



新唐 M451 无感 FOC 方案代码说明

www.nuvoton.com



nuvoTon

- ◆ 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
- ◆ 起转电流配置
- ◆ 起转加速度配置
- ◆ 代码总体框架
- ◆ 代码中的计算公式
- ◆ 数学函数简介

◆ #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_Pull_Max_f12 = CURRENT_Max_f12*9 >>5 ;    // 拖动时电流最大值
42 int32_t const CURRENT_MotorLock_f12 = CURRENT_Max_f12*3 >>5 ;    // 锁定时电流值
43
```

◆ 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 ;                                   // 清中断标志
34     EADC->INTSRC[0] = 1ul << 2 ;                         // 使能模块2触发ADC中断0
35 }
```

↑
ADC通道号

↑
SCTL[2]转换完产生中断

◆ 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 $L_q \geq L_d$

```
238 #if 1
239     Lq_Inductor = 690 ;           // 电机电感 uH, 风机
240     Ld_Inductor = 665 ;           // 必须  $L_q \geq L_d$ 
241     Unit_Magnify = CURRENT_Adc1000*25*8 ; // H=8, 电流ADC值乘8后才用于计算
242 #else
243     Lq_Inductor = 690 >>3;        // 电感太大若LqIq会溢出(32位符号数), 此仁变量都除8即可
244     Ld_Inductor = 665 >>3;        // 建议LqIq最大29位: 0x1FFF_FFFF
245     Unit_Magnify = CURRENT_Adc1000*25 ; // H=8, 与除8抵消
246 #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
247
```

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

- 可三三测量求平均

```
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 $0.01\Omega * 5\text{倍} * 4096/5V$, 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

```
425 // 电压 =(ADC值/4096)*5V*(分压比(91+10)/10), 再乘10变0.1V值 =ADC值*505/4096
426 temp32 = ((EADC->DAT[4]&0xFFFF)*505 +1024) >>11; // 少右移一位, 2x
427 temp32 = temp32 + (Vbus_0v1 <<1); // (2x+2y), 25mV数值
428 Vbus_0v1 = (temp32 +2) >>2; // 母线电压0.1V的数值, (2y+2x)/4
429 Section = M4_Svpwm(&temp_Vd, &temp_Vq, &temp32, Pwm_Duty_Uplimit); // 计算PWM 占空比, 用25mV数值
```

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

● 若启转时电机滋滋响, 电流快速波动, 就是PI参数过大了，取值小点再测。

```
43
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 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
56 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

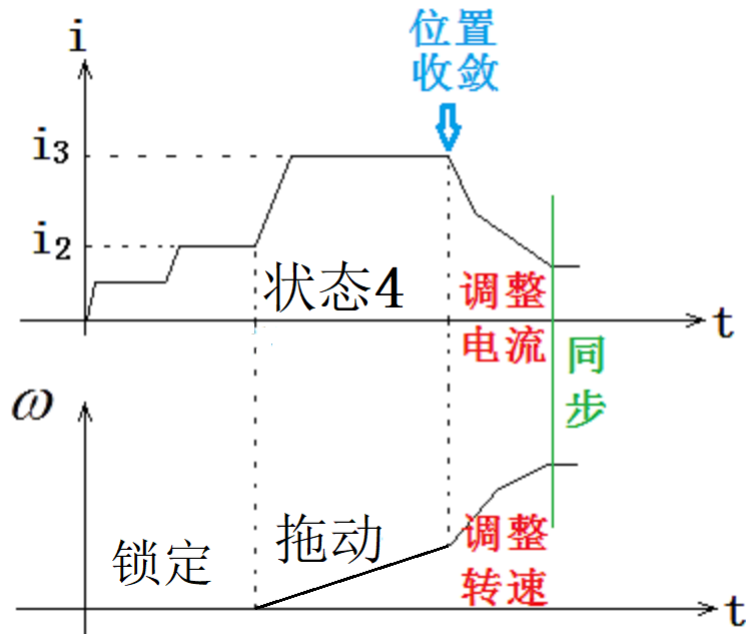
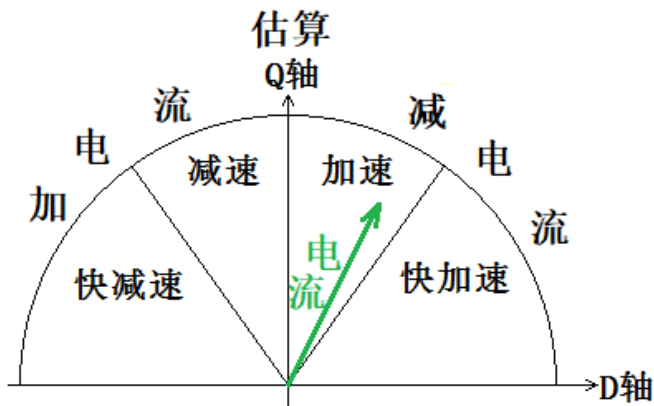
```
54 // PI运算:电流ADC值先增5位小数,PI运算结果是17位小数的0.1V数值,从数值上看是 PI 运算让数据又增加了12位小数
55 int32_t    Kp_Current = 1500, Ki_Current = 50; // PI前加5位小数,PI结果是17位小数,所以此值相当于有12位小数
56 int32_t    Kp_Speed   = 200,  Ki_Speed   = 3;  // 转速PI运算结果是12位小数的电流设定值,相当于此有12位小数
```

- ◆ 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 (9<<12) // 电流 PI 时右移7位
38 int32_t const CURRENT_Min_f12 = -CURRENT_MIN; // 下限负值防0值漂移停不下来, 负多了降速充母线
39 int32_t const CURRENT_Max_f12 = 4*CURRENT_1A_Value_f12 +(8<<12); // 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_Pull_Max_f12 = CURRENT_Max_f12*9 >>5 ; // 拖动时电流最大值
42 int32_t const CURRENT_MotorLock_f12 = CURRENT_Max_f12*3 >>5 ; // 锁定时电流值
```

◆ 状态4先匀加速拖动，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

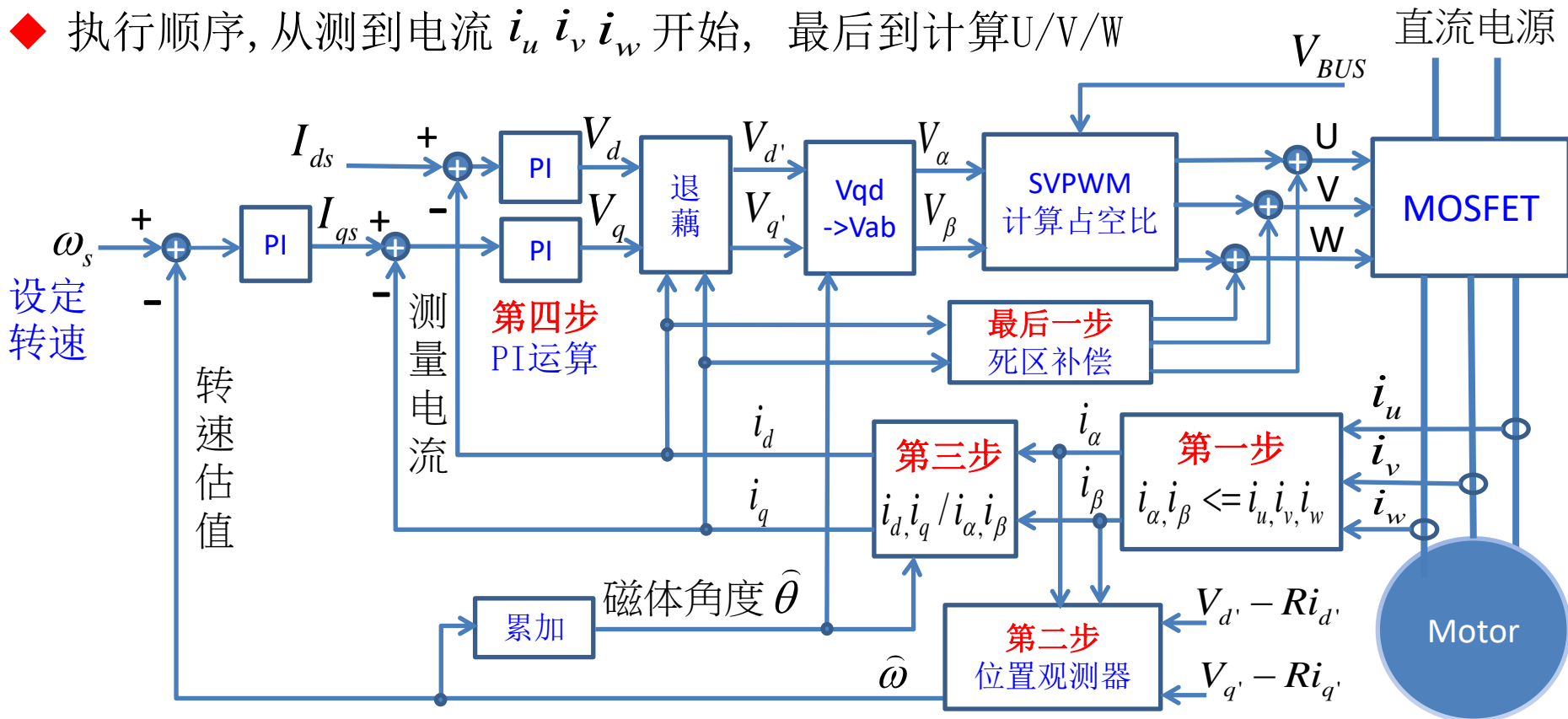


- ◆ 拖动转速上限PULL_Ommege_Max_16, 取能同步转的转速
- ◆ 加速度PULL_Ommege_Inc_f16 先小点, 磁铁能跟着转, 后期再调整
- ◆ 拖动转速下限, 可先取上限的 $\frac{1}{2} \sim \frac{1}{4}$, 后期再调整
 - 一圈角度360 用 65536 表示, 加16位小数就是低16位成了小数部分

```
59
60 #define RPM_ACTIVATE_MAX          (POLE_PAIR*600)    // 拖动转速Pull_Ommege_f16上限, 这个转速要能保持同步转动
61 #define RPM_ACTIVATE_MIN          (POLE_PAIR*300)    // 拖动转速调整下限, 参考能同步转动的最低转速, 太小起转不顺
62
63 int32_t const PULL_Ommege_Max_f16 = ((RPM_ACTIVATE_MAX*65536/60/PWM_Frequency)<<16); // 一PWM周期角度增量上限
64 int32_t const PULL_Ommege_Min_f16 = ((RPM_ACTIVATE_MIN*65536/60/PWM_Frequency)<<16); // 角度增量(转速)的下限
65 int32_t const PULL_Ommege_Inc_f16 = PULL_Ommege_Max_f16/(9*PWM_10th_sec);           // 启转加速度, 转速增量
66
```

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

◆代码总体分为以下三大部分main()+两个中断

◆Main():电压监测, 控制启转、停转等

- 启转: 设定转速大于最小值
- 停转: 设定转速 ≤ 0
- 可随时变转向

◆ADC中断:磁体位置估算, 电流PI运算, PWM占空比计算

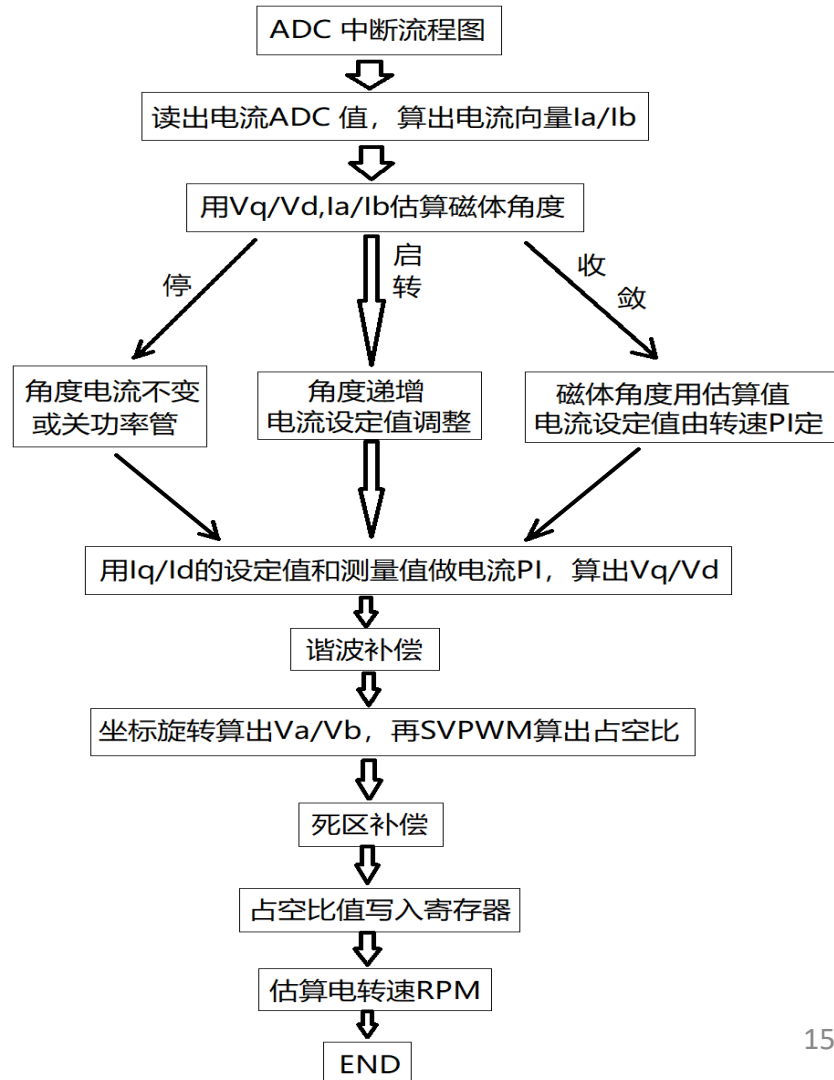
- 坐标变换q轴角度总是用Angle_q
- 启转未收敛时代码控制Angle_q 递增
- 同步后 Angle_q 用估算值

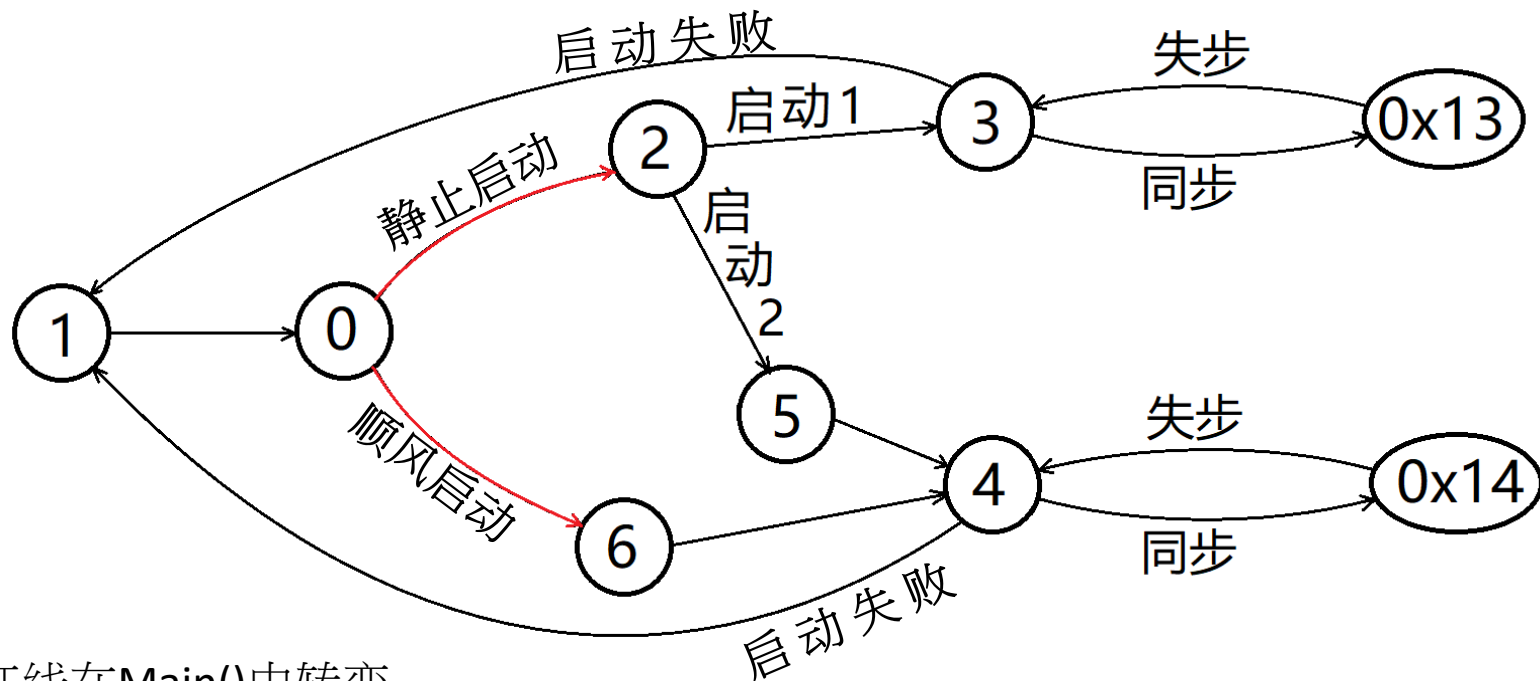
◆Systick中断:转速PI运算, 功率计算, 调速旋钮监测等

ADC 中断代码流程图

◆ 起转, 停转, 同步由以下状态机控制

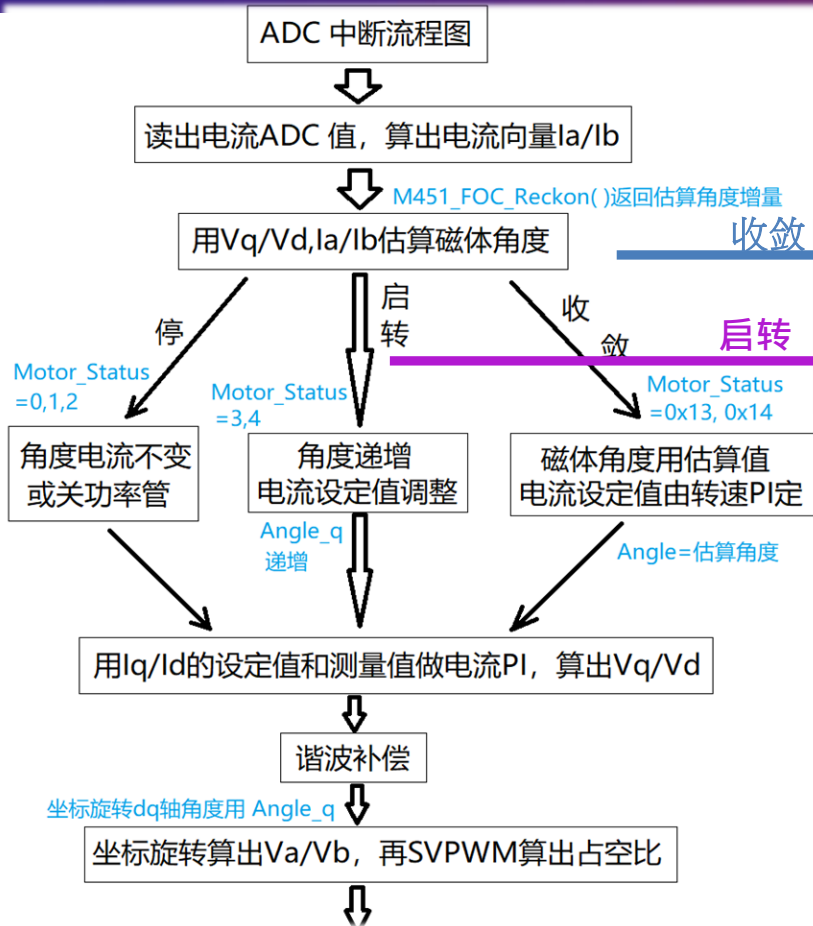
- 状态0, 关功率管, 不控制电机
- 状态1, 输出0电压, 用于刹车
- 状态2, 只下MOS输出, 用于自举电容充电
- 状态3, V/F起转
- 状态4, I/F起转
- 状态5, 起转前的锁定, 然后去状态4
- 状态6, 顺风起转准备, 然后去状态4
- 状态7, 啥也不做, 用于调试
- Bit4=1, 同步转动状态 (0x13, 0x14)
- 其它, 变状态0





红线在Main()中转变
启动1和启动2, 二选一

ADC 中断代码流程图



```
204 if(Motor_Status & 0x10){ Angle_q += temp16 ; // 已同步,
205 }
206
207 //==== 状态3电压拖动起转 =====
208 else if(Motor_Status == 3){
209     #ifndef __DO_CURRENT_PI_WHILE_ACTIVE // 未定义,
210
211
```

停转,起转由状态机Motor_Status控制代码流程

◆ 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)

◆ 96位唯一序列号用于代码加密

◆ 12位DAC 实时输出中间量数值观察效果

◆ 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....., 若不移位就是增加了小数位
- ◆ `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Estimate_Q_Position(PLL输出), 返回值是反电势, 除Unit_Magnify后, 小数位与实参电压一样。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

