq_set_f12 = CURRENT_Max_f12 ; // 这个是状态3起转时的限流值
377             Emf_ADC_Max = 10 <<17; // 初始电压值让轻载能转起来, 并且不会触发短路保护
378                                     // 状态3 Emf_ADC_Max逐步加到VOLTAGE_UpLimit_while Active_f17
379             Angle_q = -16384 -65536; // 负角度线性加速, 从-90度起转A相电流刚好从0开始上升
380             Pull_Omega_f16 = 0 ;
381             Init_Variable(Angle_q, 0, 0); // 三个参数是: 当前角度、转速和反电势
382             ptrSectionClear = 0 ; // 赋0值让Motor_Active()清0静态变量(函数内赋非0值)
383
384             Do_Spd_PI = -SECTION_RPM_AVERAGE -2; // 计算转速时加1, 一个机械周期后加到正值, SysTick中断里才做转速PI
385             Exec_Counter = 3 ; // 在状态3, 4, 此值非0 被认为是刚起转
386             Motor_Status = 3 ; // 下个PWM周期变状态3
387             //Angle_Jump = 0 ;
388         #endif
389     }
390 }
```

起转、停转的代码流程

- ◆ 6, 状态3是V/F 拖动起动的。在状态2也可以跳转到状态5(再变状态4)开始I/F起动过程
- ◆ 7, 状态3起动成功后, 状态码变 0x13, 开始同步转动。

```
238 if((Motor_Status & 0x10) == 0){ // 函数 Motor_Active() 未切同步
239     Circle_Tick = Angle_q; // 起转时 Angle_q 是负值, 就会先执行到此
240     if(RPM_Set <= 0){ // 指令停转
241         Exec_Counter = 3*PWM_10th_sec; Motor_Status = 1; // 刹车, 尽快停转
242     }
243     else if((Angle_q > 65536*20) || (Exec_Counter == 0)){ // N 圈起转不成功, 或同步转后又失步执行到此
244         Exec_Counter = 9*PWM_10th_sec; Motor_Status = 0; // 去状态0休息一下, 功率管需要散热
245     }
246     else if(LastptrSection != ptrSectionClear){ LastptrSection = ptrSectionClear; // 转到了下一个扇区
247         if(Emf_ADC_Max < VOLTAGE_UpLimit_while_Active_f17) Emf_ADC_Max += 2<<17; // 起转电压增加到上限
248     } // 顺风转时限压值(反电势)或高于此值, 不能减, 减会降速
249 }
```

- ◆ 8, 同步转时, 若转速设定值RPM_Set<=0, 实际转速低到无法同步时, 状态码0x13变成0x03, 又回到起动状态, 而这时会因Exec_Counter=0而变到停转状态0(243行)。

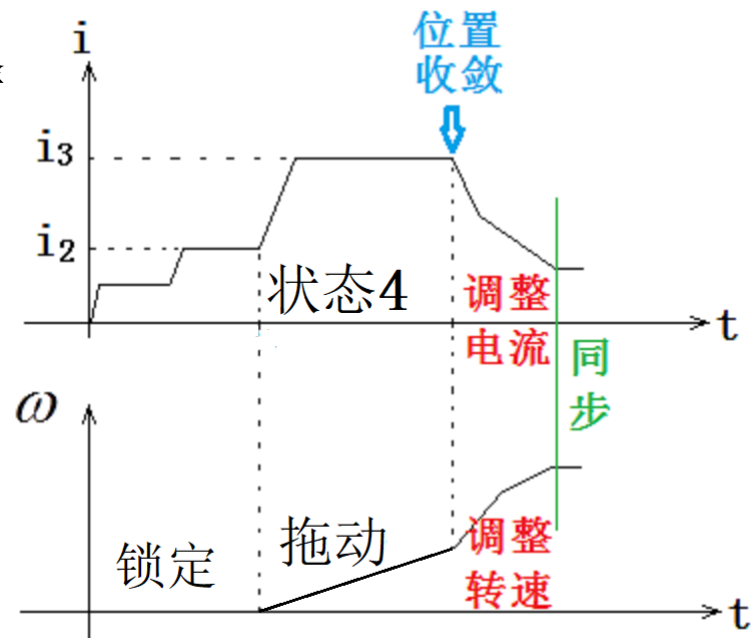
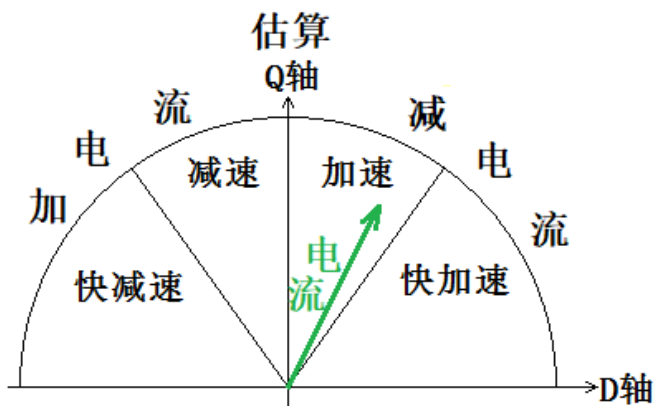
起转、停转的代码流程

- ◆ 9, 同步转时, 若RPM_Set<0, 基本都会转速降到观测器不收敛, 又回到状态3(或状态4)。但理论上不排除迭代运算一直收敛的情况。所以在main()里, 320行, 增加一个判断, 若转速已很低, 直接变状态1刹车尽快停转, 这里也可以变状态0慢慢停转。

```
317 //===== 在同步转 =====
318 if((Motor_Status & 0x10) == 0x10){
319
320     if((RPM_Set <=0)
321         &&((RPM_Measure*65536/60/PWM_Frequency) <(PULL_Ommega_Min_f16>>16))) {
322         Exec_Counter = 3*PWM_10th_sec; Motor_Status = 1; // 转速低了就停吧, 防1
323     }
324
325 }
326 //===== 未同步(停转或在起转), 且设定转速 RPM_Set 小于0, 不需要启转 ==
327 else if(RPM_Set <= 0){
    . . . . .
}
```

◆ 状态4先匀加速托动，位置收敛后调整转速和电流

- 电流方向(绿)与估算Q轴重合时，变为同步
- 锁定过程分俩阶段是考虑轻重载
- 启动时限压, 启动成功后电流自然就会下降



◆ #Define定义的常量，全大写

```
8  #define VIN_WORK_MIN      ( 11*10)           // 工作电压下限, 0.1V电压值, 所以乘10
9  #define VIN_WORK_MIN_UP   ( 13*10)           // 最低工作电压, 与上面值形成回差
10 #define VIN_WORK_MAX      (1600*10)          // 最高工作电压, 大于1638.3V坐标旋转溢出
```

◆ const前缀常量，第一个单词全大写, 最后一个词小写

```
40
41 int32_t const CURRENT_MotorLock_f12 = CURRENT_Max_f12*3 >>5 ;    // 锁定时电流值
42 int32_t const VOLTAGE_UpLimit_while_Active_f17 = 30 << 17 ;      // 起转时的电压上限,;
```

◆ 某个变量的位变量，变量名 + 全大写位定义名

```
44 INT_EXT uint16_t volatile Stop_Runing ;           // 非0就停转
45 #define Stop_15V_LOW      0x0001                 // 15V 电压低
46 #define Stop_TEMPERATURE_OVER 0x0002             // 超温
47 #define Stop_SHORT        0x0004                 // 短路了
48 #define Stop_OPEN         0x0008                 // 缺相
```

◆ 多个字母定义的普通变量，只有首字母大写

- 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

- ◆ 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，124行注释掉就可以了

```
124 // PWM024_Out_0_MOS_On() ;  
125 PWM135_Out_1_MOS_On() ;
```


◆ 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

```
23 void ADC_Init(void)
24 {
25     EADC->SCTL[0] = EADC_PWMOTGO_TRIGGER | 1 ;           //PWM0 立即触发, 低4位是通道号, ADC_PWM01
26     EADC->SCTL[1] = EADC_PWMOTGO_TRIGGER | 2 ;           // ADC_PWM23
27     EADC->SCTL[2] = EADC_PWMOTGO_TRIGGER | 3 ;           // ADC_PWM45
28     EADC->SCTL[3] = EADC_PWMOTGO_TRIGGER | 4 ;           // Speed
29     EADC->SCTL[4] = EADC_PWMOTGO_TRIGGER | 15 ;          // VBUS
30
31     EADC->SCTL[5] = EADC_SOFTWARE_TRIGGER | 6 ;          // 软件触发ADC测初始位置
32
33     EADC->STATUS2 = 1 ;
34     EADC->INTSRC[0] = 1ul << 2 ;
35 }
```

↑
ADC通道号

↑
SCTL[2]转换完产生中断

// 清中断标志
// 使能模块2触发ADC中断0

◆ 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
- 必须 $L_q \geq L_d$

```
242 #if 1
243     Lq_Inductor = 690 ;           // 电机电感 uH, 风机
244     Ld_Inductor = 665 ;           // 必须  $L_q \geq L_d$ 
245     Unit_Magnify = CURRENT_Adc1000*25*8 ; // H=8, 电流ADC值乘8后才用于计算
246
247 #else
248     Lq_Inductor = 690 >>3;        // 建议LqLq最大29位: 0x1FFF_FFFF
249     Ld_Inductor = 665 >>3;        // 电感太大若LqLq会溢出(32位符号
250     Unit_Magnify = CURRENT_Adc1000*25 ; // H=8 与除8 抵消
251
252 #endif                               //电流ADC出12位符号数, 乘8后15位, Lq超32.'
```

◆ 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*4096/5 ; // 1安电流ADC值乘1000的常数值(10毫欧5倍)
28 int32_t const CURRENT_1A_Value_f12 = (CURRENT_Adc1000<<9)/125; // 12位小数的 1A 电流值常量 (应除1000乘4096)
```

◆ 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, $V_{ref}=5V$, 所以1A的ADC值是 $10\text{毫欧} \times 5 \times 4096 / 5V$, 再乘1000, 算式如下

```
26 #define RESISTER_COIL 1000 // 一相线圈电阻, 毫欧值
27 int32_t const CURRENT_Adc1000 = 50*4096/5; // 1安电流ADC值乘1000的常数值(10毫欧5倍)
28 int32_t const CURRENT_1A_Value_f12 = (CURRENT_Adc1000<<9)/125; // 12位小数的 1A 电流值常量 (应除1000乘4096)
```

◆ Vbus_0v1是0.1V的数值

```
458 // 电压=(分压比(91+10)/10)*5V*ADC值/4096,再乘10变0.1V值 =50*(91+10)/10*ADC值/4096
459 temp32 = (50*(91+10)/10*(EADC->DAT[4]&0xFFFF) +1024) >>11; // 少右移一位, 2x
460 temp32 = temp32 + (Vbus_0v1 <<1); // (2x+2y), 25mV数值
461 Vbus_0v1 = (temp32 +2) >>2; // 母线电压0.1V的数值, (2y+2x)/4
462 Section = M4_Svpwm(&temp_Vd, &temp_Vq, &temp32, Pwm_Duty_Uplimit); // 用25mV数值计算PWM 占空比、低电平时间
```

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 $20 \sim 100$ 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。

```
44 // 1A电流的ADC 值是 M =(A*r*4096)/5V, 电流 I 的ADC 值是 MI, 做电流 PI 时再增5位小数是 32MI
45 // 32MI 做比例运算V=Kp*I, 得17位小数的0.1V电压值, 所以比例运算是: (10*2^17)*V = [(10*2^17)*Kp/32M] * (32MI)
46 // 所以电流数据 32MI 前面的系数 Kp_Current = 10*2^17*Kp/32M = (10/M)*Kp*(2^12) = 50Kp/(Ar)
47
48 // Kp 取值 2*3.1415*F*L/N, N 取值 =10*3.1415 得 Kp =F*L/5, 代入 Kp_Current公式得
49 // Kp_Current = 10*F*L/(Ar), F=PI运算频率, L=电机电感, A=运放倍数, r=电流采样电阻
50
51 // 积分系数Ki/R <=Kp/L 阶跃响应无上冲, 若加大积分参数, 响应会加快, 但转速调整时会有点过冲
52 // Ki_Current =Ki*T =Kp*R/(F*L) =10R/(Ar), R=电机电阻, A=运放倍数, r=电流采样电阻
53
54 // PI运算:电流ADC值先增5位小数, PI运算结果是17位小数的0.1V数值, 从数值上看是 PI 运算让数据又增加了12位小数
55
56 int32_t    Kp_Current = 1500,  Ki_Current = 50;    // PI前加5位小数, PI结果是17位小数, 所以此值相当于有12位小数
57 int32_t    Kp_Speed   = 200,   Ki_Speed   = 3;    // 转速PI运算结果是12位小数的电流设定值, 相当于此有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 $1/20 \sim 1/100$

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

- ◆ 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

```
54 // PI运算:电流ADC值先增5位小数,PI运算结果是17位小数的0.1V数值,从数值上看是 PI 运算让数据又增加了12位小数
55
56 int32_t      Kp_Current = 1500,  Ki_Current = 50;    // PI前加5位小数,PI结果是17位小数,所以此值相当于有12位小数
57 int32_t      Kp_Speed   = 200,   Ki_Speed   = 3;     // 转速PI运算结果是12位小数的电流设定值,相当于此有12位小数
```

- ◆ 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。

```
36 // 12位小数设定值 =CURRENT_Adc1000 就是=1000A/4096= 约1/4安, ADC=1时, 电流=1/CURRENT_Adc1000 毫安
37 #define CURRENT_MIN (30<<7) // 做电流 PI 时会先右移7位, 做小数对齐
38 int32_t const CURRENT_Min_f12 = -CURRENT_MIN; // 下限负值防0值下漂停不下来, 负多了降速充母线
39 int32_t const CURRENT_Max_f12 = 4*CURRENT_1A_Value_f12 +(30<<7); // 正常转动时电流上限, 加一点显示时高位不跳动
40
41 int32_t const CURRENT_MotorLock_f12 = CURRENT_Max_f12*3 >>5 ; // 锁定时电流值
42 int32_t const VOLTAGE_UpLimit_while_Active_f17 = 30 << 17 ; // 起转时的电压上限, 有限流并防转速上冲的作用
43
```

- ◆ 拖动转速上限PULL_Ommega_Max_16, 参考能同步的最低转速
- ◆ 加速度PULL_Ommega_Inc_f16 先小点, 磁铁能跟着转, 后期再调整
- ◆ 拖动转速下限, 可先取上限的 $\frac{1}{2} \sim \frac{1}{4}$, 后期再调整
 - 一圈角度360 用 65536 表示, 加16位小数就是低16位成了小数部分

```
60
61 #define RPM_ACTIVATE_MAX          (600*POLE_PAIR)          // 拖动转速Pull_Ommega_f16上限,参考能同步的最低转速
62 #define RPM_ACTIVATE_MIN          (300*POLE_PAIR)          // 拖动转速下限
63
64 int32_t const PULL_Ommega_Max_f16 = ((RPM_ACTIVATE_MAX*65536/60/PWM_Frequency)<<16); // 拖动上限变成角度增量
65 int32_t const PULL_Ommega_Min_f16 = ((RPM_ACTIVATE_MIN*65536/60/PWM_Frequency)<<16); // 角度增量下限(转速下限)
66 int32_t const PULL_Ommega_Inc_f16 = PULL_Ommega_Max_f16/(9*PWM_10th_sec); // 启转加速度, 转速增量
```


◆ 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



- ◆ `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_beta, int32_t I_alpha)` 迭代运算函数, 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 (PLL输入, 同步转时很小) 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。函数内对I_beta, I_alpha做了静轴到动轴的坐标变换。
- ◆ `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, 如果两次PWM做一次迭代运算, 参数一就用PWM频率值的一半。
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

