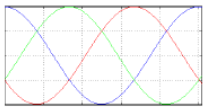
## FOC电流环控制：

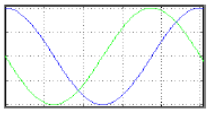
ADC1_2_IRQHandler (stm32f10x_MC_it.c) ->
TSK_HighFrequencyTask (MCTasks.c) ->
FOC_CurrController ->
|_____ SPD_GetElAngle 得到电角度 (hElAngledpp)
|_____ PWMC_GetPhaseCurrents ,得到相电流Ia和Ib (Iab)
|_____ MCM_Clarke ,clark变换得到Ialpha和Ibeta (Ialphabeta)
|_____ MCM_Park, park变换得到Iq和Id (Iqd)
|_____ PI_Controller执行两次，分别处理Iq和Id, 得到Vq和Vd (Vqd)
|_____ FF_VqdConditioning 前馈条件
|_____ MCM_Rev_Park 反Park变换，根据Vqd和电角度hElAngledpp计算出Valpha和Vbeta (Valphabeta)
|_____ PWMC_SetPhaseVoltage (PWMnCurrFdbkClass.c) 根据Valphabeta设置相电压

## Clark变换:

- clarke 变换: ia,ib,ic (120°) 转换为 iα,iβ(90°); (假设ia+ib+ic=0)，交流->交流：



$$i_\alpha = i_{as}$$
$$i_\beta = -\frac{i_{as} + 2i_{bs}}{\sqrt{3}}$$



MC_Math.c

```
/**
  * @brief  This function transforms stator currents Ia and qIb (which are
  *       directed along axes each displaced by 120 degrees) into currents
  *       Ialpha and Ibeta in a stationary qd reference frame.
  *                    Ialpha = Ia
  *                Ibeta = -(2*Ib+Ia)/sqrt(3)
  * @param  Curr_Input: stator current Ia and Ib in Curr_Components format
  * @retval Stator current Ialpha and Ibeta in Curr_Components format
  */
Curr_Components MCM_Clarke(Curr_Components Curr_Input)
{
  Curr_Components Curr_Output;

  int32_t qIa_divSQRT3_tmp, qIb_divSQRT3_tmp ;
  int32_t wIbeta_tmp;
  int16_t hIbeta_tmp;

  /* qIalpha = qIas*/
  Curr_Output.qI_Component1= Curr_Input.qI_Component1;

  qIa_divSQRT3_tmp = divSQRT_3 * (int32_t)Curr_Input.qI_Component1;

  qIb_divSQRT3_tmp = divSQRT_3 * (int32_t)Curr_Input.qI_Component2;

  /*qIbeta = -(2*qIbs+qIas)/sqrt(3)*/

  wIbeta_tmp = (-(qIa_divSQRT3_tmp)-(qIb_divSQRT3_tmp)-(qIb_divSQRT3_tmp))>> 15;
```

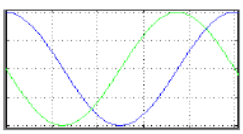
Curr_Input.qI_Component1 为 Ias, Curr_Input.qI_Component2 为 Ibs
Ialpha = Curr_Output.qI_Component1 = Ias
Ibeta = Curr_Output.qI_Component2 = -(Ias+2Ibs)*divSQRT_3>>15 = -(Ias+2Ibs)/sqrt(3)

```
#define divSQRT_3 (int32_t)0x49E6      /* 1/sqrt(3) in q1.15 format=0.5773315*/
```

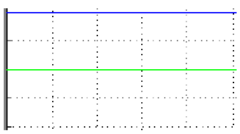
右移15位叫做q1.15格式，这是为了方便运算，避免用浮点运算，以降低运算时间。
1/sqrt(3) = 0.57735026918962576450914878050196
(1<<15)/sqrt(3) = 32768*0.57735026918962576450914878050196 = 18918.61362
去掉小数点就是18918 = ox49E6

## Park变换：

- Park变换：$i_\alpha$,$i_\beta$ 转换为$i_d$,$i_q$ (90°)，交流->直流：



$$i_{qs} = i_\alpha \cos\theta_r - i_\beta \sin\theta_r$$
$$i_{ds} = i_a \sin\theta_r + i_\beta \cos\theta_r$$



```
/**
  * @brief  This function transforms stator currents Ialpha and Ibeta, which
  *       belong to a stationary qd reference frame, to a rotor flux
  *       synchronous reference frame (properly oriented), so as Iq and Id.
  *             Id= Ialpha *sin(theta)+qIbeta *cos(Theta)
  *             Iq=qIalpha *cos(Theta)- qIbeta *sin(Theta)
  * @param  Curr_Input: stator current Ialpha and Ibeta in Curr_Components format
  * @param  Theta: rotating frame angular position in q1.15 format
  * @retval Stator current Iq and Id in Curr_Components format
  */
Curr_Components MCM_Park(Curr_Components Curr_Input, int16_t Theta)
{
  Curr_Components Curr_Output;
  int32_t qId_tmp_1, qId_tmp_2, qIq_tmp_1, qIq_tmp_2;
  Trig_Components Local_Vector_Components;
  int32_t wIqd_tmp;
  int16_t hIqd_tmp;

  Local_Vector_Components = MCM_Trig_Functions(Theta);

  /*No overflow guaranteed*/
  qIq_tmp_1 = Curr_Input.qI_Component1 * (int32_t)Local_Vector_Components.hCos;

  /*No overflow guaranteed*/
  qIq_tmp_2 = Curr_Input.qI_Component2 * (int32_t)Local_Vector_Components.hSin;

  /*Iq component in Q1.15 Format */
  wIqd_tmp = (qIq_tmp_1-qIq_tmp_2)>>15;
  hIqd_tmp = (int16_t)(wIqd_tmp);


  Curr_Output.qI_Component1 = hIqd_tmp;

  /*No overflow guaranteed*/
  qId_tmp_1 = Curr_Input.qI_Component1 * (int32_t)Local_Vector_Components.hSin;

  /*No overflow guaranteed*/
  qId_tmp_2 = Curr_Input.qI_Component2 * (int32_t)Local_Vector_Components.hCos;

  /*Id component in Q1.15 Format */
  wIqd_tmp = (qId_tmp_1+qId_tmp_2) >>15;
  hIqd_tmp = (int16_t)(wIqd_tmp);

  Curr_Output.qI_Component2 = hIqd_tmp;

  return (Curr_Output);
} ? end __attribute__ ?
```

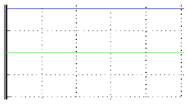
Curr_Input.qI_Component1 为 Ialpha, Curr_Input.qI_Component2 为 Ibeta , Theta为电角度

Iq= Curr_Output.qI_Component1 = Ialpha *cos(Theta)-Ibeta *sin(Theta)

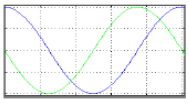
Id= Curr_Output.qI_Component2 = Ialpha *sin(theta)+Ibeta *cos(Theta)

## 反Park变换：

- 反Park变换：$V_q$,$V_d$ 转换为Vα , Vβ: 由于$V_q$,$V_d$ 为直流信号，因此PID算法可直接运用于此类信号的控制



$$v_\alpha = v_{qs}\cos\theta_r + v_{ds}\sin\theta_r$$
$$v_\beta = -v_{qs}\sin\theta_r + v_{ds}\cos\theta_r$$



```
/**
  * @brief  This function transforms stator voltage qVq and qVd, that belong to
  *       a rotor flux synchronous rotating frame, to a stationary reference
  *       frame, so as to obtain qValpha and qVbeta:
  *             Valfa= Vq*Cos(theta)+ Vd*Sin(theta)
  *             Vbeta=- Vq*Sin(theta)+ Vd*Cos(theta)
  * @param  Curr_Input: stator voltage Vq and Vd in Volt_Components format
  * @param  Theta: rotating frame angular position in q1.15 format
  * @retval Stator voltage Valpha and Vbeta in Volt_Components format
  */
Volt_Components MCM_Rev_Park(Volt_Components Volt_Input, int16_t Theta)
{
  int32_t qValpha_tmp1,qValpha_tmp2,qVbeta_tmp1,qVbeta_tmp2;
  Trig_Components Local_Vector_Components;
  Volt_Components Volt_Output;

  Local_Vector_Components = MCM_Trig_Functions(Theta);

  /*No overflow guaranteed*/
  qValpha_tmp1 = Volt_Input.qV_Component1 * (int32_t)Local_Vector_Components.hCos;
  qValpha_tmp2 = Volt_Input.qV_Component2 * (int32_t)Local_Vector_Components.hSin;

  Volt_Output.qV_Component1 = (int16_t)(((qValpha_tmp1)+(qValpha_tmp2))>>15);

  qVbeta_tmp1 = Volt_Input.qV_Component1 * (int32_t)Local_Vector_Components.hSin;
  qVbeta_tmp2 = Volt_Input.qV_Component2 * (int32_t)Local_Vector_Components.hCos;

  Volt_Output.qV_Component2 = (int16_t)((qVbeta_tmp2-qVbeta_tmp1) >>15);

  return(Volt_Output);
} ? end __attribute__ ?
```

Volt_Input.qV_Component1 为 Vq, Volt_Input.qV_Componen2 为 Vd , Theta为电角度

Valfa = Volt_Output.qV_Component1 =Vq*Cos(Theta)+ Vd*Sin(Theta)

Vbeta = Volt_Output.qV_Component2 =-Vq*Sin(Theta)+ Vd*Cos(Theta)

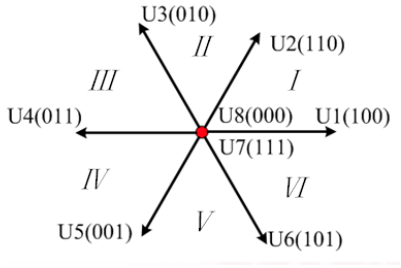
**PWMC_SetPhaseVoltage:**

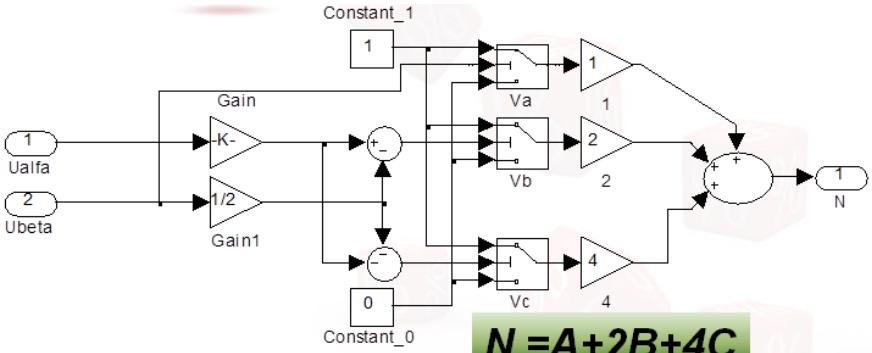

Clark反变换，与AN908不一致

$$\begin{cases} V_{r1} = U_\beta & V_{r1}>0,\text{则}A=1,\ \text{反之}A=0; \\ V_{r2} = \frac{\sqrt{3}}{2}U_\alpha - \frac{1}{2}U_\beta & V_{r2}>0,\text{则}B=1,\ \text{反之}B=0; \\ V_{r3} = -\frac{\sqrt{3}}{2}U_\alpha - \frac{1}{2}U_\beta & V_{r3}>0,\text{则}C=1,\ \text{反之}C=0。 \end{cases}$$

**N =A+2B+4C**

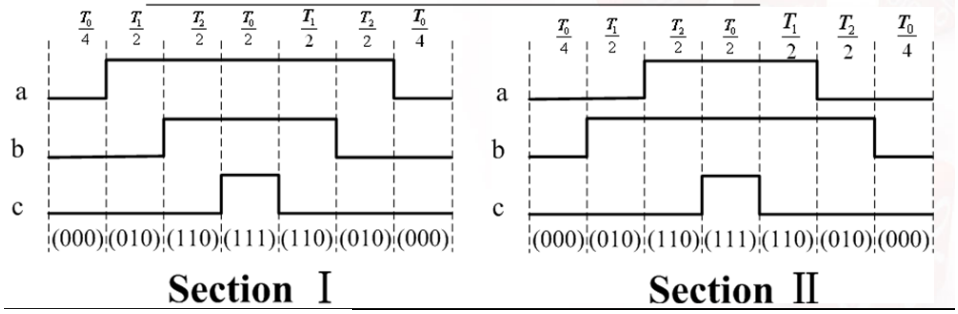
当N=3时，$U_{ref}$位于第Ⅰ扇区；
当N=1时，$U_{ref}$位于第Ⅱ扇区；
当N=5时，$U_{ref}$ 位于第Ⅲ扇区；
当N=4时，$U_{ref}$ 位于第Ⅳ扇区；
当N=6时，$U_{ref}$ 位于第Ⅴ扇区；
当N=2时，$U_{ref}$ 位于第Ⅵ扇区。




**N =A+2B+4C**

| 作用时间 / 扇区 | T0/4 零矢量 | T1/2 第一矢量 | T2/2 第二矢量 | T0/2 零矢量 | T2/2 第二矢量 | T1/2 第一矢量 | T0/4 零矢量 |
|---|---|---|---|---|---|---|---|
| Ⅰ | 000 | 100 | 110 | 111 | 110 | 100 | 000 |
| Ⅱ | 000 | 010 | 110 | 111 | 110 | 010 | 000 |
| Ⅲ | 000 | 010 | 011 | 111 | 011 | 010 | 000 |
| Ⅳ | 000 | 001 | 011 | 111 | 011 | 001 | 000 |
| Ⅴ | 000 | 001 | 101 | 111 | 101 | 001 | 000 |
| Ⅵ | 000 | 100 | 101 | 111 | 101 | 100 | 000 |




根据Valpha和Vbeta判断在哪个扇区
先将Valfa_beta转成X,Y,Z电压

```
wUAlpha = Valfa_beta.qV_Component1 * (int32_t)pVars->hT_Sqrt3;
wUBeta = -(Valfa_beta.qV_Component2 *
          (int32_t)(CLASS_PARAMS->hPWMperiod))*2;

wX = wUBeta;
wY = (wUBeta + wUAlpha)/2;
wZ = (wUBeta - wUAlpha)/2;
```

备注:

```
#define SQRT3FACTOR (uint16_t) 0xDDB4 /* = (16384 * 根号3 * 2) = (16384 * 1.732051 * 2)*/
pVars->hT_Sqrt3= (pParams->hPWMperiod*SQRT3FACTOR)/16384u =  (PWM周期* 16384 * 根号3 * 2)/16384 = PWM周期 * 根号3 * 2);
```

（2^15）*4 = 32768*4 = 131072 #define T (PWM_PERIOD * 4)，这里有一个4倍的放大。然后电流采用了Q15表示（左对齐），2^15 = 32768。所以最后计算需要除以131072。 #define divSQRT_3 (int32_t)0x49E6 /* 1/sqrt(3) in q1.15 format=0.5773315*/ Q15，也就是0.5773315*32768 = 18918

再根据wX,wY,wZ来决定当前是在什么扇区

```
/* Sector calculation from wX, wY, wZ */
if (wY<0)
{
  if (wZ<0)
  {
    pVars->hSector = SECTOR_5;
    wTimePhA = (int32_t)(CLASS_PARAMS->hPWMperiod)/4 + ((wY - wZ)/(int32_t)262144);
    wTimePhB = wTimePhA + wZ/131072;
    wTimePhC = wTimePhA - wY/131072;
    pSetADCSamplingPoint = CLASS_METHODS.pPWMC_SetADCSampPointSect5;
  }
  else /* wZ >= 0 */
    if (wX<=0)
    {
      pVars->hSector = SECTOR_4;
      wTimePhA = (int32_t)(CLASS_PARAMS->hPWMperiod)/4 + ((wX - wZ)/(int32_t)262144);
      wTimePhB = wTimePhA + wZ/131072;
      wTimePhC = wTimePhB - wX/131072;
      pSetADCSamplingPoint = CLASS_METHODS.pPWMC_SetADCSampPointSect4;

    }
    else /* wX > 0 */
    {
      pVars->hSector = SECTOR_3;
      wTimePhA = (int32_t)(CLASS_PARAMS->hPWMperiod)/4 + ((wY - wX)/(int32_t)262144);
      wTimePhC = wTimePhA - wY/131072;
      wTimePhB = wTimePhC + wX/131072;
      pSetADCSamplingPoint = CLASS_METHODS.pPWMC_SetADCSampPointSect3;

    }
} ? end if wY<0 ?
```

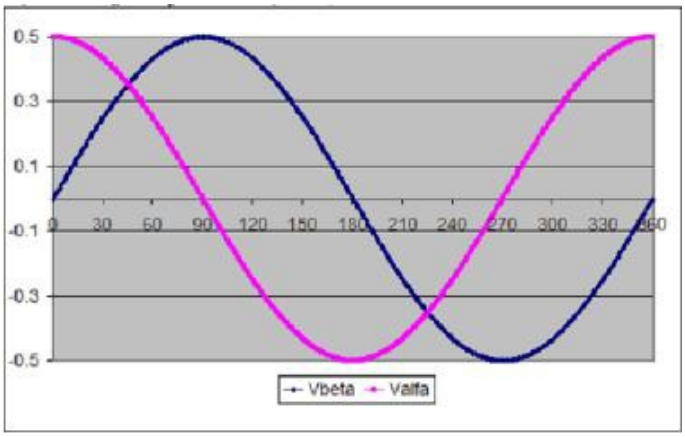
```
else /* wY > 0 */
{
  if (wZ>=0)
  {
    pVars->hSector = SECTOR_2;
    wTimePhA = (int32_t)(CLASS_PARAMS->hPWMperiod)/4 + ((wY - wZ)/(int32_t)262144);
    wTimePhB = wTimePhA + wZ/131072;
    wTimePhC = wTimePhA - wY/131072;
    pSetADCSamplingPoint = CLASS_METHODS.pPWMC_SetADCSampPointSect2;
  }
  else /* wZ < 0 */
    if (wX<=0)
    {
      pVars->hSector = SECTOR_6;
      wTimePhA = (int32_t)(CLASS_PARAMS->hPWMperiod)/4 + ((wY - wX)/(int32_t)262144);
      wTimePhC = wTimePhA - wY/131072;
      wTimePhB = wTimePhC + wX/131072;
      pSetADCSamplingPoint = CLASS_METHODS.pPWMC_SetADCSampPointSect6;
    }
    else /* wX > 0 */
    {
      pVars->hSector = SECTOR_1;
      wTimePhA = (int32_t)(CLASS_PARAMS->hPWMperiod)/4 + ((wX - wZ)/(int32_t)262144);
      wTimePhB = wTimePhA + wZ/131072;
      wTimePhC = wTimePhB - wX/131072;
      pSetADCSamplingPoint = CLASS_METHODS.pPWMC_SetADCSampPointSect1;

    }
} ? end else ?
```

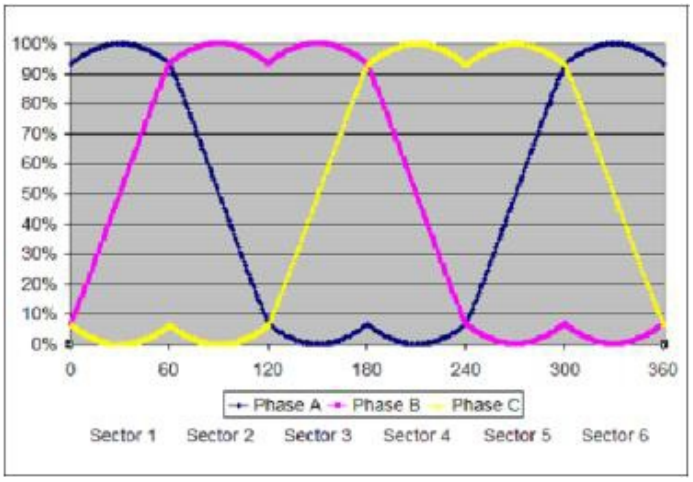
**参考文档：**

SVPWM原理详解.ppt
2016/06/16 10:47, 2.16MB

以及： STM32F103_永磁同步电机_PMSM_FOC软件库_用户手册_中文版.pdf




Vα 和Vβ定子电压




SVPWM 相电压波形

阐明每同步六相空间矢量区域对应的PWM。

定义如下：

$$U_\alpha = \sqrt{3} \times T \times V_\alpha, U_\beta = -T \times V_\beta$$

$$X = U_\beta, Y = \frac{U_\alpha + U_\beta}{2}, Z = \frac{U_\beta - U_\alpha}{2}$$

空间矢量区域:

| | Y < 0 | | | Y ≥ 0 | | |
|---|---|---|---|---|---|---|
| | Z < 0 | Z ≥ 0 | | Z < 0 | | Z ≥ 0 |
| | | X ≤ 0 | X > 0 | X ≤ 0 | X > 0 | |
| Sector | V | IV | III | VI | I | II |

通过下式关系分别计算PWM应用于A、B、C三相的正相脉冲宽度持续的时间：

Sector I, IV: $t_A = \frac{T + X - Z}{2}$, $t_B = t_A + Z$, $t_C = t_B - X$

Sector II, V: $t_A = \frac{T + Y - Z}{2}$, $t_B = t_A + Z$, $t_C = t_A - Y$

Sector III, VI: $t_A = \frac{T - X + Y}{2}$, $t_B = t_C + X$, $t_C = t_A - Y$

现在考虑PWM波形是中心对称，因此相电压必须集中到周期的50%，根据PWM
输出带负载的值，各自的参考电阻如下：

Sector I, IV: $\text{TimePhA} = \frac{T}{4} + \frac{T/2 + X - Z}{2}$, $\text{TimePhB} = \text{TimePhA} + Z$, $\text{TimePhC} = \text{TimePhB} - X$

Sector II, V: $\text{TimePhA} = \frac{T}{4} + \frac{T/2 + Y - Z}{2}$, $\text{TimePhB} = \text{TimePhA} + Z$, $\text{TimePhC} = \text{TimePhA} - Y$

Sector III,VI: $\text{TimePhA} = \frac{T}{4} + \frac{T/2 + Y - X}{2}$, $\text{TimePhB} = \text{TimePhC} + X$, $\text{TimePhC} = \text{TimePhA} - Y$

PWMC_SetPhaseVoltage
|_____ SVPWM变换得到
hCntPhA/B/C, 也就是三相的占空比
|_____
pSetADCSamplingPoint
|_____
R3HD2_SetADCSampPointSect1～6 设置ADC采样位置，然后设置占空比