使用TMS320C2812控制异步电机的程序，采用SVPWM空间矢量控制算法，运行正常，加减速，正反转等.

 首先，初始化设备，

/\*初始化系统\*/

 InitSysCtrl();

 /\*关中断\*/

 DINT;

 IER = 0x0000;

 IFR = 0x0000;

 /\*初始化PIE控制寄存器\*/

 InitPieCtrl();

 /\*初始化PIE矢量表\*/

 InitPieVectTable();

 /\*初始化SCIb寄存器\*/

 InitSci();

 /\*设置CPU定时器\*/

 InitCpuTimers();

 ConfigCpuTimer(&CpuTimer2, 150, 20000);

 StartCpuTimer2();

 /\*初始化IO口\*/

 InitGpio();

 /\*初始化EV\*/

 eva.Init(&eva);

 evb.Init(&evb);

下步，（个人习惯写个显示程序）

**void** **ShowDisp**(**void**) //显示

{

 **static** **unsigned** **int** i=0;

 **switch**(i)

 {

 **case** 0:

 i++;

 ScibRegs.SCITXBUF =(ku&0xf)+(3<<5);

 **break**;

 **case** 1:

 **if**(RunFlag) ScibRegs.SCITXBUF =23+(2<<5);

 **else** ScibRegs.SCITXBUF =24+(2<<5);

 i++;

 **break**;

 **case** 2:

 **if**(RunFlag) ScibRegs.SCITXBUF =f\_now/10+(1<<5);

 **else** ScibRegs.SCITXBUF =f\_given\_disp/10+(1<<5);

 i++;

 **break**;

 **case** 3:

 **if**(RunFlag) ScibRegs.SCITXBUF =f\_now%10;

 **else** ScibRegs.SCITXBUF =f\_given\_disp%10;

 i=0;

 **break**;

 **default**:

 i=0;

 **break**;

 }

下面再写，各功能模块：

1. 矢量计算和PWM生成

以下给出步骤1中的控制参数及其调节范围

EnableFlag：0、1；启停控制位

SpeedRef：（0~0.99）；速度给定值

VdTesting：（0~0.9）；D轴电流给定

VqTesting：（0~0.9）；Q轴电流给定





**void** **rampgen\_calc**(RAMPGEN \*v)

{

// Compute the angle rate

 v->Angle += \_IQmpy(v->StepAngleMax,v->Freq);

// Saturate the angle rate within (-1,1)

 **if** (v->Angle>\_IQ(1.0))

 v->Angle -= \_IQ(1.0);

 **else** **if** (v->Angle<\_IQ(-1.0))

 v->Angle += \_IQ(1.0);

// Compute the ramp output

 v->Out = \_IQmpy(v->Angle,v->Gain) + v->Offset;

// Saturate the ramp output within (-1,1)

 **if** (v->Out>\_IQ(1.0))

 v->Out -= \_IQ(1.0);

 **else** **if** (v->Out<\_IQ(-1.0))

 v->Out += \_IQ(1.0);

}

**void** **RotateVecotr\_calc**(RotateVecotr\_Handle v)

{

 \_iq Ua,Ub;

// Using look-up IQ sine table

 Ub = \_IQsinPU(v->Angle);

 Ua = \_IQcosPU(v->Angle);

 v->Ualpha = \_IQmpy(v->k,Ua);

 v->Ubeta = \_IQmpy(v->k,Ub);

}

**void** **scope**(**void**)

{

 **long** tl,tm,t0;

 tl = svpwm.tl;

 tm = svpwm.tm;

 t0 = ((**long**)1<<19) - tl - tm;

 **switch**(svpwm.vect)

 {

 **case** 2:ua = t0+tm; ub = t0; **break**;

 **case** 3:ua = 0; ub = tl; **break**;

 **case** 1:ua = t0; ub = ((**long**)1<<19); **break**;

 **case** 5:ua = tl; ub = tl+tm; **break**;

 **case** 4:ua = ((**long**)1<<19); ub = tm+t0; **break**;

 **case** 6:ua = tl+tm; ub = 0; **break**;

 **default**: **break**;

 }

 uab=ua-ub;

}

2、这就是传说中的精华所在：电流、直流母线电压、速度测试

**void** **svgendq\_calc**(SVGENDQ \*v)

{

 \_iq Va,Vb,Vc,t1,t2;

 **unsigned** **long** Sector = 0; // Sector is treated as Q0 - independently with global Q

// Inverse clarke transformation

 Va = v->Ubeta;

 Vb = \_IQmpy(\_IQ(-0.5),v->Ubeta) + \_IQmpy(\_IQ(0.8660254),v->Ualpha); // 0.8660254 = sqrt(3)/2

 Vc = \_IQmpy(\_IQ(-0.5),v->Ubeta) - \_IQmpy(\_IQ(0.8660254),v->Ualpha); // 0.8660254 = sqrt(3)/2

// 60 degree Sector determination

 **if** (Va>\_IQ(0))

 Sector = 1;

 **if** (Vb>\_IQ(0))

 Sector = Sector + 2;

 **if** (Vc>\_IQ(0))

 Sector = Sector + 4;

// X,Y,Z (Va,Vb,Vc) calculations

 Va = v->Ubeta; // X = Va

 Vb = \_IQmpy(\_IQ(0.5),v->Ubeta) + \_IQmpy(\_IQ(0.8660254),v->Ualpha); // Y = Vb

 Vc = \_IQmpy(\_IQ(0.5),v->Ubeta) - \_IQmpy(\_IQ(0.8660254),v->Ualpha); // Z = Vc

 **if** (Sector==0) // Sector 0: this is special case for (Ualpha,Ubeta) = (0,0)

 {

 v->Ta = \_IQ(0.5);

 v->Tb = \_IQ(0.5);

 v->Tc = \_IQ(0.5);

 }

 **if** (Sector==1) // Sector 1: t1=Z and t2=Y (abc ---> Tb,Ta,Tc)

 {

 t1 = Vc;

 t2 = Vb;

 v->Tb = \_IQmpy(\_IQ(0.5),(\_IQ(1)-t1-t2)); // tbon = (1-t1-t2)/2

 v->Ta = v->Tb+t1; // taon = tbon+t1

 v->Tc = v->Ta+t2; // tcon = taon+t2

 }

 **else** **if** (Sector==2) // Sector 2: t1=Y and t2=-X (abc ---> Ta,Tc,Tb)

 {

 t1 = Vb;

 t2 = -Va;

 v->Ta = \_IQmpy(\_IQ(0.5),(\_IQ(1)-t1-t2)); // taon = (1-t1-t2)/2

 v->Tc = v->Ta+t1; // tcon = taon+t1

 v->Tb = v->Tc+t2; // tbon = tcon+t2

 }

 **else** **if** (Sector==3) // Sector 3: t1=-Z and t2=X (abc ---> Ta,Tb,Tc)

 {

 t1 = -Vc;

 t2 = Va;

 v->Ta = \_IQmpy(\_IQ(0.5),(\_IQ(1)-t1-t2)); // taon = (1-t1-t2)/2

 v->Tb = v->Ta+t1; // tbon = taon+t1

 v->Tc = v->Tb+t2; // tcon = tbon+t2

 }

 **else** **if** (Sector==4) // Sector 4: t1=-X and t2=Z (abc ---> Tc,Tb,Ta)

 {

 t1 = -Va;

 t2 = Vc;

 v->Tc = \_IQmpy(\_IQ(0.5),(\_IQ(1)-t1-t2)); // tcon = (1-t1-t2)/2

 v->Tb = v->Tc+t1; // tbon = tcon+t1

 v->Ta = v->Tb+t2; // taon = tbon+t2

 }

 **else** **if** (Sector==5) // Sector 5: t1=X and t2=-Y (abc ---> Tb,Tc,Ta)

 {

 t1 = Va;

 t2 = -Vb;

 v->Tb = \_IQmpy(\_IQ(0.5),(\_IQ(1)-t1-t2)); // tbon = (1-t1-t2)/2

 v->Tc = v->Tb+t1; // tcon = tbon+t1

 v->Ta = v->Tc+t2; // taon = tcon+t2

 }

 **else** **if** (Sector==6) // Sector 6: t1=-Y and t2=-Z (abc ---> Tc,Ta,Tb)

 {

 t1 = -Vb;

 t2 = -Vc;

 v->Tc = \_IQmpy(\_IQ(0.5),(\_IQ(1)-t1-t2)); // tcon = (1-t1-t2)/2

 v->Ta = v->Tc+t1; // taon = tcon+t1

 v->Tb = v->Ta+t2; // tbon = taon+t2

 }

3、这个工程还没做完，先上这么多。

4、这块是上面初始化部分的，模块化程序，凑个字数，嘻嘻！！

**void** **InitXintf**(**void**)

{

 **#if** F2812

 // Example of chaning the timing of XINTF Zones.

 // Note acutal values should be based on the hardware

 // attached to the zone - timings presented here are

 // for example purposes.

 // All Zones:

 // Timing for all zones based on XTIMCLK = SYSCLKOUT

 XintfRegs.XINTCNF2.bit.XTIMCLK = 0x0000;

 // Zone 0:

 // Change write access lead active trail timing

 // When using ready, ACTIVE must be 1 or greater

 // Lead must always be 1 or greater

 // Use timings based on SYSCLKOUT = XTIMCLK

 XintfRegs.XTIMING0.bit.XWRTRAIL = 3;

 XintfRegs.XTIMING0.bit.XWRACTIVE = 7;

 XintfRegs.XTIMING0.bit.XWRLEAD = 3;

 // Do not double lead/active/trail for Zone 0

 XintfRegs.XTIMING0.bit.X2TIMING = 0;

 // Zone 2

 // Ignore XREADY for Zone 2 accesses

 // Change read access lead/active/trail timing

 XintfRegs.XTIMING2.bit.USEREADY = 0;

 XintfRegs.XTIMING2.bit.XRDLEAD = 3;

 XintfRegs.XTIMING2.bit.XWRACTIVE = 7;

 XintfRegs.XTIMING2.bit.XRDTRAIL = 3;

 // Double lead/active/trial timing for Zone 2

 XintfRegs.XTIMING2.bit.X2TIMING = 1;

 // Zone 2 is slow, so add additional BCYC cycles when ever switching

 // from Zone 2 to another Zone. This will help avoid

 // bus contention.

 XintfRegs.XBANK.bit.BANK = 2;

 XintfRegs.XBANK.bit.BCYC = 3;

 **#endif**

}