



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():控制启转、停转, 故障监测及后期处理等

- 判断是否欠压、过温短路, 以及处理转速设定值等等
- 启转: 设定转速 RPM_Set 大于最小值, 状态0变2, 然后启动
- 停转: 设定转速 RPM_Set \leq 0, 降速停转

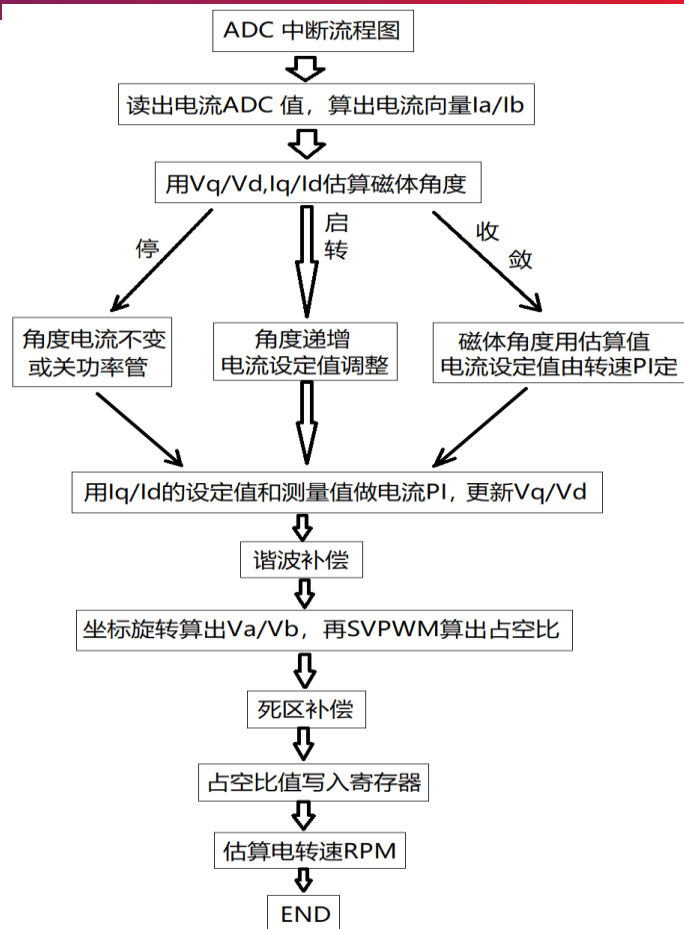
◆ADC中断: 磁体位置估算, 电流PI运算, PWM占空比计算

- 停或转都会执行ADC中断代码, 停转时功率管关了, 计算没停止。
- 坐标变换q轴角度总是用 Angle_q
- 启转时代码控制 Angle_q 递增, 并控制电流设定值
- 同步后 Angle_q 用估算值, 电流设定值用转速 PI 运算的结果

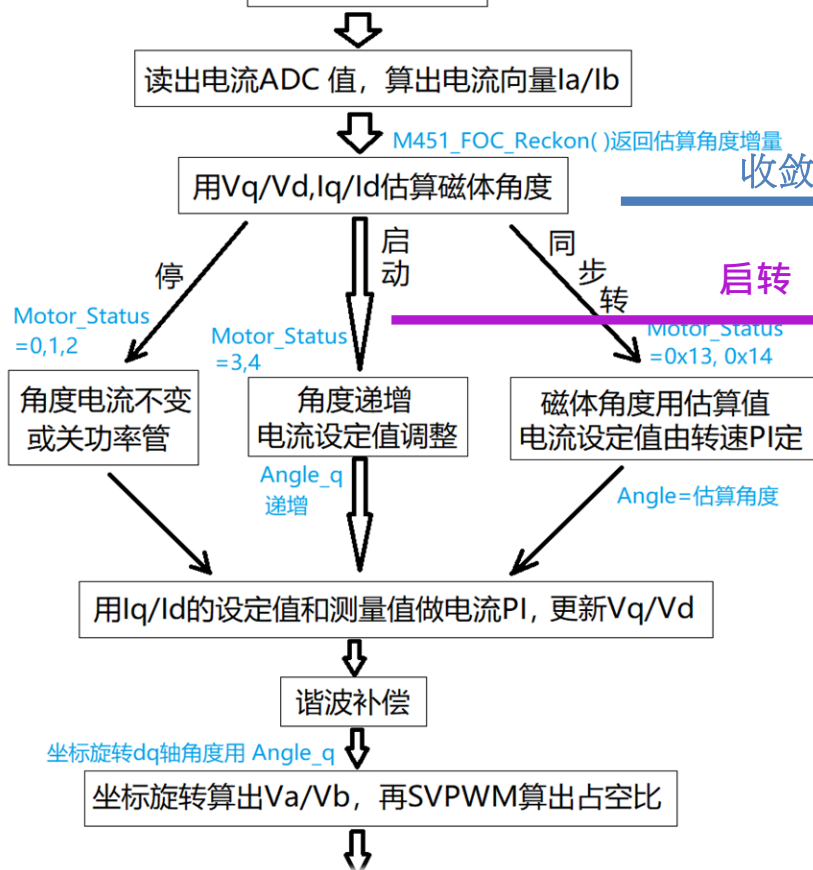
◆Systick中断: 转速PI运算, 故障显示等

◆ 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, 从任意角度起转
- 状态4, 从固定角度起转
- 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 中断流程图



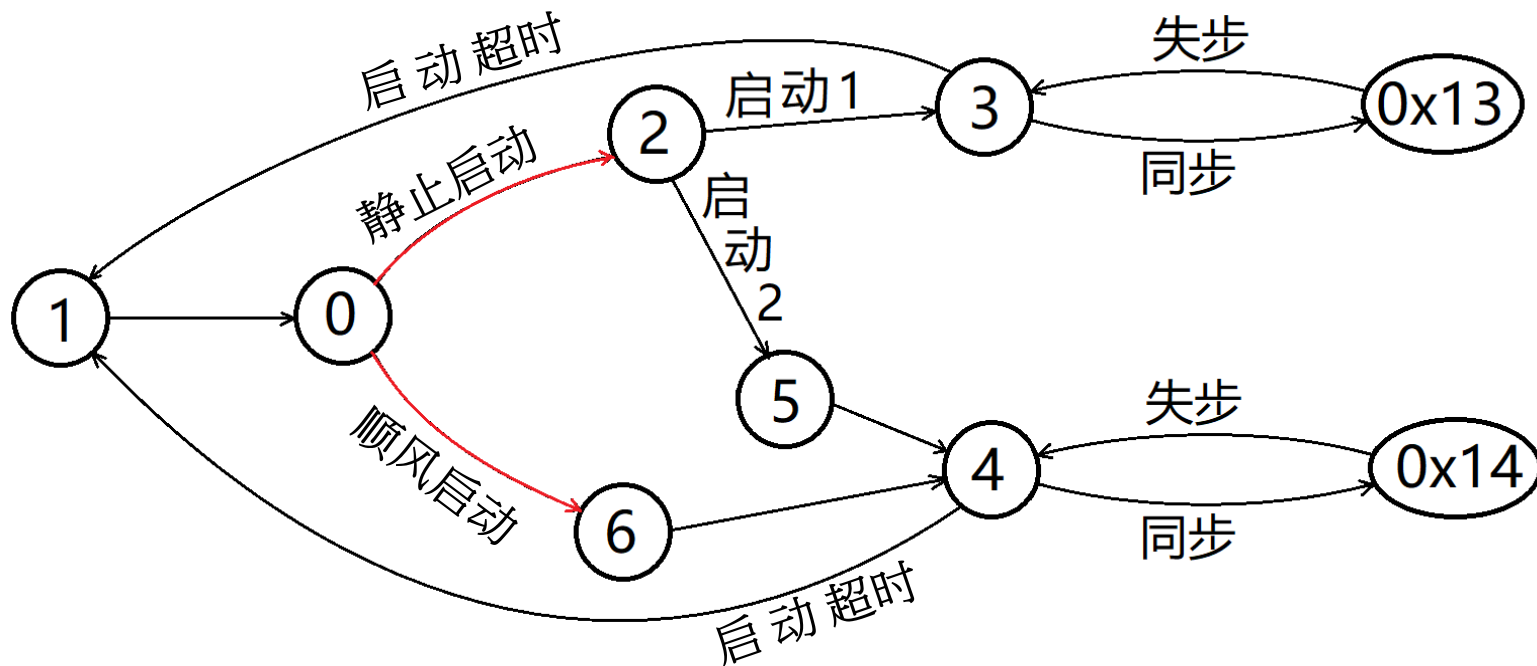
```
245 //=====
246 // 接下来, 按 Motor_Status 不同的值, 执行不同代码
247 //=====
248
249 if(Motor_Status & 0x10){ Angle_q += temp16 ; // 已同步,角度加增量, An
250 }
251
252 //=== 状态3 拖动电流每转一扇区增大一些 ===
253 else if(Motor_Status == 3){ // 进状态3前Iq_set_f12,V
254 // 未定义从固定角度起转,
255 #ifndef __ACTIVE_FROM_FIXED_ANGLE
256 uint16_t static const *LastptrSection;
257 }
```

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

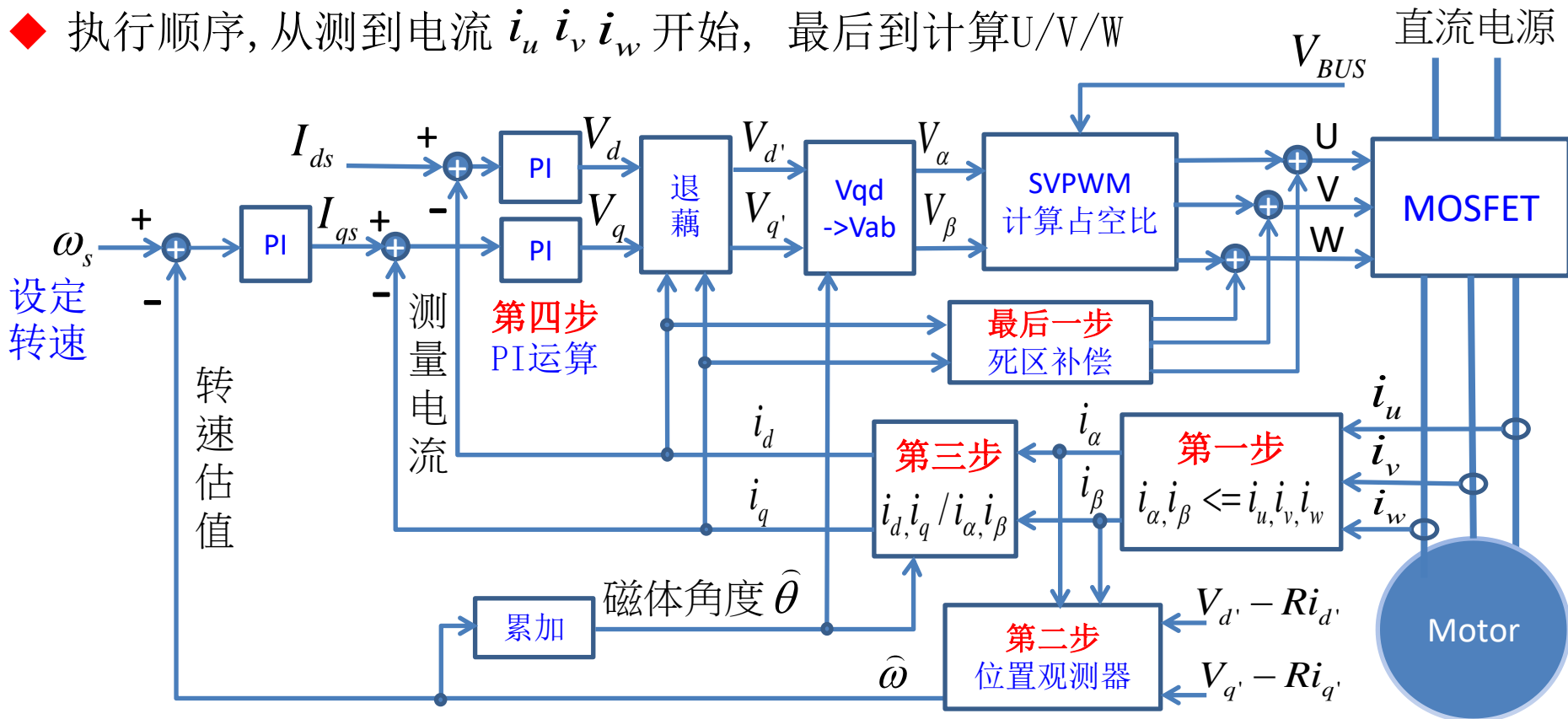
同步转时,角度用估算值,电流转速PI 计算结果



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() 中定时做功率PI算出转速增量，控制加速度后赋值给设定转速 RPM_Set_Target, 此值若大于转速下限RPM_SET_MIN, 后续会变状态2开始启动

```
447 //== 功率 PI 计算转速增量，再按旋钮转速控制增量，取二者较小值，加到 RPM_Set_Target =====
448
449 temp32 = Power_Limit - Motor_Power ;
450 temp32_1 = Kp_Power*(temp32 - Err_Power) + Ki_Power*temp32 ; // 功率PI 运算结果，转速增量
451 temp32_1 >>= 8 ; // 功率PI 算出的转速增减量，
452
453 temp32 = RPM_Command - RPM_Set_Target ; // 旋钮增量
454 if(temp32 < -50) temp32 = -50 ; // 控制降速
455 else{
456     if(RPM_Measure < RPM_SET_MIN*2){
457         if(temp32 > 40) temp32 = 40 ; // 低速时的加速度可以快一点
458     }
459     else if(temp32 > 20) temp32 = 20 ; // 加速度
460 }
461 //== 旋钮增量temp32, 功率 PI 增量 temp32_1, 取二者较小的加到 RPM_Set_Target =====
462 // 如果功率不超，功率PI 计算结果都是增速，所以前面
463 if(temp32 > temp32_1) temp32 = temp32_1 ;
464
465 temp32 += RPM_Set_Target ;
466 if(temp32 > RPM_SET_MAX) temp32 = RPM_SET_MAX ;
467 else if(temp32 < 0) temp32 = 0 ;
468 RPM_Set_Target = temp32 ; // 慢变化值 0 ~ RPM_SET_MAX
```


起转、停转的代码流程

- ◆ 2, 若在同步转(347行), 就检测设定转速=0时是否提前停转, 不等失步再停转

```
345 //===== 在同步转 =====
346 if((Motor_Status & 0x10) == 0x10){
347     if((RPM_Set<=0) &&((RPM_Measure*65536/60/PWM_Frequency) <(PULL_Ommege_Min_f16 >>16))){ // 转速低于下限
348         Exec_Counter = 1*PWM_10th_sec;
349         // 或发生ADC 中断进状态0x13, 0x14, 进(RPM_Set<=0)的分支, 但下句变状态1
350         Motor_Status = 1; // 去状态1 刹车, 准备又立即再次启动
```

- ◆ 3, 若未在同步转就判断是否需要启转。若RPM_Set>0就在状态0执行结束后开始起动(394行)。

- ◆ 4, 起动前先测是否在顺风转(398行)

```
393 //===== 未同步, 且设定转速 RPM_Set 大于0, 需要启转 =====
394 else{ // 状态2, 3, 4, 5, 6是在起转, 只有状态0不自动变其它状态
395     if(Stop_Runing + Motor_Status + Exec_Counter ==0){
396
397         #if 1
398             temp32 = Test_Period(); // 若在顺风转, 此函数在 Angle_q == -16384 时刻返回转速
399         #endif
```

起转、停转的代码流程

- ◆ 5, 在ADC中断里, 状态1或2执行次数=0后, 若RPM_Set>0, 就去状态5做启动前的准备(470行)。

```
451 if(Motor_Status <= 2){  
452     Vout_Ov1_f4 = 0;  
453     Motor_Power = 0;  
454     RPM_Measure = 0;  
455     Vq_out_f17 = 0;    Vd_out_f17 = 0;  
456     Id_set_f12 = 0;    Iq_set_f12 = 0;  
457  
458     if(Exec_Counter) --Exec_Counter ;  
459 else if(Motor_Status != 0){  
460     if(RPM_Set < RPM_STOP_VALUE) Motor_Status = 0 ;  
461     else if(RPM_Set <= 0) Motor_Status = MOTOR_STOP_STATUS;  
462     else{  
463         Vout_UpLimit_f17 = VOLTAGE_UpLimit_while_Active_f17;  
464  
465 #ifdef __ACTIVE_FROM_FIXED_ANGLE  
466     Exec_Counter = TIME_MotorLock ;  
467 #else  
468     Exec_Counter = PWM_10th_sec >> 3 ;  
469 #endif  
470     Motor_Status = 5 ;  
471 }
```

// Motor_Status=0时看Exec_Counter值可知从哪里跳过来
// 输出电压的模
// 电机功率

// 输出0电压, 角度值Angle_q 放在中断外写
// 状态0关功率管, 状态1, 2输出0电压不变, 电流设定值用不到

// 执行计数

// 转速设定值比停转值还负, 是在给自举电容充电
// 这个停转状态可以是0, 也可以是5

// 启动时的电压, 限压可以防止启动时转速上冲

// 定义了从固定角度起转
// 锁定时间能让电机稳定在 -120 度

// 时长参考2倍L/R, 电流升到较大电机没转, 检测是否缺相

// 先状态5磁铁吸到固定角度, 再跳转到状态4 起转

起转、停转的代码流程

- ◆ 6, 状态3启动时, 函数Motor_Active() (262行) 调整拖动转速逐步增加, 输出电流每转过一扇区增加一次(292行)。

```
253 //=== 状态3 拖动电流每转一扇区增大一些 =====  
254 else if(Motor_Status == 3){ // 进状态3前Iq_set_f12,Vout_UpLimit_f17 赋初值  
255  
256 #ifndef __ACTIVE_FROM_FIXED_ANGLE // 未定义从固定角度起转, 就是从状态3 起转  
257     uint16_t static const *LastptrSection;  
258  
259     Id_set_f12 = 0 ; // 可能是同步转时失步又回到状态3, 所以需清0  
260     Vd_out_f17 = 0 ;  
261  
262     temp16 = Motor_Active(10, PULL_Ommega_Inc_f16, PULL_Ommega_Max_f16, 160*1000); // Angle_q 逐增, 或调整  
263 // Angle_q < 0 时仅线性加速  
264 //== 判断是否可以变同步转了 =====  
265 if((Motor_Status & 0x10) == 0 ){ // 函数 Motor_Active() 未切同步  
266     Circle_Tick = Angle_q ; // 起转时若ptrSectionClear=0不会立即同步, 会执行到此  
267     if((RPM_Set <= 0) || (Motor_LastStatus == 0x13)){ // 启动过程中又停止启动, 或失步回到此状态  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291 else if(LastptrSection != ptrSectionClear){ LastptrSection = ptrSectionClear; // 已转过一个扇区  
292     if(Iq_set_f12 < CURRENT_Max_f12/2) Iq_set_f12 += CURRENT_Max_f12 >>4; // 一圈把电流加到上限  
293 }
```

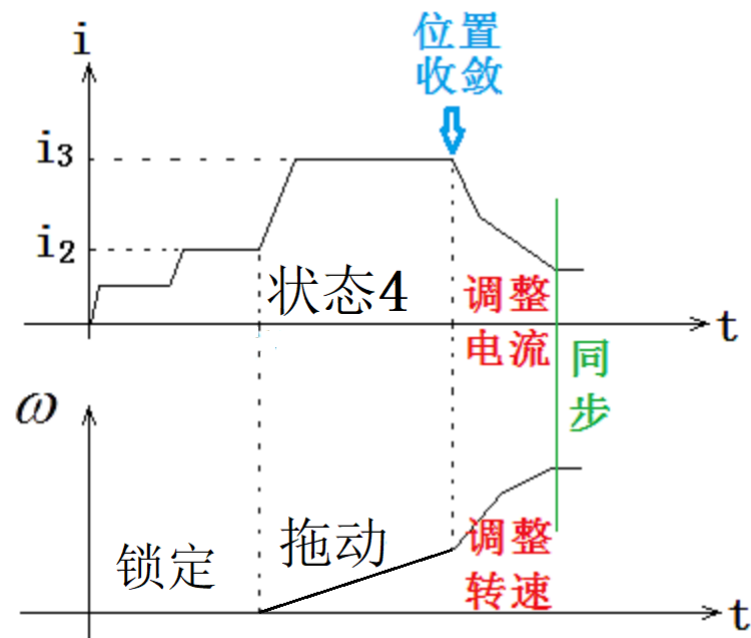
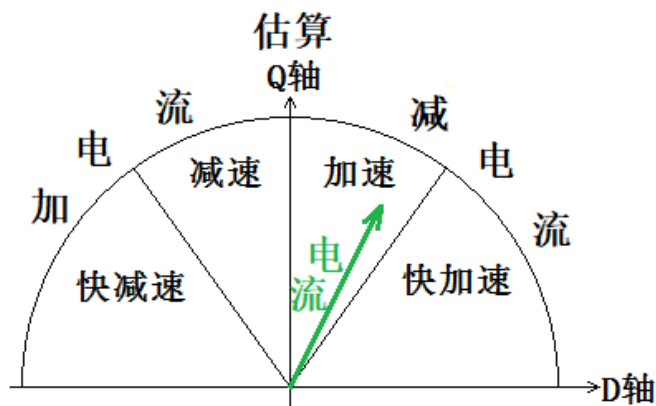
起转、停转的代码流程

- ◆ 7, 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), 若设定转速已负值并且转速已到下限(347行), 就不再等失步、直接刹车停转, 这里也可以变状态0自由停转。

```
345 //===== 在同步转 =====
346 if((Motor_Status & 0x10) == 0x10){
347     if((RPM_Set<=0) &&((RPM_Measure*65536/60/PWM_Frequency) <(PULL_Ommege_Min_f16 >>16))){ // 转速低于下限
348         Exec_Counter = 1*PWM_10th_sec ;
349         // 或发生ADC 中断进状态0x13, 0x14, 进(RPM_Set<=0)的分支, 但下句变状态1
350         Motor_Status = 1 ; // 去状态1 刹车, 准备又立即再次启动
351     }
```

◆ 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虑轻重载
- 电流方向(绿)与估算Q轴重合时，变为同步



◆ #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，第一个词全大写，最后词不能全大写，中间词随意

```
43 int32_t const CURRENT_MotorLock_f12 = CURRENT_Max_f12*3 >>5 ;    // 锁定时电流值
44 int32_t const VOLTAGE_UpLimit_while_Active_f17 = 30 << 17 ;      // 起转时电压上限, 参考额定电压的10%
```

◆ 某个变量的位变量，变量名 + 全大写位定义名

```
41 //==== 低 4bit 非0, PWM中断里关功率管 ==== 低8bit非0 设定转速=0 ==== 其它非0 降速受控
42 INT_EXT uint16_t Stop_Runing ;                               // 非0就停转
43 #define Stop_15V_LOW      0x0001                            // 15V 电压低
44 #define Stop_SHORT        0x0002                            // 短路了, 发生了 Brake
45 #define Stop_ACTIVE_FAIL  0x0004                            // 多次启动失败
46
47 #define Stop_PHASE_OPEN   0x0010                            // 缺相
```

◆ 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

◆ 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，129行注释掉就可以了

```
128 PWM0->POLCTL = 0x002A ; // PWM135 反相, 输出0 MOS导通
129 // PWM024_Out_0_MOS_On() ;
130 PWM135_Out_1_MOS_On() ;
131 PWM0->MSK = ShutDown_AllMOS ;
132 PWM0->MSKEN = 0x003F ; // 使能 MASK 功能, 先关断所有输出
133 PWM0->POEN = 0x003F ; // 低6bit输出使能, 但引脚输出值 = ShutDown_AllMOS
134
```


◆ 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->SCTL[0] = EADC_PWMOTGO_TRIGGER | ADC_CHANNEL_PWM01 ;           // ADC_PWM0零点触发, 低4位是ADC 通道号
26     EADC->SCTL[1] = EADC_PWMOTGO_TRIGGER | ADC_CHANNEL_PWM23 ;           // ADC_PWM23
27     EADC->SCTL[2] = EADC_PWMOTGO_TRIGGER | ADC_CHANNEL_PWM45 ;           // ADC_PWM45
28
29     EADC->SCTL[3] = EADC_PWMOTGO_TRIGGER | ADC_CHANNEL_SPEED ;           // Speed, 调整旋钮
30     EADC->SCTL[4] = EADC_PWMOTGO_TRIGGER | ADC_CHANNEL_VBUS ;           // VBUS
31     EADC->SCTL[5] = EADC_PWMOTGO_TRIGGER | ADC_CHANNEL_TEMPERATURE ;     // 功率管温度
32
33     EADC->SCTL[10] = EADC_SOFTWARE_TRIGGER | 14 ;                       // 软件触发ADC 测初始位置, 用于顺风起转
34                                     // ADC参考电压若选片内4096mV, Vref 引脚外接1uF电容, 不能再接到5V
35     SYS->VREFCTL = 0 ;                                                   // Vref选择: 0 =外接Vref, 0xF=4096mV, 注意与常量定义ADC_VREF_MV
36     SYS->IVSCTL = 1 ;                                                   // 使能片内温度 Sensor 的测量
37     EADC->SCTL[17] = 16 << 24 ;                                         // 片内温感的 ADC 采样时间扩展 16个时钟
38
39     EADC->STATUS2 = 1 ;                                                  // 开中断前先清中断标志
40     EADC->INTSRC[0] = 1 << 2 ;                                           // 模块0, 1, 2, 测完三相电流就进中断开始计算
41
42     NVIC_SetPriority(ADC00_IRQn, 1) ;                                   // 优先级=1, 最高优先级0保留急用
43     NVIC_EnableIRQ(ADC00_IRQn) ;                                       // 使能ADC 中断
44 }

```


◆ 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 $Lq \geq Ld$

```
251 白 #if 1
252     Lq_Inductor = 690 ;           // 电机电感 uH, 风机
253     Ld_Inductor = 665 ;           // 必须 Lq >= Ld
254     Unit_Magnify = CURRENT_Adc1000*25*8 ; // H=8, 表示电流ADC值乘8后才用于角度估算
255
256 #else
257     Lq_Inductor = 690 >>3;         // 建议LqLd最大29位: 0x1FFF_FFFF
258     Ld_Inductor = 665 >>3;         // 电感太大若LqLd会溢出(32位符号数), 此仨变量都除8
259     Unit_Magnify = CURRENT_Adc1000*25 ; // H=8 与除8 抵消
260
261 #endif                             //电流ADC出12位符号数, 乘8后15位, Lq超32.768mH就可以除2, 4, 8了
262
```

◆ 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) 7
```

◆ 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, $V_{ref}=5000\text{mV}$, 所以1A的ADC值是: $10\text{毫欧} \times 5\text{倍} \times 4096 / 5\text{V}$, 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*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)
```

◆ 母线电压Vbus_0v1是0.1V的电压数值

```
484 //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
485 temp32 = (ADC_VREF_MV*(91+10)/10*(EADC->DAT[4]&0xFFFF)/100+2048) >>12;
486 // Vbus_0v1 = temp32;
487 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
488
489 temp32 = Vbus_0v1*150 >>8; // 调制比=1是0.577 =150/256, 过调制 172/256=2/3
490 if(temp32 >16383)temp32 =16383; // 电压最大14位 = 16383 (1638.3V)
491 Vout_Max_f17 = temp32 << 17; // Vdq 限幅值加17位小数, 最大到31位
```

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 $15 \sim 100$ 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。

```
46 // 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
47 // 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)
48 // 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)
49
50 // Kp 取值 2*3.1415*F*L/N, N 取值 10*3.14159(一般不小于 5*3.14159) 得 Kp =F*L/5, 代入 Kp_Current公式得
51 // 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
52
53 // 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, 但转速调整时会有过冲
54 // 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
55
56 // 本例 L=0.7mH, F=15kHz, R=1欧, Ar=0.05欧, 算得 2100, 200 ( 大多数情况, PI参数可以再大一倍)
57
58 int32_t      Kp_Current = 2100,  Ki_Current = 200; // 电流留5位小数做PI, 结果是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 $1/20 \sim 1/100$

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

- ◆ 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

```
01  
68 int32_t    Kp_Speed = 400,    Ki_Speed = 2;           // 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  
69
```

- ◆ 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10%

```
38 // 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
39 #define CURRENT_MIN (30<<7) // 做电流 PI 时会先右移7位, 做小数对齐
40 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
41 int32_t const CURRENT_Max_f12 = 40*CURRENT_OA1_Value_f12 +(30<<7); // 正常转动时电流上限, 加一点显示时高位不跳动
42
43 int32_t const CURRENT_MotorLock_f12 = CURRENT_Max_f12*3 >>5 ; // 锁定时电流值
44 int32_t const VOLTAGE_UpLimit_while_Active_f17 = 30 << 17 ; // 起转时电压上限, 参考额定电压的10%
45
```

- ◆ 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
 - 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

```
72 #define RPM_ACTIVATE_MAX          (600*POLE_PAIR) // 拖动转速Pull_Ommega_f16上限, 比能闭环转的最高转速略大
73 #define RPM_ACTIVATE_MIN          (300*POLE_PAIR) // 拖动转速下限, 取值一般比能闭环转的最低速略小一点
74
75 int32_t const PULL_Ommega_Max_f16 = ((RPM_ACTIVATE_MAX*65536/60/PWM_Frequency)<<16); // 拖动转速变成角度增量
76 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)
77 int32_t const PULL_Ommega_Inc_f16 = PULL_Ommega_Max_f16/(10*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

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

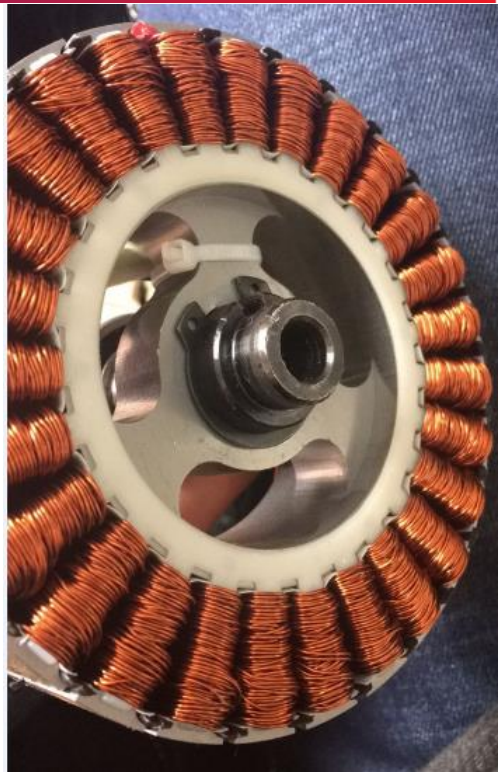
- 令电流ADC值 $Mi = i'$, 微亨值 L' , 电势变 ε' , 公式变为

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

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

