单电阻 FOC 资料整理　lcwswust　20260803

参考资料：

《无感无刷直流电机之电调设计全攻略》

《PMSM 的无传感器磁场定向控制（AN1078）》

《PMSM 无传感器 FOC 的单分流三相电流重构算法（AN1299）》

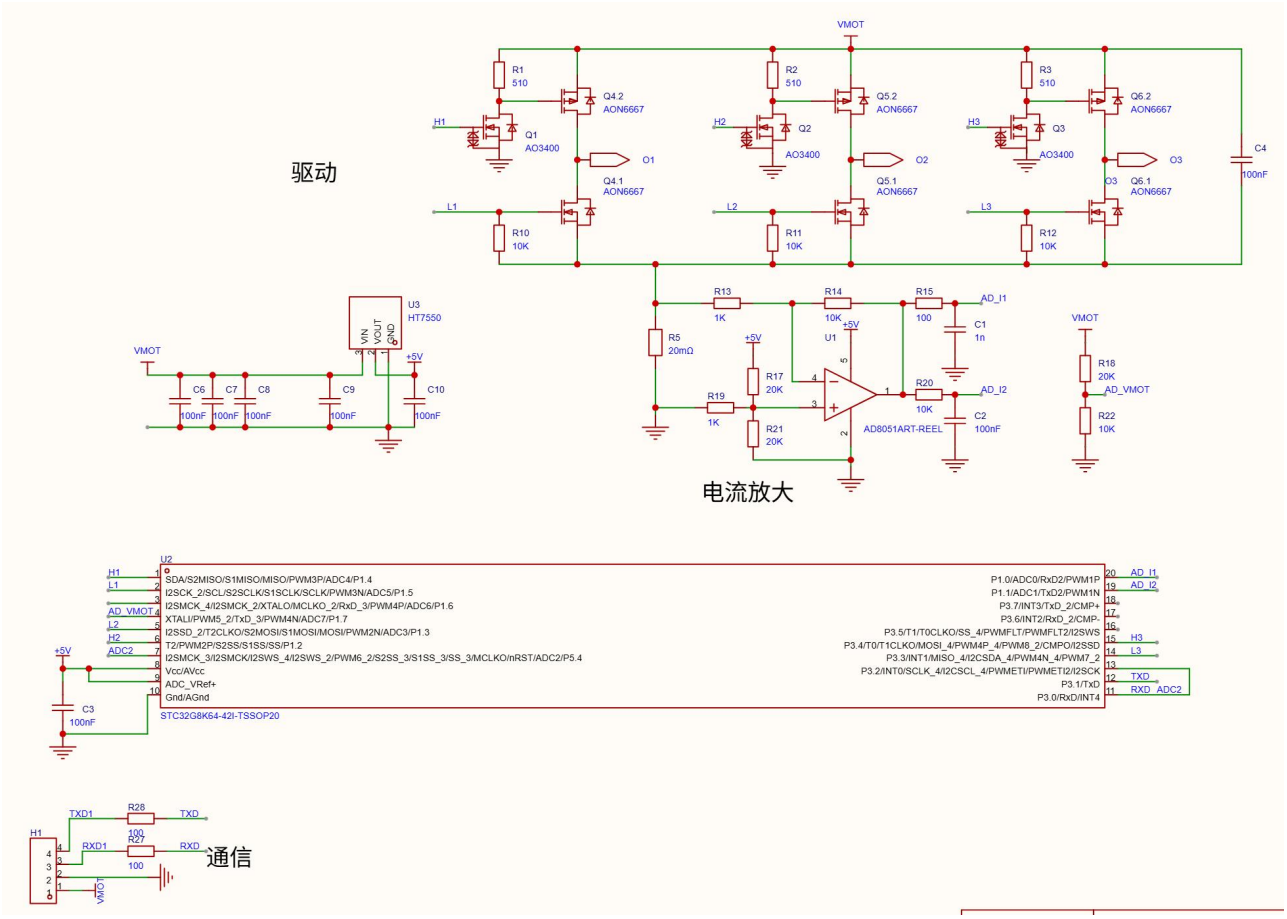
<分享一个低成本 STC8H1K08 无刷小电调，支持 oneshot125 协议>

https://www.stcaimcu.com/thread-17391-1-1.html

<便宜的车规-FOC，24 万转@STC32G12K128-24A 车规，STC-FOC 新算法库,4/1 更新>

https://www.stcaimcu.com/thread-19545-1-1.html

1.原理图

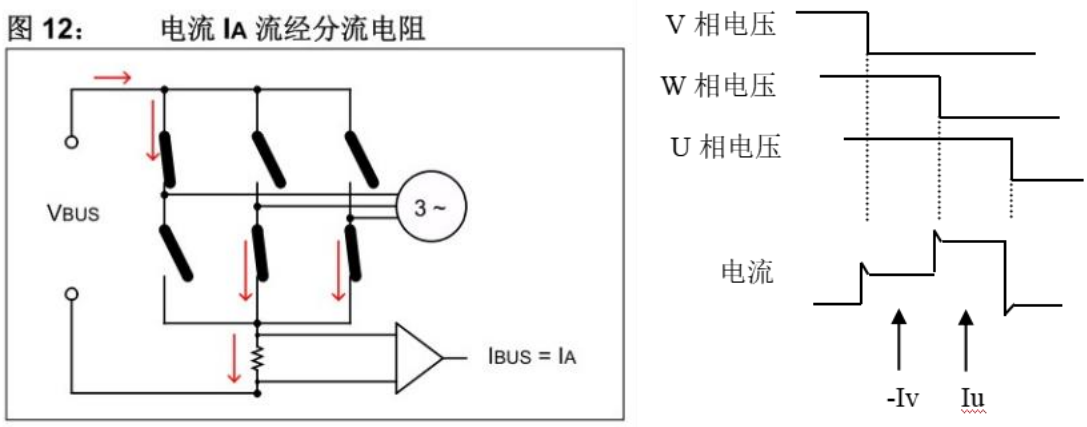


MCU 为 STC32G8K64，利用其带死区的互补 PWM 功能，驱动 N 沟道+P 沟道的 MOS 管，母线电流经运放放大后进 ADC 采样。注意，电流放大电路为反相放大，计算电流时要注意极性。为加快边沿、减小死区，建议 R1、R2、R3 尽量小。

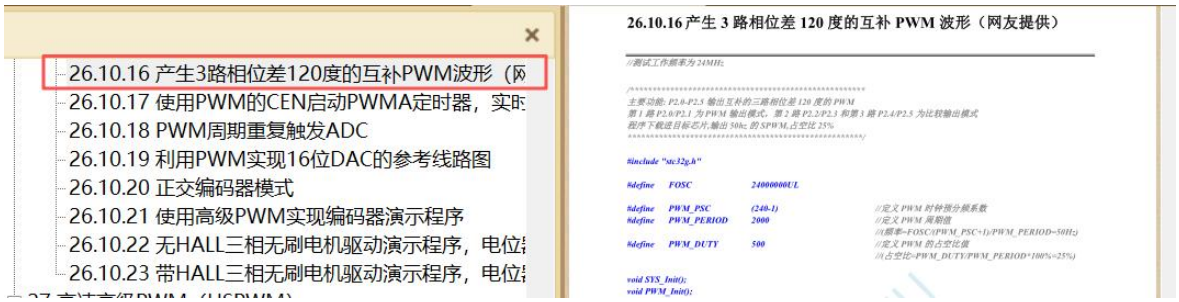
2.三相电流重构

原理参考文档 AN1299，我这里采用了一个笨办法：

移相，使三相 PWM 的后沿相差固定的 2us 左右的时间；为了使得 ADC 采样时的电平是固定的，需要限制最低脉宽和最高脉宽，如下图：



移相原理参考《STC32G 系列技术手册》中的例程“20.10.16 产生 3 路相差 120 度的互补 PWM 波形（网友提供）”，一路 PWM 工作于“PWM 模式 1”，另两路工作于“翻转模式”。代码位于“main.c\void pwma_sr(void) interrupt 26”。



采集电流使用了 DMA，启动 DMA 的时机较为重要，建议调试时先设置固定的相电压或开环转动，进行验证，电流采准了才好进行后续工作。

3.反电势估算

引用文档 AN1078，假如有一只普通的直流电机，已知电阻 R、电感 L、供电电压 Vs，根据这些参数就能估算出电流，再根据估算电流与实际检测电流 i_s 之差，可以估算出感应电动势 e_s。在无刷电机里，这个模型相当于其中一相，当得到两个相的感应电动势之后，就可以估算出转子（磁铁）的位置了，然后通过驱动电路产生合适的磁场令其转动。

AN1078

PMSM 的无传感器 FOC

该算法中最为重要的一部分就是如何计算 FOC 需要的换相角度。本应用笔记的这一部分将说明估算换相角度（θ）和电机速度 （ω）的过程。

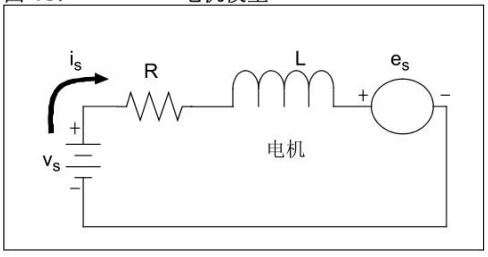
无传感器控制技术通过估算电机的位置（不使用位置传感器）来实现 FOC 算法。图 16 为位置估算器的简化框图。

电机的位置和速度是根据测量电流和计算电压估算出的。

电机模型

通过使用一个直流电机模型来估算 PMSM 的位置，该电机模型由绕组电阻、绕组电感和反电动势来表示，如图 15 所示。

图 15：　电机模型



在该电机模型中，输入电压可由公式 1 计算得到。

在我的程序中，R 与 L 是自动计算的，对绕组施加固定的电压，通过前两个电流之差得到 L，通过一段时间后的稳定电流得到 R（类似于阶越响应）；代码位于“stcfoclcw.c\ void MotInit(void)”。

stcfoclcw.c

```
190     //------------------------------获取电机参数
191     //指定相电压
192     Mot_AMP=50;//
193     Mot_State=2;//取得电机参数
194     delayms(100);//由于电机会略有转动，等长一点时间后重新采集数据
195     Mot_State=0;//停止
196     delayms(1); //等到续流为0后再开始采集
197     adcnt=0;
198     while(adcnt<2);//用于同步AD采样
199     adcnt=0;//开始采集数据
200     Mot_State=2;//取得电机参数
201     while(Mot_Icnt<ADCBUFSIZE);
202     Mot_State=0;
203
204     //由于PWM滞后1个周期，从下标1开始算
205     Mot_R=(U32)Mot_AMP*500/Mot_Iubuf[4];//R=V/i，电阻放大500倍
206     if(Mot_Iubuf[2]<=Mot_Iubuf[1])//防止除数为0 或为负
207         Mot_Iubuf[2]=Mot_Iubuf[1]+1;
208     Mot_L=(U32)Mot_AMP*500/(Mot_Iubuf[2]-Mot_Iubuf[1]);//L=V/(di/dt)电感放大500倍
209     //-1,171,194,58,51,
210     //Mot_L=((U32)Mot_AMP*500-(U32)Mot_R*Mot_Iubuf[3])/(Mot_Iubuf[2]-Mot_Iubuf[1])
211     if(Mot_L<1)Mot_L=1;//防止除数为0
212     //-3,169,193,8,51,
213     //用于查看计算值
214     adbuf[0][ADCBUFSIZE-1]=Mot_Iubuf[1];
215     adbuf[1][ADCBUFSIZE-1]=Mot_Iubuf[2];
216     adbuf[2][ADCBUFSIZE-1]=Mot_Iubuf[4];
217     adbuf[3][ADCBUFSIZE-1]=Mot_L;//在上位机中查看此数据
```

文档 AN1078 中，对感应电动势的估算流程如下图：

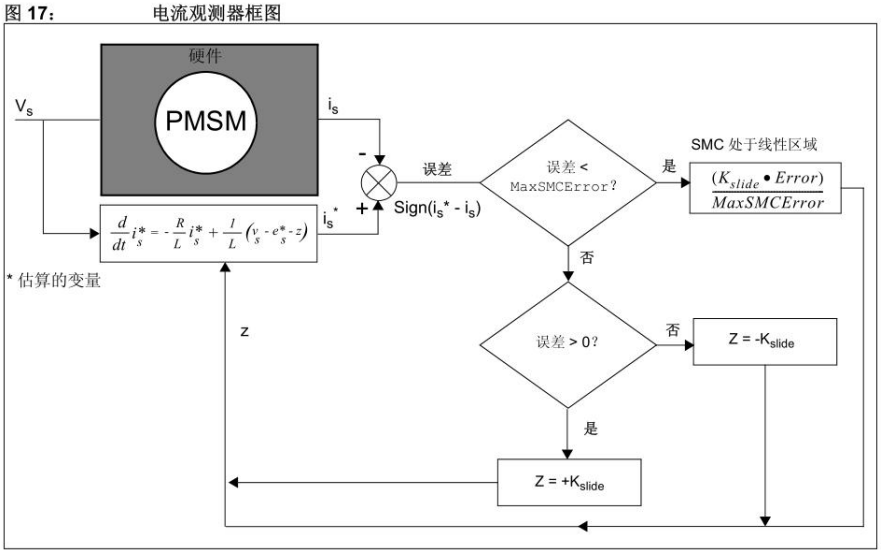
AN1078

该数字化模型对硬件使用了软件表示方式。然而，为了使测量电流和估算电流相匹配，数字化电机模型需要使用闭环控制来进行校正，如图 17 所示。

考虑用两种方式表示电机，一种是硬件方式（阴影区），一种是软件方式，两个系统中使用相同的输入电压（v_s），使用模型中的估算电流（i_s*）来匹配测量电流（i_s），我们假设数字化模型的反电动势（e_s*）与电机的反电动势（e_s）相同。

注： * 代表估算的变量。

图 17： 电流观测器框图

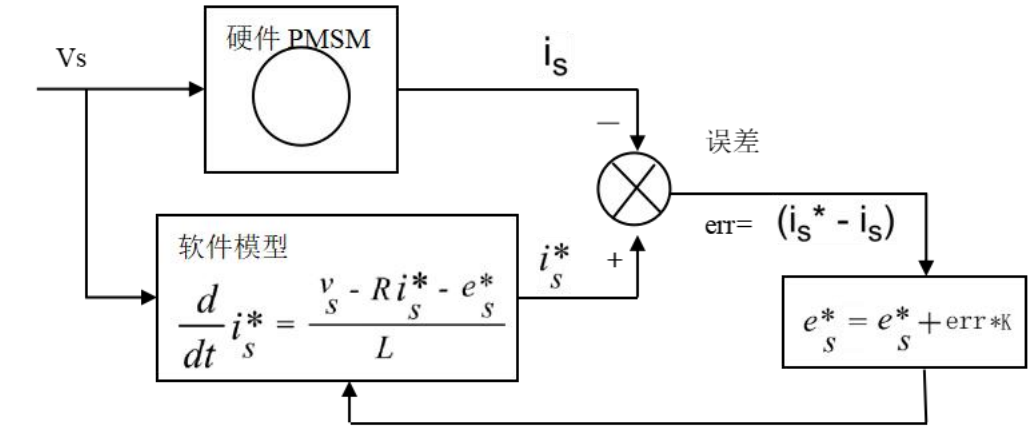


滑动模式控制器（Slide Mode Controller，SMC）在 UserParms.h 中定义了一个限制值 MaxSMCError。当误差值小于 MaxSMCError 时，滑动模式控制器的输出工作在图 17 中 PMSM 框下的公式所给定的线性范围内。

反电动势估算

对数字化模型进行补偿之后，电机模型的输入电压（V_s）和电流（i_s*）与数字化电机模型中的值相同。一旦对数字化模型补偿完后，下一步就要通过对校正因子（Z）滤波来估算反电动势（e_s*），如图 18 所示。随后

而我觉得这里有点复杂，有限幅，有校正因子 Z，我的程序中作了简化，原理如下图：



*估算的变量

代码：

// Mot_IestU 为 U 相电流估算值

// Mot_IU 为 U 相电流检测值

// Mot_EmfU 为 U 相估算感应电动势

err=Mot_IestU-Mot_IU;//如果估算电流大于实测电流，则感应电动势应增加

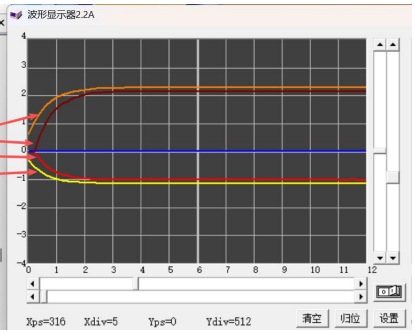
Mot_EmfU+=err*Mot_Kslide;

Mot_IestU+=((S32)Vu*500-Mot_IestU*Mot_R-Mot_EmfU)/Mot_L;//电感两端电压/电感

验证的代码及效果如下图：

```
stcfoclcw.c
219     SendADbuf();//发给上位机查看波形
220     //while(1);
221     //------------------------------验证电感、电阻计算公式
222     Mot_AMP=40;//
223     Mot_State=2;//取得电机参数
224     delayms(100);//由于电机会略有转动，等长一点时间后重新采集数
225     Mot_State=0;//停止
226     delayms(1);
227     adcnt=0;
228     while(adcnt<2);//用于同步AD采样
229     adcnt=0;//开始采集数据
230     Mot_State=2;//取得电机参数
231     while(Mot_Icnt<ADCBUFSIZE);
232     Mot_State=0;
233     SendADbuf();
```

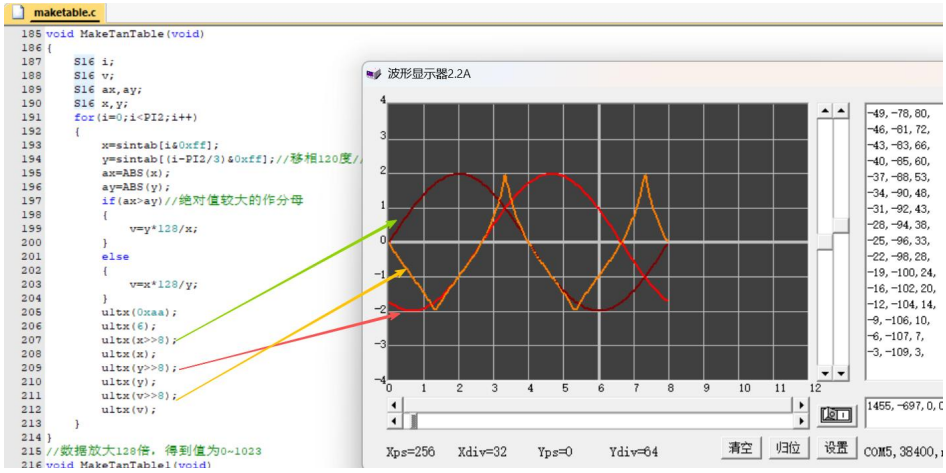
```
main.c
454             Mot_EmfVlp=(Mot_EmfVlp+Mot_EmfV)/2;
455         }
456         //------------------------------波形监测
457         if(adcnt<ADCBUFSIZE)
458         {
459             adbuf[0][adcnt]=Mot_IU;//ad_v_zp_v;//adc_dma
460             adbuf[1][adcnt]=Mot_IV;//ad_u_zp_u;//adc_dma
461             adbuf[2][adcnt]=Mot_IestU;//
462             adbuf[3][adcnt]=Mot_IestV;//
463             adbuf[4][adcnt]=Mot_EmfU>>8;//Mot_EmfDeglp;//
464             adbuf[5][adcnt]=Mot_EmfV>>8;//Mot_Speed;//Mot
465             adcnt++;
466         }
467         PTEST=0;
468         /*
469         mot_vp+=mot_step;
```



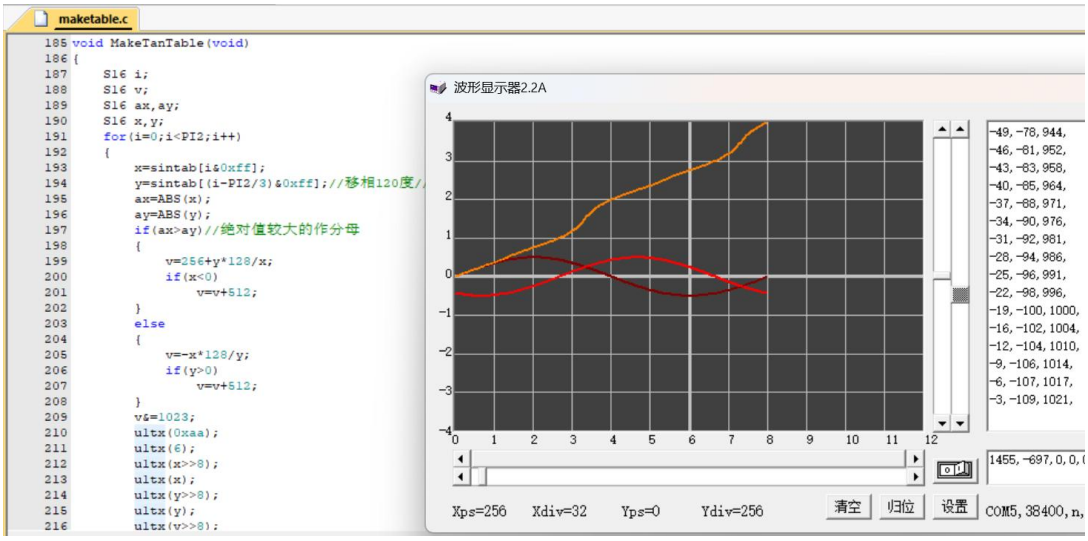
4.转子位置估算

通过上面的算法，得到 U 相与 V 相的感应电动势（Mot_EmfU、Mot_EmfV），经低通滤波（Mot_EmfUlp、Mot_EmfVlp）,再经类似 C 语言 atan2(反正切)的算法后，即可得到转子位置（角度）。由于两个相电压之间的相位差是 120 度，而不是 90 度，所以不能直接用 atan2 计算；一种方法是利用 Clarke 变换(AN1078 中有说明)将其变为 90 度的相位差后用 atan2 计算，另一种方法是手搓查表法。我这里用的就是“另一种方法”，实现过程如下：

(1)对于相差 120 度的波形，作除法，以绝对值大的一方作分母，放大 128 倍，得到结果如橙色曲线；



(2)想办法让橙色曲线单调上升，如取负值，加偏移量，得到如同 tan(正切)的效果；

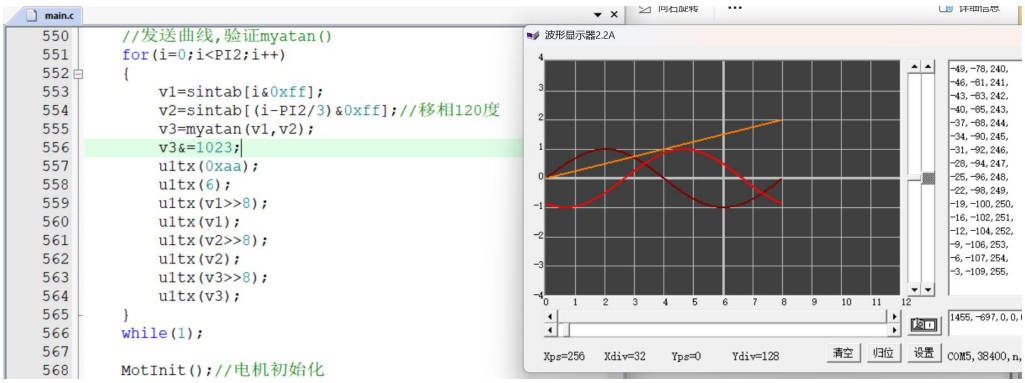


(3)把橙色曲线存入数组 tantab[]，并反向查表，即可得到类似 atan 的效果，得到数组 atantab[]；

maketable.c*

```
254         }
255     }
256     ultxstr("\r\n};\r\n");
257 }
258 //利用MakeTanTable生成的表格，作反查
259 void MakeAtanTable(void)
260 {
261     S16 i,n;
262     ultxstr("\r\nconst u8 atantab[1024]={\r\n");
263     for(i=0;i<1024;i++)
264     {
265         //查询i在tantab中位于哪个区间，如同NTC查表转换为温度
266         for(n=0;n<(PI2-1);n++)
267         {
268             if((i>=tantab[n])&&(i<tantab[n+1]))
269                 break;
270         }
271         SendN10(n);//数据转字符串
272         ultxstr(",");
273         if((i%16)==15)//每16个数据换一行
274         {
275             ultxstr("\r\n");
276         }
277     }
278     ultxstr("\r\n};\r\n");
279 }
280
```

(4)验证算法,myatan()函数输入相差 120 度的两个波形，得到 0~255 的一条斜线，效果不错；



5.转起来；

先开环转动、加速，达到一定速度后进入闭环，根据感应电动势经 myatant()得到的转子位置 Mot_EmfDeglg，指定输出电压角度（Mot_Vp）；经测，输出电压角度随转速的提高而越发的提前于转子位置，程序中加了一点关于转速的处理。

```
399             #warning "debug"
400             //速度/补偿，35/10,90/20,190/41
401             //set_deg=19+Mot_Speed/8;//转速越快，角度越超前,8.7
402             set_deg=Mot_Speed*10/45;
403             if(set_deg>60)set_deg=60;//加以限制
404             if(err_v>set_deg)err_v--;//渐变到64
405             if(err_v<set_deg)err_v++;//set_deg最小64，速度越快越大，最大110
406         }
407
408         Mot_Vp=(Mot_EmfDeglp+err_v)&0xff;//指定输出角度
409     }
```

产生三相电压时，作了一点处理，统一下移并限制最小占空比。

```
423             //------------------产生三相电压
424             p=Mot_Vp;
425             Vu=Vpwm*sintab[p]/256;
426             p-=256/3;//360/3=120度
427             Vv=Vpwm*sintab[p]/256;
428             Vw=0-Vu-Vv;
429             //求三相最小电压值
430             Vmin=Vu;
431             if(Vmin>Vv)Vmin=Vv;
432             if(Vmin>Vw)Vmin=Vw;
433             Vmin-=MOTPWMMIN;//统一下移并限制最小占空比
434             SETPWM3(Vu-Vmin,Vv-Vmin,Vw-Vmin);//MOTPWMMIN~MOTPWMMAX
```