MC SDK 5.x 中增加位置环

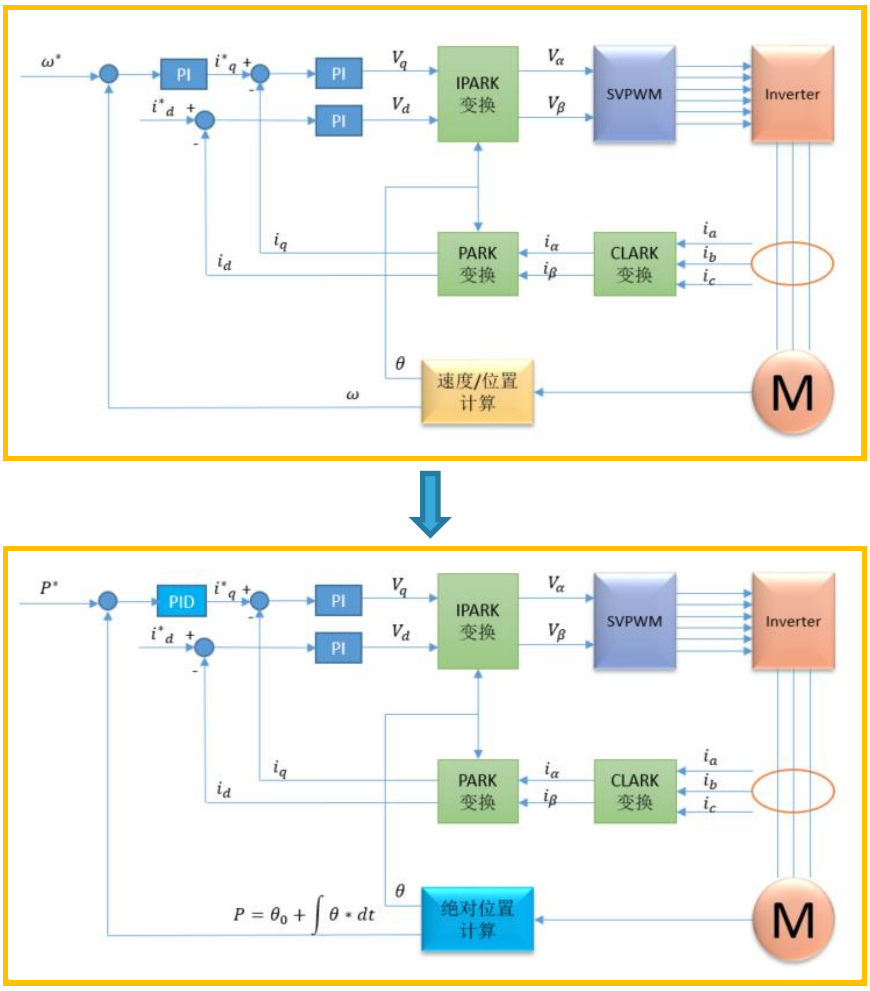
前言

在 MC SDK 5.x 中有两个环路控制，分别是电流环和速度环，有些应用需要使用到位置环，比如无人机的云台电机控制，地铁闸机等等应用，本文将阐述如何在现有 ST 电机库基础上增加简易的位置环。

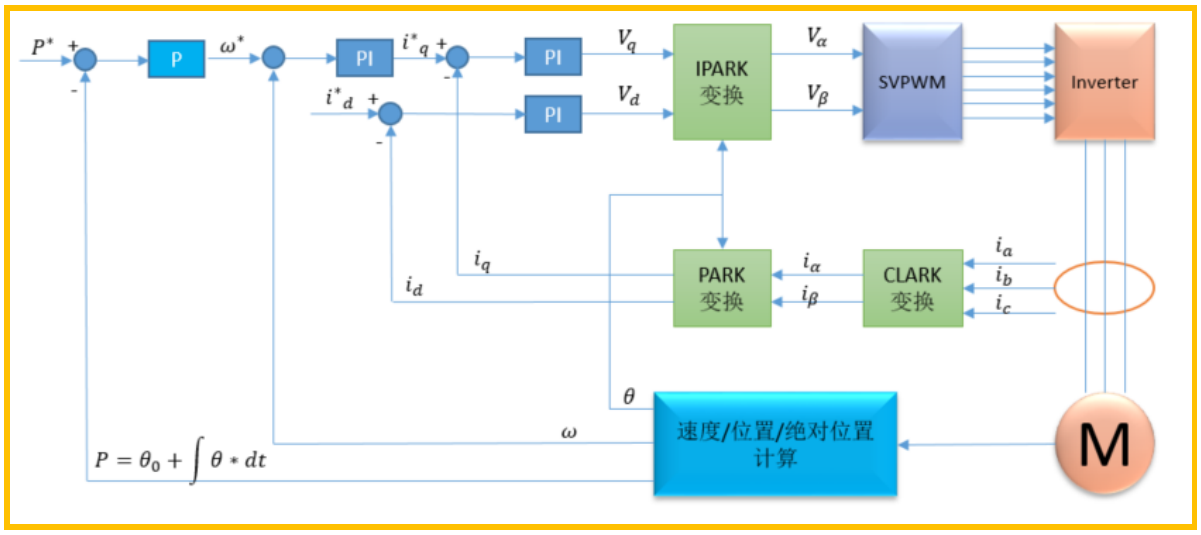
1. 理论基础

以表贴电机为例，一般的 FOC 电机控制是两环控制，速度环+电流环的控制方式，速度环为外环，电流环为内环，Idref = 0（d 轴电流参考为 0）的控制为常见控制；参考速度环输出参考转矩后供后端电流环路；

如果增加位置环最简单和直接的方式即为将速度环换为位置环，即变为位置环+电流环的方式：



如果在有减速齿轮电机上，需要加入速度环以便更快速的到达给定位置，也就是三环控制：位置环+速度环+电流环，框图如下：



为了方便控制，以及考虑精度问题，角度单位为 rad*10000，比如电机转动 10 圈，也就是 10 圈后的位置，则该位置的设定角度为：

$$\theta_{set} = 10 * 2\pi * 10000 \approx 628320$$

位置环采用的是线性系数乘以系数直接输出到速度的参考:

$$\omega_{ref} = K_p * (P_s - P_c)$$

Kp–位置环系数

Ps –设定的位置

Pc–当前的位置

当前位置的角度数字量计算如下：

$$\theta_c = Cnt * 2 * PI * 10000 + \theta_d$$

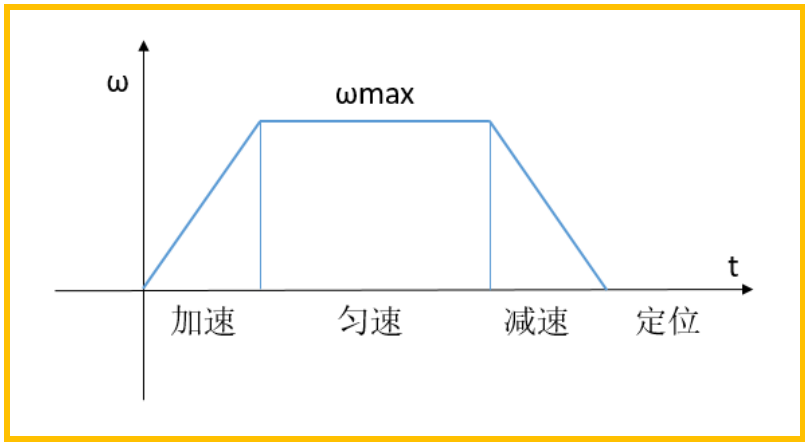
Cnt –绝对值编码器转过的整圈数

θ_d–当前的机械角度的数字量

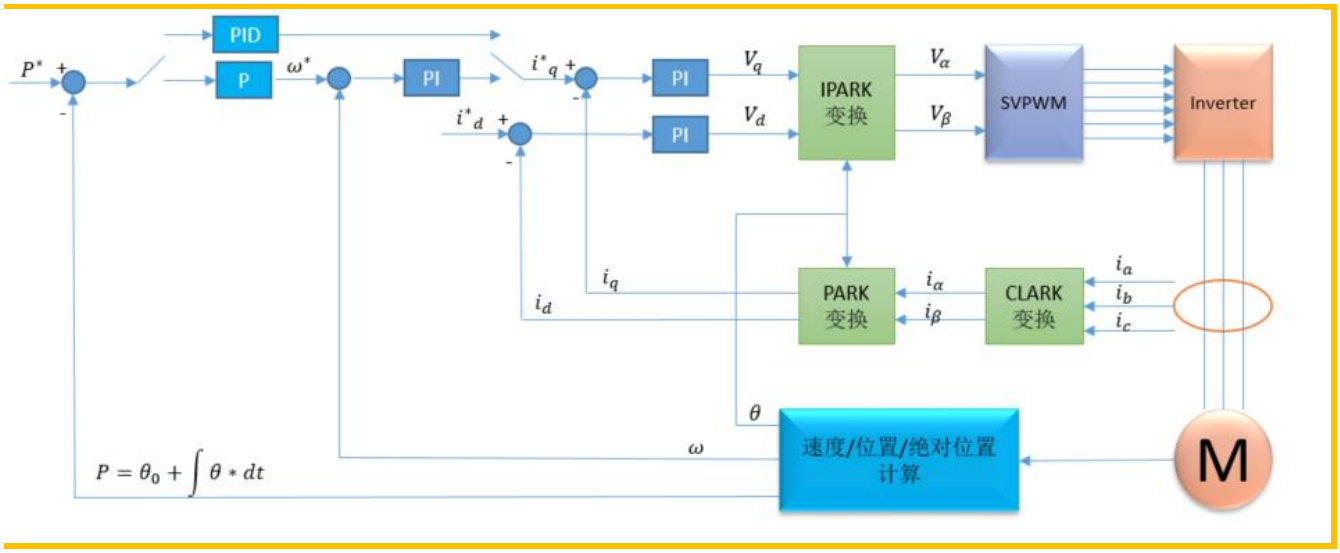
PI –取值 3.1416

2. 位置环控制改进

在三环控制中，不可避免的会涉及到加速度的计算，正常控制过程为：加速→匀速→减速→定位过程



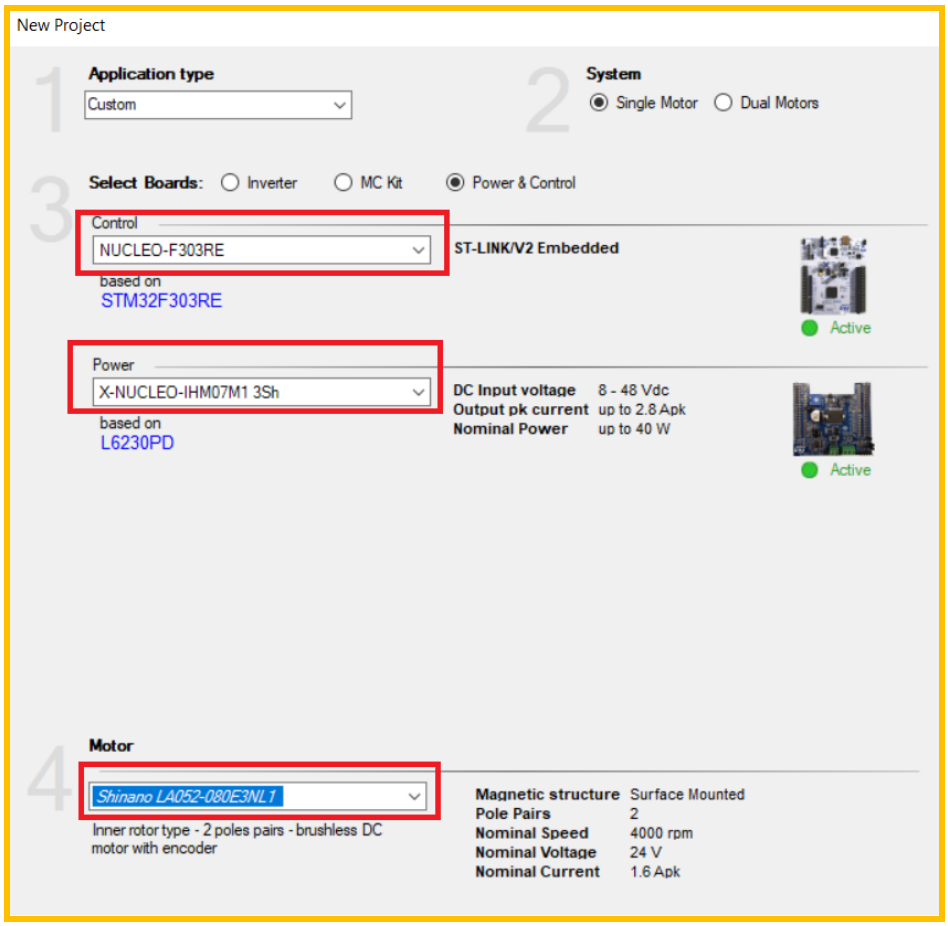
那么会涉及到加速度以及定位阶段的控制，这边我们可以按照加速度与速度差成反比，而定位阶段可以直接设定速度为 0，但是实际使用过程中我们会发现直接直接的速度模式在定位阶段力量有限，方法一：PID 需要特别设定，更能够定位到给定位置，方法二：抛开速度环，直接力矩控制，实际控制框图如下：



3. 具体操作步骤

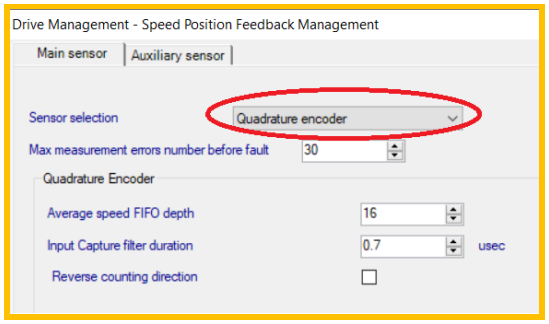
3. 1 创建编码器可以正常运行的工程

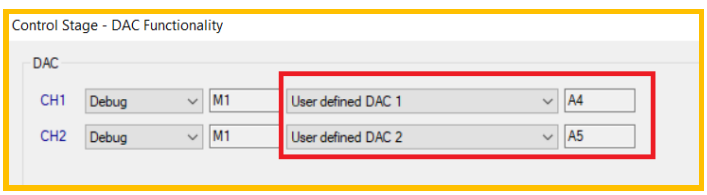
首先要建立可以正常运行的位置传感器为编码器（ENCODER）的电机工程，本文所使用的是 NUCLEO-F303RE（控制板）+ IHM07M1（功率板） + Shinano LA052-080E3NL1（电机）

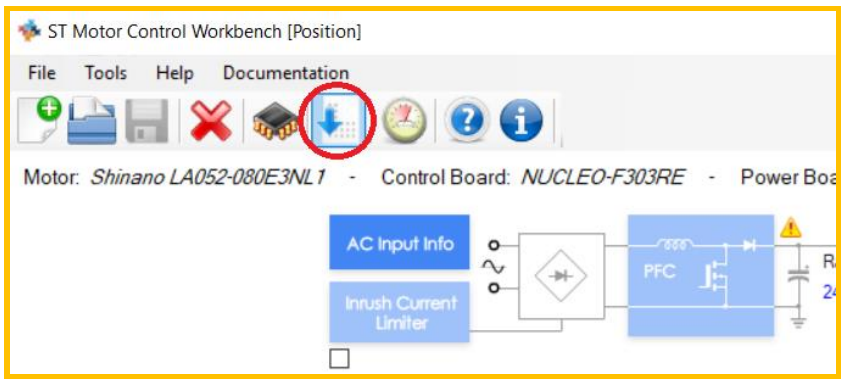


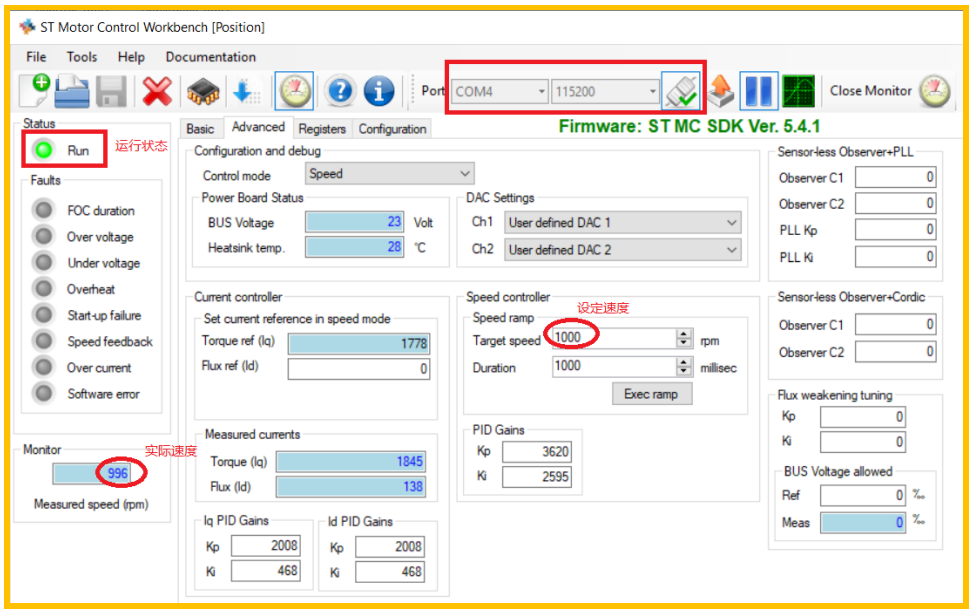
3. 2 生成工程并调试运行

产生编码器工程并且保证电机正常运行，为后续动作做准备，同时配置 DAC 的两路 User 数据输出









3. 3 增加以下变量或函数用于位置环控制

1) 位置环 PID 结构体 PID_Handle_t PIDAngleHandle_M1
2) 位置控制的结构体 Position_Handle_t
3) 位置角度误差 Position_GetErrorAngle
4) 位置环速度参考输出计算 Position_CalcSpeedReferrence
5) 位置环力矩参考输出计算 Position_CalcTorqueReferrence

具体参考 mc_position.h 以及 mc_position.c

3.4 mc_task.c 文件修改

3.4.1 增加头文件

mc_task.c 中包含头文件

```
#include "mc_position.h"
```

3.4.2 初始化位置环 PID 参数

MCboot()函数中增加角度 PID 初始化函数

```
      /* if unmount this, there will be not torque under motor stand still */
      PID_HandleInit(&PIDAngleHandle_M1); // Initial position PID
```

3.4.3 定位后初始测量角度清零

```
case ALIGNMENT:
    if ( !EAC_Exec( &EncAlignCtrlM1 ) )
    {
      ……
    }
    else
    {
      ……
       /* USER CODE BEGIN MediumFrequencyTask M1 EndOfEncAlignment */
      Position_M1.hMeasuredAngle = 0;
      Position_M1.Encoder_Update_Count = 0;
       /* USER CODE END MediumFrequencyTask M1 EndOfEncAlignment */

      STM_NextState( &STM[M1], ANY_STOP );
    }
    break;
```

3.4.4 增加 DAC 输出测试程序

函数 TSK_HighFrequencyTask()中增加用户 DAC 输出信号更新数据，将目标角度（位置）以及测量角度（位置）通过 DAC 输出到示波器

```
      /* DAC output*/
      extern UI_Handle_t * pDAC;
      static int16_t temp1;
      static int16_t temp2;

      temp1 = Position_M1.hTargetAngle;
      temp2 = Position_M1.hMeasuredAngle;


      DAC_SetUserChannelValue(pDAC, DAC_USER1, temp1);
```

```
      DAC_SetUserChannelValue(pDAC, DAC_USER2, temp2);
```

3.4.5 增加位置控制程序

在 mc_task.c 的中频任务函数 TSK_MediumFrequencyTaskM1 中增加位置环差值数据计算，根据差值计算，当差值在阈值之上的话进行速度控制，否则则进行力矩控制，在 RUN 下调用下面代码：

```
case RUN:
    /* USER CODE BEGIN MediumFrequencyTask M1 2 */
#if defined(POSITION_CONTROL)
            /* Get error position/Angle, unit in rad*10000 */
            Position_GetErrorAngle(Target_Angle);

            /* If in speed control mode */
            if(Position_M1.Mode_Flag == P_SPEED_MODE)
            {
                  Position_M1.Set_Speed = Position_CalcSpeedReferrence();
            }

            /* Change mode to current mode */
            if((Position_M1.Mode_Flag == P_TORQUE_MODE) && (Position_M1.Torque_First_Flag
== 0))
            {
                  qd_t temp_qd;
                  /* Get current Iqref value */
                  temp_qd = MC_GetIqdrefMotor1();
                  MC_ProgramTorqueRampMotor1(temp_qd.q,0);
                  Position_M1.Torque_First_Flag = 1;
                  PID_SetIntegralTerm(&PIDAngleHandle_M1,0);
            }

            Position_M1.Speed_ACC_Count++;
            /* In speed control mode, calculate speed accelarate time, aslo excute speed
ramp */
            if((Position_M1.Speed_ACC_Count > SPEED_ACC_RATE) && (Position_M1.Mode_Flag
== P_SPEED_MODE))
            {
                  Position_M1.Speed_ACC_Count = 0;
                  Position_M1.Control_Speed = MC_GetLastRampFinalSpeedMotor1(); // Get
setting speed value
                  Position_M1.Real_Speed = MC_GetMecSpeedAverageMotor1();   // Get
average speed
                  Position_M1.Duration_ms =
(uint16_t)(Abs_Value(Position_M1.Control_Speed - Position_M1.Real_Speed) *
Position_M1.Speed_ACC_Gain / Position_M1.Speed_ACC_Div);


      MC_ProgramSpeedRampMotor1(Position_M1.Set_Speed/6,Position_M1.Duration_ms);
```

```
            }
#endif
    /* USER CODE END MediumFrequencyTask M1 2 */

    MCI_ExecBufferedCommands( oMCInterface[M1] );
    FOC_CalcCurrRef( M1 );


    /* USER CODE BEGIN MediumFrequencyTask M1 3 */

    /* USER CODE END MediumFrequencyTask M1 3 */
    break;
```

3.4.6 修改参考电流计算

如果差值缩小到一定范围内，则进行力矩控制，在函数 FOC_CalcCurrRef 中进行调用

```
__weak void FOC_CalcCurrRef(uint8_t bMotor)
{

      if(FOCVars[bMotor].bDriveInput == INTERNAL)
      {
      #if defined(POSITION_CONTROL)
            /* If in position torque mode, Iqref come from Angle PID*/
            if(Position_M1.Mode_Flag == P_TORQUE_MODE)
            {
                  FOCVars[bMotor].hTeref = Position_CalcTorqueReferrence();
                  FOCVars[bMotor].Iqdref.q = FOCVars[bMotor].hTeref;
            }
            else
      #endif
            {
                  FOCVars[bMotor].hTeref = STC_CalcTorqueReference(pSTC[bMotor]);
                  FOCVars[bMotor].Iqdref.q = FOCVars[bMotor].hTeref;
            }

     }
}
```

3. 5 电机累计圈数计算

stm32f30x_mc_it.c 文件的 SPD_TIM_M1_IRQHandler 函数中增加编码器位置累加程序用于计算电机转动的圈数，用于多圈马达转动控制；

```
      if((ENCODER_M1.TIMx->CR1 & 0x10) == 0x10)
      {
            Position_M1.Encoder_Direction = 1;
      }
      else
      {
```

```
            Position_M1.Encoder_Direction = 0;
      }
      /* If dirction is same*/
      if(Position_M1.Encoder_Direction == Position_M1.Encoder_Pre_Direction)
      {
            Position_M1.Encoder_Update_Count++;
      }
      else
      {
            Position_M1.Encoder_Update_Count--;
      }
```

4. 增加测试代码

在 main 函数中 while(1)中增加测试代码，我们先用 0—90 度电机一圈内的转动来测定角度控制 PID

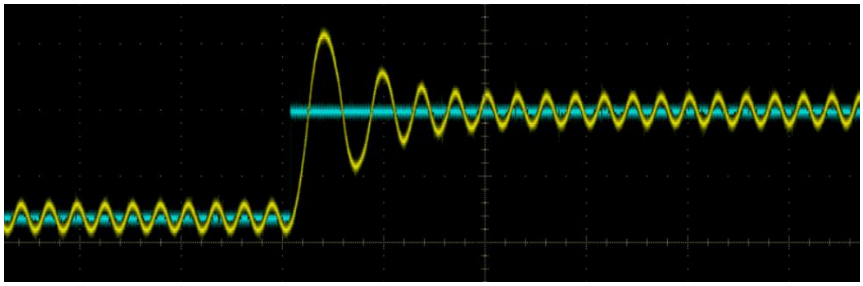
```
  while (1)
  {
      HAL_Delay(20000);
      Target_Angle = 0; // 0 degree

/* unit in rad*10000, 3141600 here means 3141600/10000/2/PI = 50 round(Mec)*/
//     Target_Angle = 3141600;
      HAL_Delay(20000);
/*If test Torque + Position mode, need set Target angle less than(628320)*/
      Target_Angle = 15708; // 90 degree
  }
```

5. 测定角度 PID 参数

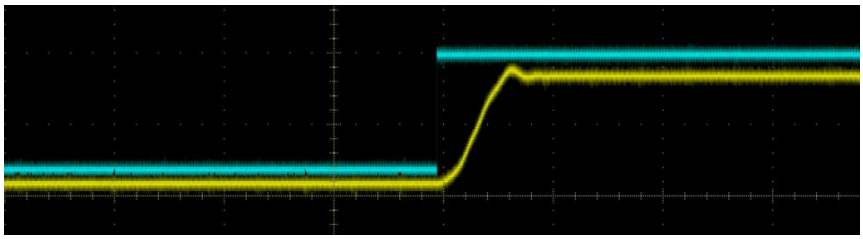
可以使用 DAC 将参考电角度和实际电角度输出，观察角度的变化，调节位置环的 PID 参数，从而达到快速准确的位置定位；定位实现后可以手动炫动电机，看到电机依然可以正常回归到给定位置。数据输出范围限制，可以使用 0 到 90 度来回位置变动来测试；

下面几张图是调整不同位置 PI 参数得到的不同运行结果，同样是由 0 度角变动为 90 度角的过程；

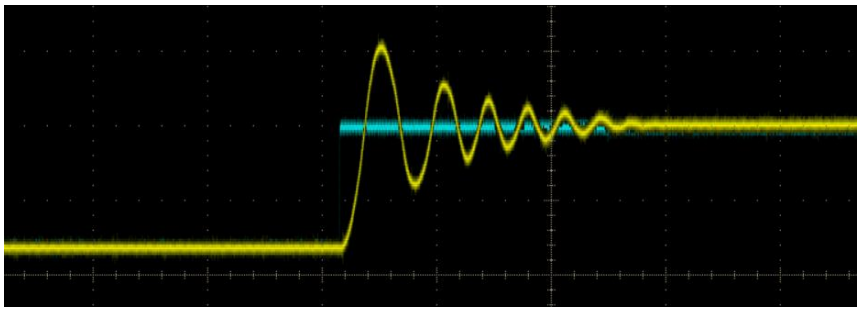
Kp = 10000/4096，Ki = 0/32768, 波形震荡状态



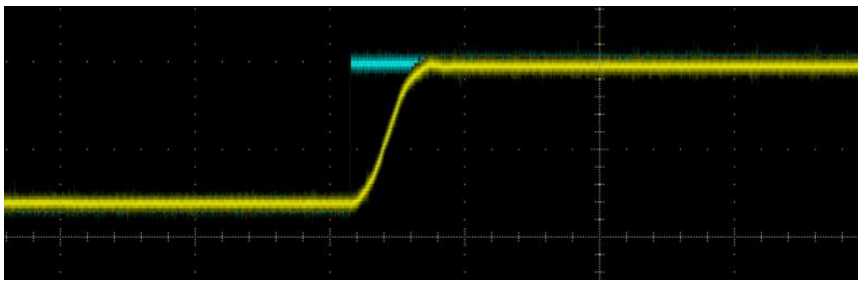
Kp = 100/4096, Ki = 0/32768，实际电机位置明显滞后



Kp = 3000/4096, Ki = 2/32768，波形震荡后收敛



Kp = 3000/4096, Ki = 10/32768，基本上当前负载下基本稳定



6. 必要时增加微分项调节

如果需要快速调节，需要增加微分项，也就是 PID 调节，库中的微分项默认是屏蔽的，因此，需要修改 pid_regulator.c 中的代码，打开 PID 函数，如下图：

pid_regulator.c

```
397  }
398
399  #if 1
400  #if defined (CCMRAM)
401  #if defined (__ICCARM__)
402  #pragma location = ".ccmram"
403  #elif defined (__CC_ARM) || defined(__GNUC__)
404  __attribute__( ( section ( ".ccmram" ) ) )
405  #endif
406  #endif
407  /**
408   * @brief  This function compute the output of a PID regulator sum of its
409   *         proportional, integral and derivative terms
410   * @param  pHandle: handler of the current instance of the PID component
411   * @param  wProcessVarError: current process variable error, intended as the
412   *         reference value minus the present process variable value
413   * @retval PID computed output
414   */
415
416  __weak int16_t PID_Controller( PID_Handle_t * pHandle, int32_t wProcessVarError )
417  {
418    int32_t wDifferential_Term;
419    int32_t wDeltaError;
420    int32_t wTemp_output;
421
422    if ( pHandle->hKdGain != 0 ) /* derivative terms not used */
423    {
424      wDeltaError = wProcessVarError - pHandle->wPrevProcessVarError;
425      wDifferential_Term = pHandle->hKdGain * wDeltaError;
```

7. 电机转动多圈位置环测试

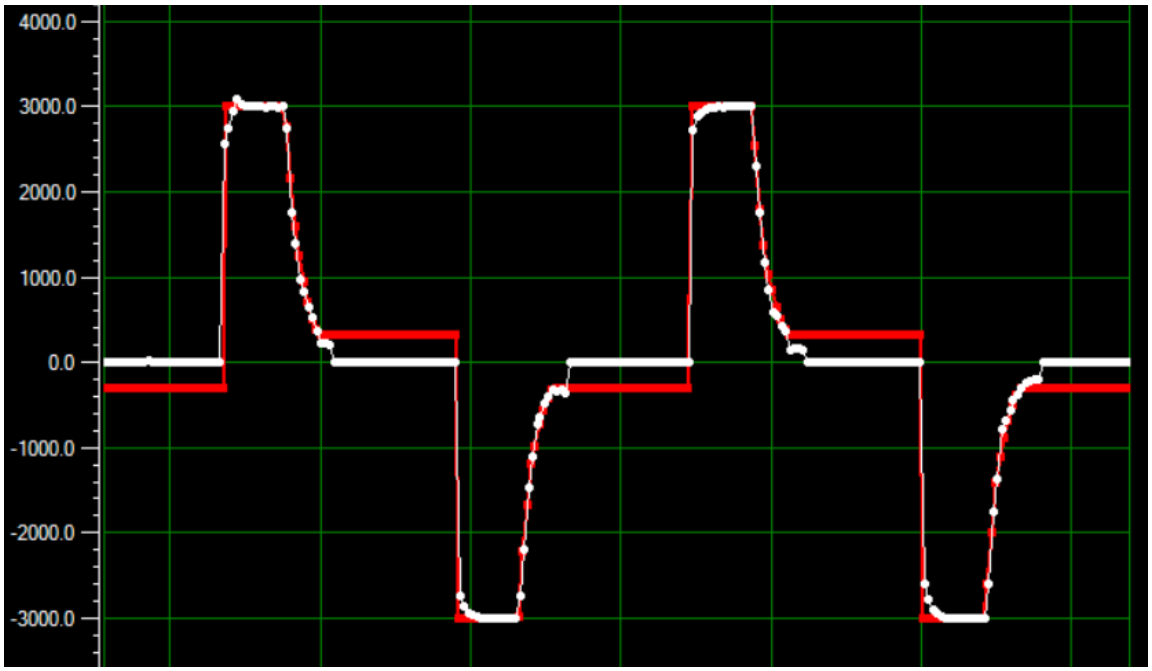
将位置设定为 0—50*360 度（电机转动 50 圈）的位置变动

```
  while (1)
  {
      HAL_Delay(20000);
      Target_Angle = 0; // 0 degree

/* unit in rad*10000, 3141600 here means 3141600/10000/2/PI = 50 round(Mec)*/
      Target_Angle = 3141600;
      HAL_Delay(20000);
  }
```

实际运行效果

位置偏差较大时，电机加速到最大速度，偏差小时将进入力矩控制模式，直到目标位置，在目标位置直接力矩锁定。

设定位置角度值在 0←→3141600 (电机转动 50 圈) 实际速度与参考速度曲线



扩展说明

1，本文中使用的是相对位置编码器，如果实际过程中所使用的是绝对位置编码器，则需要增加读取绝对位置编码器的程序；

2，如果是线性位置运动，则位置参量需要修改为角度所对应的线性位置，从而进行有效运算；

3，对于 PID 调节部分本文只是根据实际波形 DAC 输出进行调整，理论以及有赖于后期研究；

4，三环控制环路，两两环路调整比较好些，调好后再进行三环测试。

5，本文的位置输出到速度的 Kp 值没细致考量，这个部分可根据实际应用自行调整。

重要通知 – 请仔细阅读

意法半导体公司及其子公司（“ST”）保留随时对ST 产品和/ 或本文档进行变更、更正、增强、修改和改进的权利，恕不另行通知。买方在订货之前应获取关于ST 产品的最新信息。ST 产品的销售依照订单确认时的相关ST 销售条款。

买方自行负责对ST 产品的选择和使用， ST 概不承担与应用协助或买方产品设计相关的任何责任。

ST 不对任何知识产权进行任何明示或默示的授权或许可。

转售的ST 产品如有不同于此处提供的信息的规定，将导致ST 针对该产品授予的任何保证失效。

ST 和ST 徽标是ST 的商标。所有其他产品或服务名称均为其各自所有者的财产。

本文档中的信息取代本文档所有早期版本中提供的信息。

© 2015 STMicroelectronics – 保留所有权利