《自动控制实践 B》
综合实验

任务书

2024.5

目录

任务一 电机控制库认知

1.1 任务描述

通过“预习视频”，对意法半导体的电机控制库 5.x 版本，即英文缩写“ST MC SDK5.x ”建立基本的认知，了解 ST MC SDK5.x 的总体概况与软件。在此基础上，通过 workbench5.2 生成综合实验基础工程，并熟悉通过 workbench 的监测 UI 进行面板控制、参数调试、波形监测的过程。

1.2 任务分析

根据“预习视频”我们了解到，ST MotorControl Workbench（以下简称为 workbench）是意法半导体公司为电机控制库 5.x 版本开发的上位机应用软件。主要的功能是根据电机与驱动板配置参数，快速生成开发者需要的基于 HAL 库的 Cubemx 基础工程和 Keil 基础工程。

在本任务中，我们对于 workbench 的参数配置不做过多要求。同学们通过“预习视频”简要了解一下各个部分的功能即可。在实验中，我们将直接基于“硬石”开发板提供的 workbench 配置文件，“YSF4_HAL_MOTOR-654.FOC_v5.2.0_42PMSM_Ecoder.stmcx”生成基础工程。该文件所在位置如下图所示。

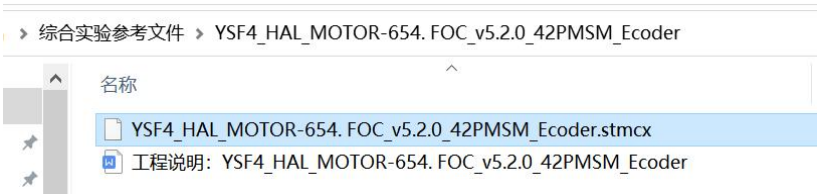


图 1.1 配置文件位置

生成基础工程之后，则可以在 workbench 的监控 UI 界面，控制电机的转动，还可以通过参数实时交互界面，对控制参数进行整定，使速度指令的跟踪效果更好。因为这部分的实验是认知性实验，所以对于参数调试，我们不做指标要求，同学们了解一下通过 workbench UI 进行电机控制、参数调试、曲线监测的过程即可。

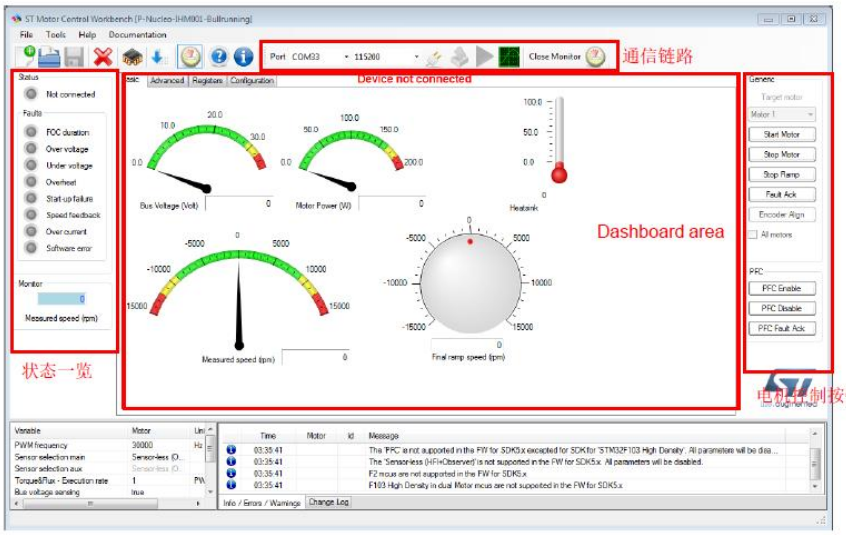


图 1.2 Workbench UI

1.3 任务实现

任务实现参考步骤如下。

1) 使用 USB 转 Mini USB 线连接电脑与开发板，因为后面所建立的基础工程使用 USART1 作为与上位机通信的接口，所以需要使用跳线连接 JP2，JP2 的 TXD、RXD 到 USRT1 的 PB6、PB7。
2) 在桌面上建立一个工程文件夹用来保存后面的工程，可以命名为“My_HAL_MOTOR-654. FOC_v5.2.0_42PMSM_Ecoder”。
3) 打开“综合实验参考文件”文件夹，将“YSF4_HAL_MOTOR-654. FOC_v5.2.0_42PMSM_Ecoder.stmcx”文件复制到新建工程文件夹下面。重命名为“My_HAL_MOTOR-654. FOC_v5.2.0_42PMSM_Ecoder”。

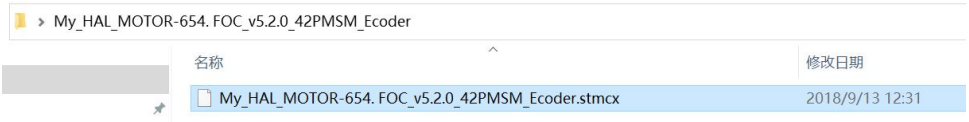


图 1.3 新建工程文件夹

4) 打开“MotorControl Workbench 5.2.0”应用软件，选择“打开工程”。

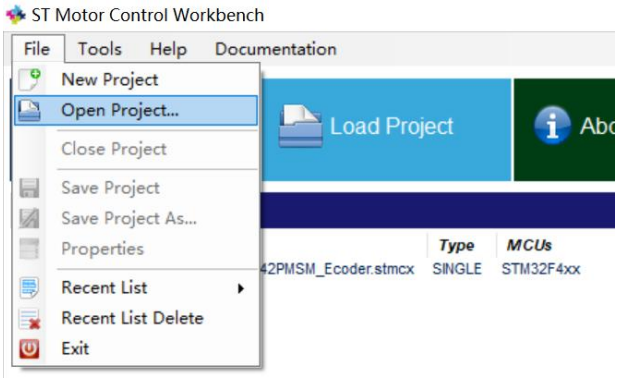


图 1.4 打开 Workbench 工程

5) 选择“My_HAL_MOTOR-654. FOC_v5.2.0_42PMSM_Ecoder”文件夹下的 stmcx 文件。

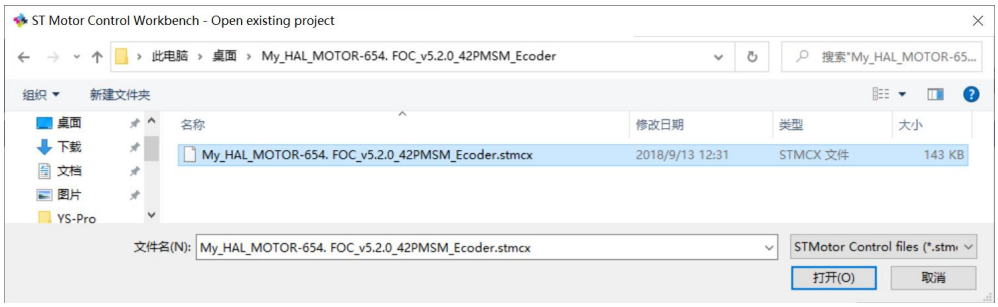


图 1.5 选择 stmcx 文件

6) 可以看到 workbench 中已经配置好的电机、功率板与控制参数。

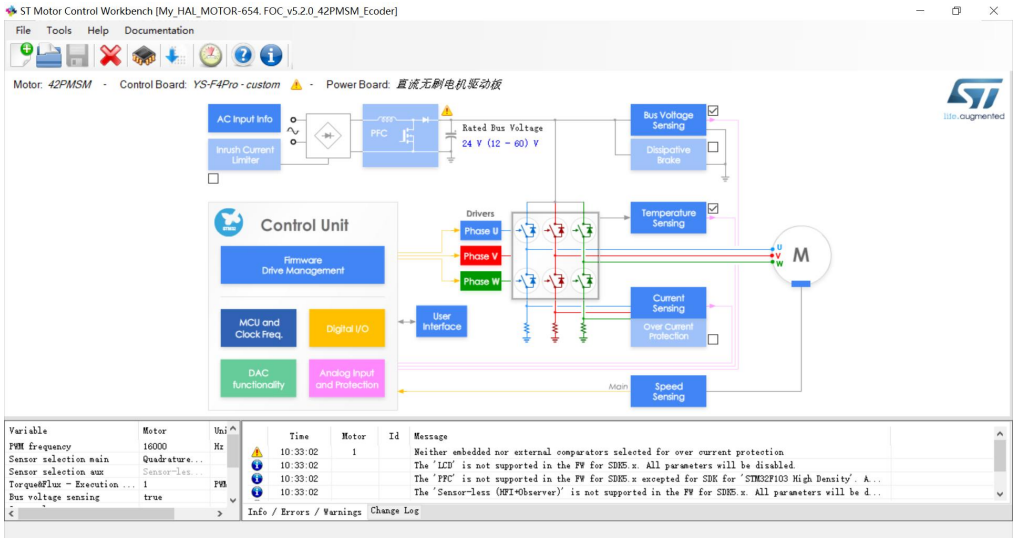


图 1.6 Workbench 参数配置

7) 点击图中的模块可以查看各个部分的参数配置，注意我们采用的速度测量方

式为编码器反馈。

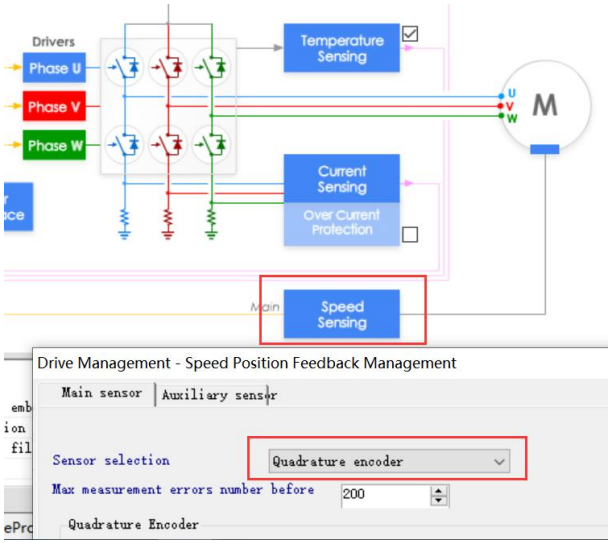


图 1.7 速度测量采用编码器

8) 点击“Firmware Drive Management”，查看“Start Up Parameter”部分参数配置。“Start Up Parameter”参数配置影响着编码器对齐功能。编码器对齐功能是在电机初始运行前首先要完成的，没有编码器对齐会使得电机无法判断正方向，导致“飞车”现象出现。注意“Final Current Ramp Value”值的配置，在该参数配置过小的情况下，编码器对齐功能会失效。“1”为经验值，请同学们结合所用设备实际运行情况调整该值。

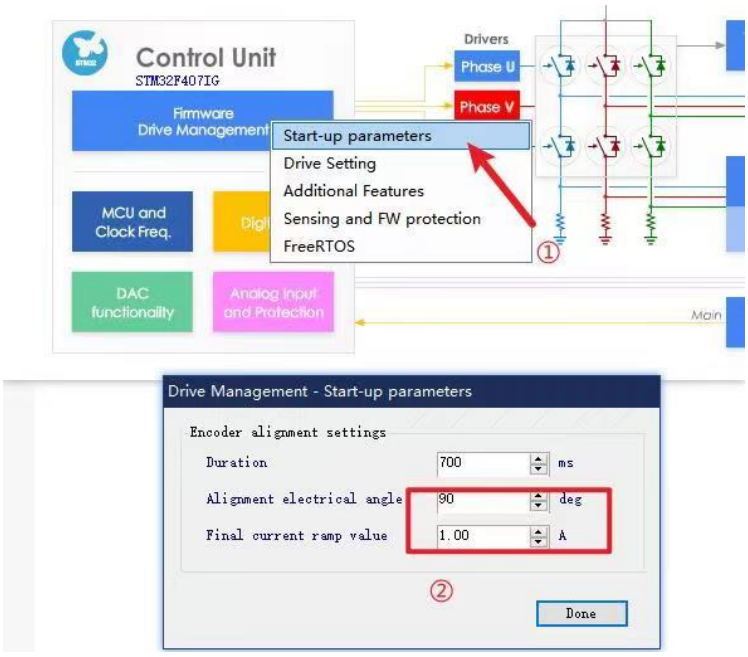


图 1.8 编码器对齐参数配置

9) 点击工具栏中的“generation”，生成电机控制库基础工程。

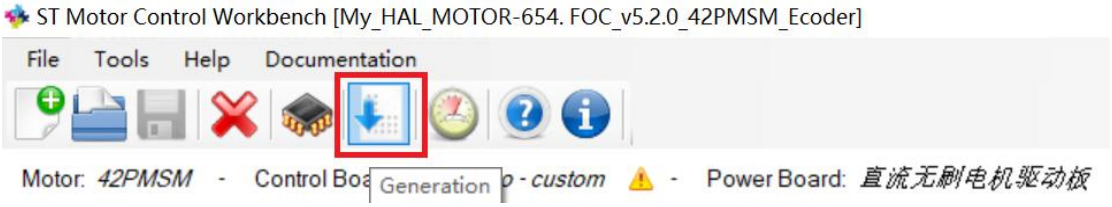


图 1.9 Workbench 工具栏

10) 在弹出的“Project generation”窗口配置如下，配置完成点击“Generate”。

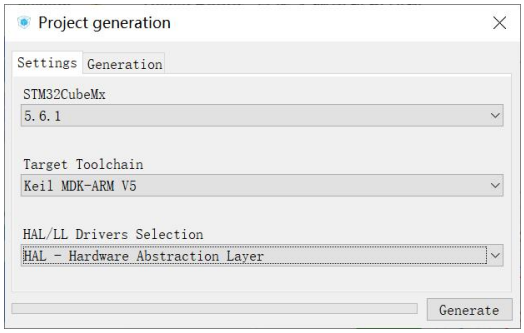


图 1.10 Workbench 生成工程

11) 生成完成，关闭窗口。

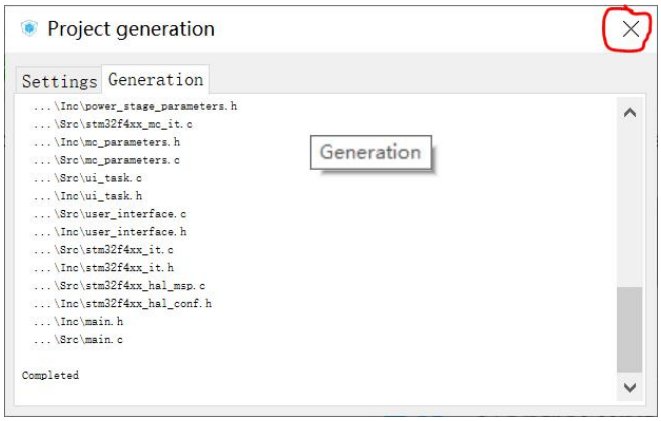


图 1.11 生成成功

12) 这时可以看到“My_HAL_MOTOR-654. FOC_v5.2.0_42PMSM_Ecoder”文件夹下面自动生成了 Cubemx 工程和 Keil 工程。

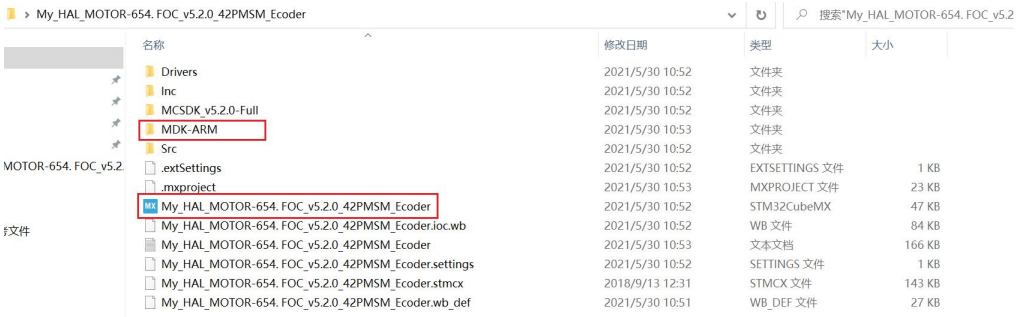


图 1.12 生成 Cubemx 与 Keil 工程

13) 打开 stm32cubemx 工程，可以看到模板工程已经配置好了基本的控制接口和通信接口。

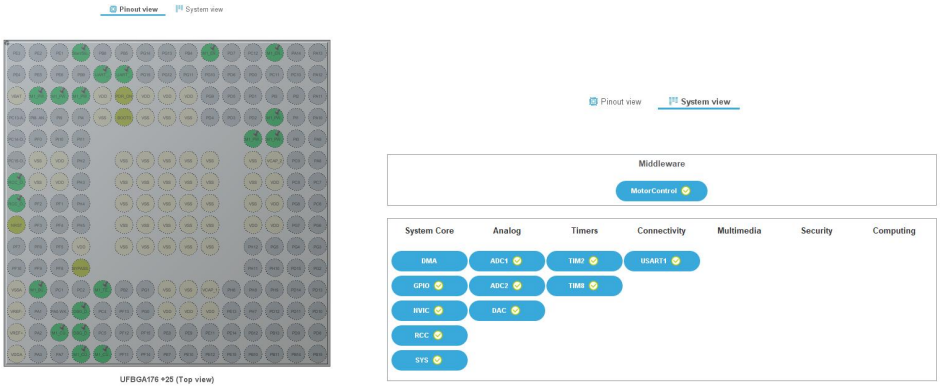


图 1.13 Cubemx 工程引脚配置

14) ST workbench 在生成 cubemx 工程的同时也生成了 keil 工程，打开 Keil 工程，将程序编译并下载到 STM32 开发板运行。

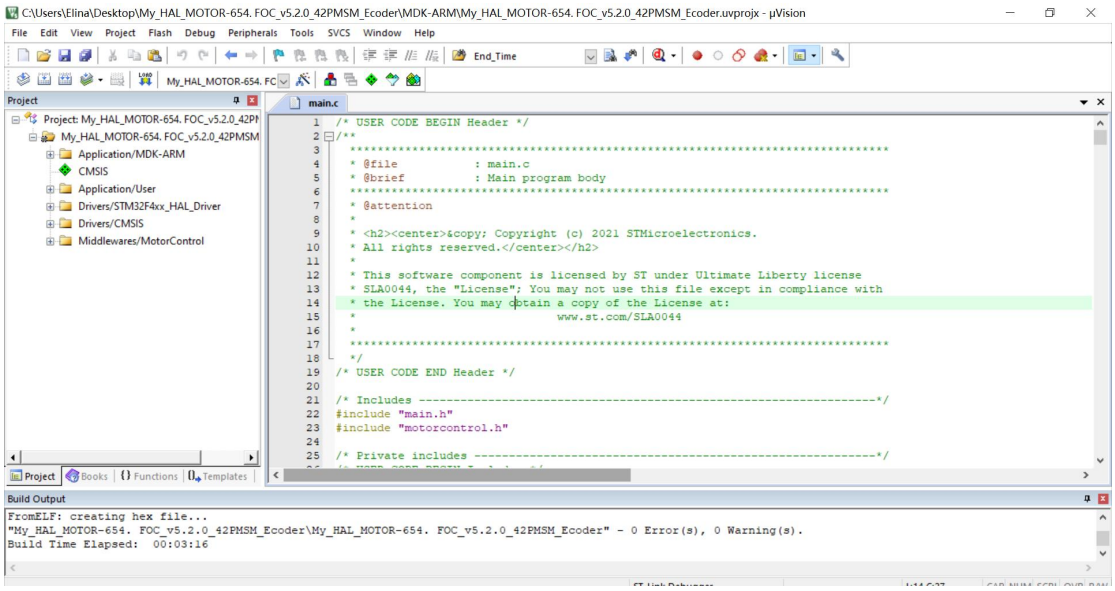


图 1.14 Keil 工程编译与下载

15) 点击 workbench 工具栏的监控器，打开监控 UI，然后配置好串口的波特率，点击连接，就可以在上位机控制电机转动，连接成功后会在工具栏下方显示固件版本号。

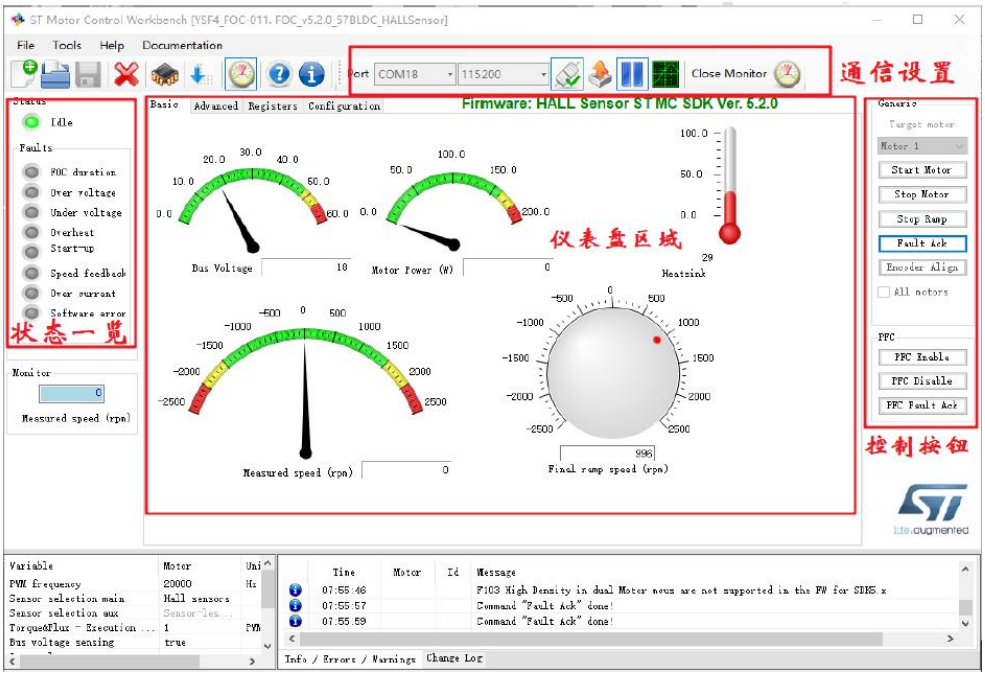


图 1.15 Workbench UI 与 STM32 连接

16) 所有的操作都只能在左边的 Status 栏亮绿灯 Idle （空闲状态）才可以操作。可以点击右边的 Start Motor 或者 Stop Motor 按钮来启动/停止电机转动。程序中默认速度值是 1000，可以通过下方旋钮改变速度。同样可以使用 KEY1 启动/停止电机。

17) 如果左边的 Status 出现下面的状况，很有可能是先给主控板上电，再给驱动板上电，这种情况导致开发板检测驱动板电压过低，是属于误报。可以通过点击右边的 Fault Ack 来消除报警。

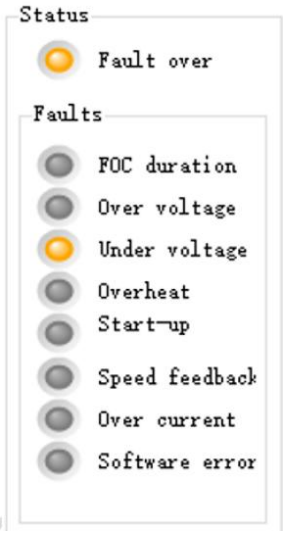


图 1.16 监控 UI Status 状态栏

18) 如果出现其他错误，则 Faults 会亮红灯，解除错误之后变黄灯，点击 Fault Ack 后灯灭，Status 会变成绿灯，回到 Idle 状态。

19) 如果不能连上上位机，可以试试按下开发板复位键，很有可能是刚下载完程序，还没有开始运行。

20) 在 Advanced 页，可以实时调整电机的 PID 参数。可以在运行的时候，更改运行模式，由 Speed 速度模式改为 Torque 转矩模式。同时可以修改 PID 参数。转矩环的 PID 参数通常是经过计算得到的，可以不修改。同学们可以尝试修改速度环 PID 参数，以达到更好的速度跟踪效果。

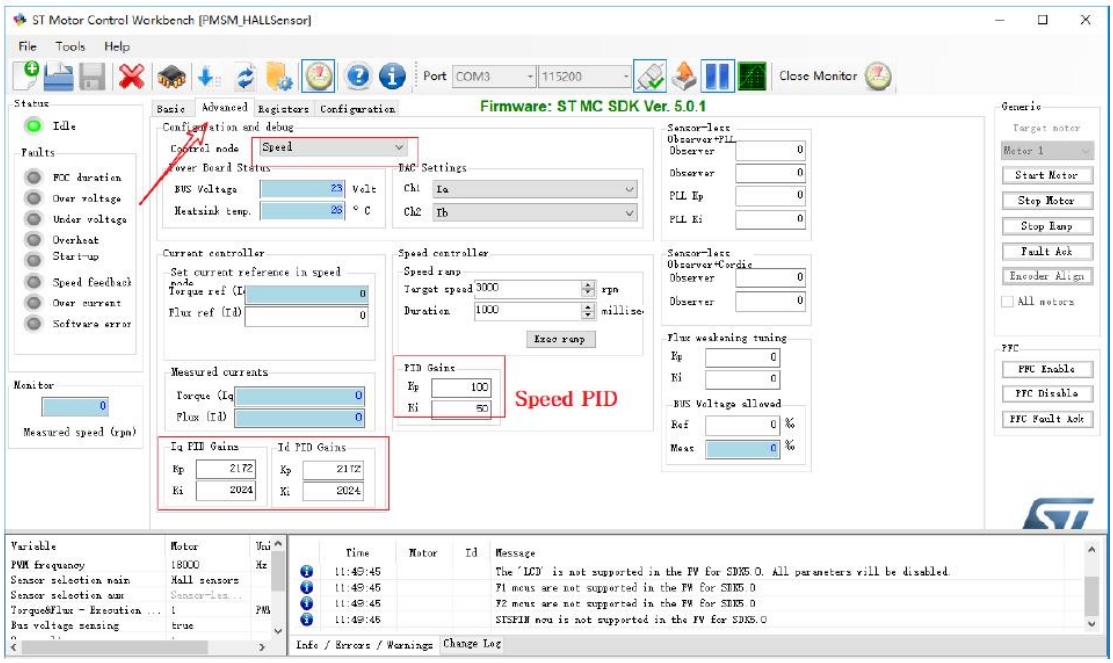


图 1.17 实时参数调试界面

21) 点击 图标，可以在表格观察电机转速。可以从下图看到，红线是目标速

度，白线是测量的速度值。可以实时观察电机转速，然后修改速度环 PID 参数，反复调整之后可以得到最佳的参数值。本实验是认知实验，对于参数调试不做具体要求，同学们熟悉操作，实测速度能基本跟随上设定转速即可。

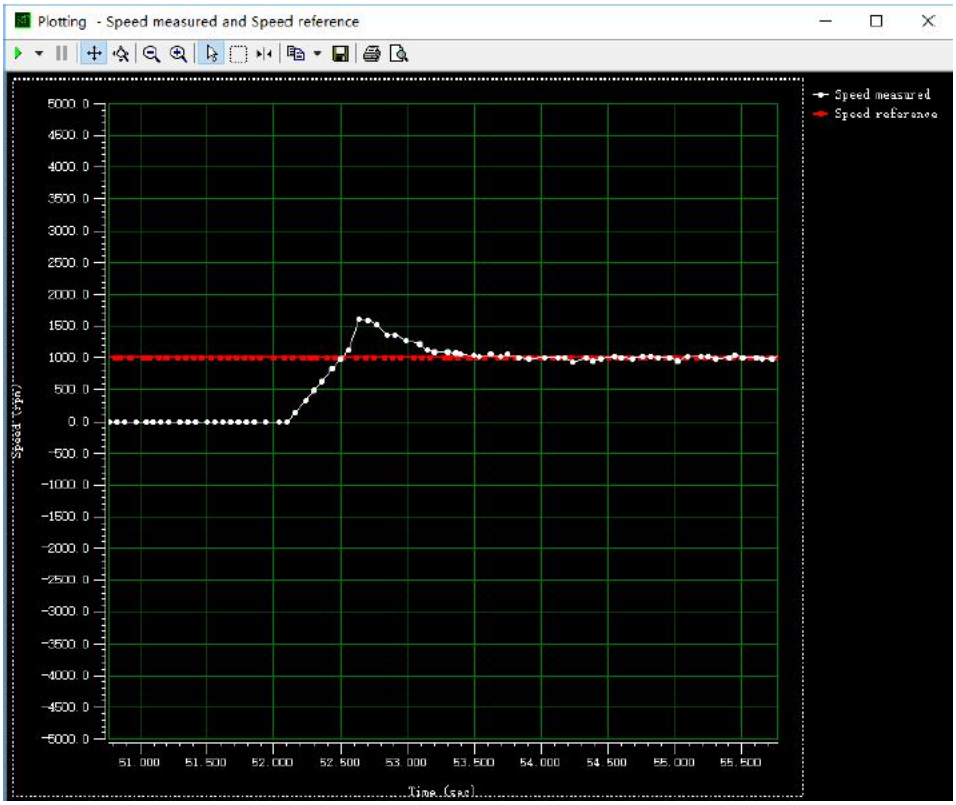


图 1.18 速度波形显示界面

1.4 任务验收

1.4.1 当堂验收

在 workbench 上位机波形显示界面，实测速度能跟随上设定转速。

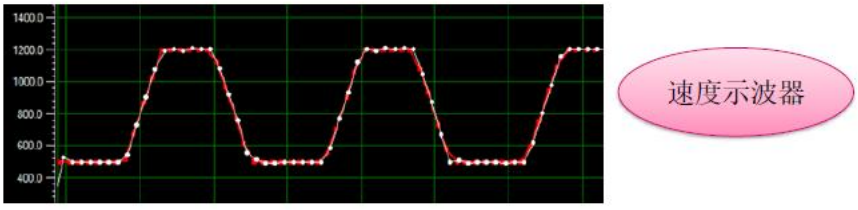


图 1.19 速度跟踪曲线

1.4.2 实验报告

记录 workbench 配置界面编码器对齐功能参数配置，以及波形显示界面设定速度与实测速度曲线，按要求写到实验报告中。

任务二 按键控制滑台回零

2.1 任务描述

通过按键 key_2 控制，启动滑台回零，零点为正、负限位的中点，可重复操作。

2.2 任务分析

在我们的综合实验创新平台——单轴丝杆滑台伺服控制系统上，有 5 个光电开关作为限位信号，和一个分辨率为 0.005mm 的光栅尺，作为位置传感器。

5 个光电开关分别连接到了 STM32 芯片的 5 个 GPIO 引脚，当未被遮挡时，对应引脚为高电平；当被滑台的拨片遮挡时，对应引脚为低电平。因此，可以通过检测这 5 个光电开关对应的 GPIO 引脚的高低电平来判断滑台的位置，进而执行相应的操作。

5 个光电开关中，最外侧的一对光电开关是安全限位开关，在设备内部被连接到了开关电源的控制端口。开关电源是用来给设备供电的，当光电开关被触发时，开关电源直接掉电，电机与驱动板停止运行，以保护设备。靠近安全限位开关的一对光电开关是正负限位开关，用来标记正负限位。中间的光电开关是 Home 限位，在本实验中暂时没有用到，如下图所示。

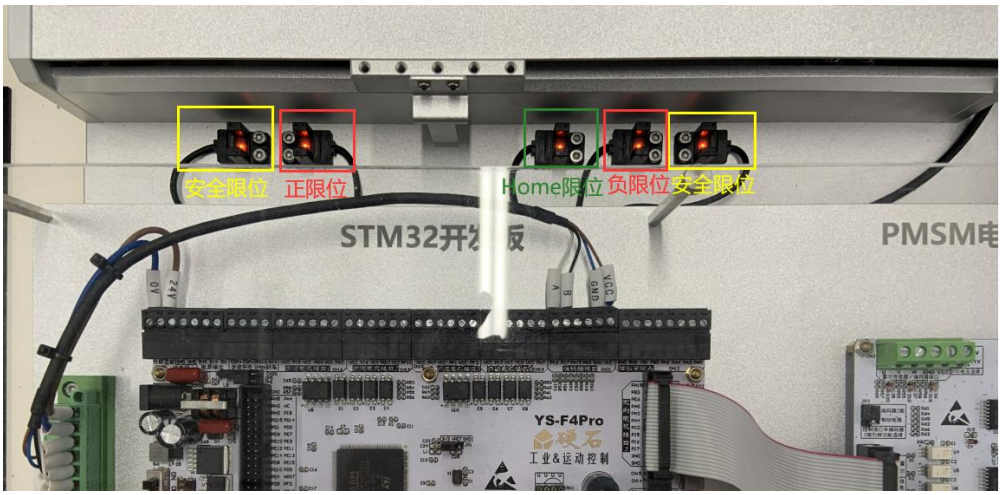


图 2.1 槽形光电开关

实验平台所用光栅尺实物图如下图所示，

图 2.2 光栅尺实物图

增量式编码器是将角位移转换成电信号，而光栅尺是将线位移转换成电信号。虽然两者的工作原理不同，但是对于 STM32 接口来说，两者是一样的，都是采用定时器的编码器测量接口，对 A、B 两相信号进行测量，当 A 相信号超前 B 相 90°时，电机正转；当 B 相信号超前 A 相 90°时，电机反转，如下图所示。

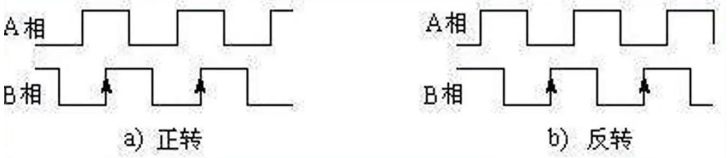


图 2.3 编码器 A、B 相信号

通过定时器的脉冲计数，可以得到光栅尺的线位移。当定时器的两个输入 TI1 和 TI2 被用来做光栅尺的 A、B 相信号的接口，并且计数器同时在 TI1 和 TI2 边沿计数时，光栅尺的线位移计算如下，

线位移（mm） = 定时器计数 x 0.005mm

综合实验平台的限位开关及光栅尺对应的 STM32 连接引脚如下表所示。

表 2.1 限位开关及光栅尺对应的 STM32 连接引脚

正限位	负限位	Home 限位	光栅尺 A 相	光栅尺 B 相
PG1	PG0	PG2	PC6	PC7

在本任务中，我们使用正、负限位光电开关和光栅尺这几个位置传感器就可以很容易地实现滑台回零的功能。

首先，我们可以使电机向正、负任一方向运动；当滑台的拨片触发到正限位时，对应引脚信号被置低，记录此时的光栅尺的位置为 limp；同理，当拨片触发到负限位开关时，负限位引脚被拉低，记录此时的光栅尺位置为 limn；则中点处的位置可以通过 limp 和 limn 计算得到；最后，控制滑台运动到中点位置处停下即可，如下图所示。

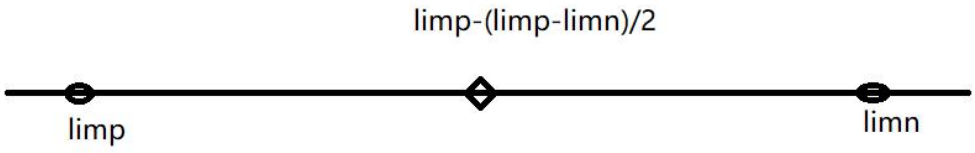


图 2.4 滑台回零功能示意图

至于 key_2 控制回零的启动，可以通过 key_2 按键的外部中断实现。

为了方便调试，我们通常还要用到串口打印一些变量的值进行分析。在任务一中我们知道，ST MC SDK 电机库已经占用了 USART1，我们要增加串口调试功能，可以选择 STM32 主控板上的 DB9 串口，其在主控板电路原理图中如下图所示。

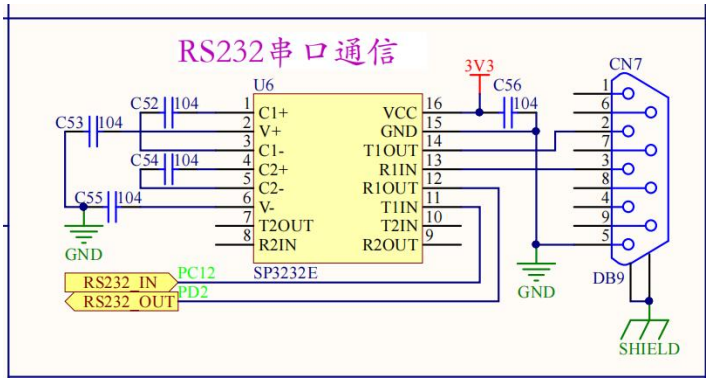


图 2.5 主控板串口电路原理图

本任务对零位位置精度不做要求，因此采用位置开环控制即可。可以使用 ST MC SDK 中的电机应用层 API 函数实现电机的启停与速度控制。

2.3 任务实现

根据任务分析，要想实现滑台归零需要完成如下内容

① 在 CubeMx 中添加配置限位开关、光栅尺、调试串口等外设接口，重新生成 Keil 工程。

② 确认光栅尺，限位开关等信号能在 keil 程序中正确读取。可以通过串口将光栅尺读数与限位开关状态打印出来。

③ 编程实现回零功能并调试。

下面将提供该任务实现的 CubeMx 配置部分的参考，并对编程实现部分做简单提示。

2.3.1 Cubemx 配置

1) 打开在任务一中，由 Workbench 生成的后缀为.ioc 的 Cubemx 工程，在自动生成的 Cubemx 工程基础上，添加下面的配置。
2) 将 timer3 作为光栅尺的 A、B 相信号的输入接口。将 PC6 与 PC7 分别设置为 timer3 的 CH1 通道和 CH2 通道。如下图所示，

图 2.6 光栅尺 A、B 相信号引脚配置

3) 将 Timer3 配置成编码器接口模式，

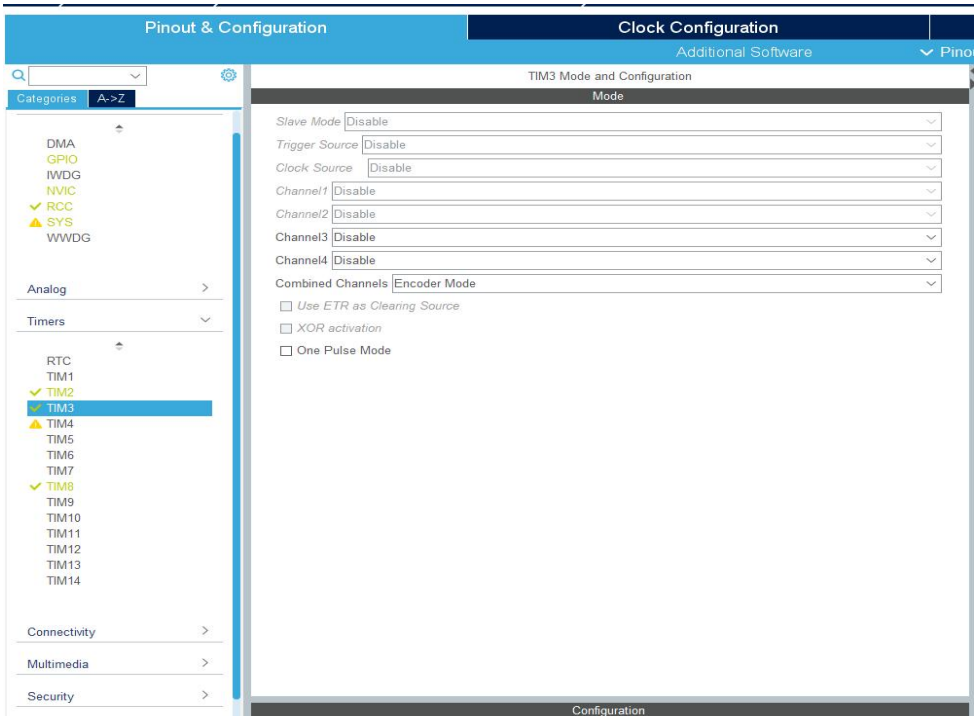


图 2.7 TIM3 配置 1

4) Timer3 的 parameter setting 配置如下，注意 Encoder Mode 的配置。

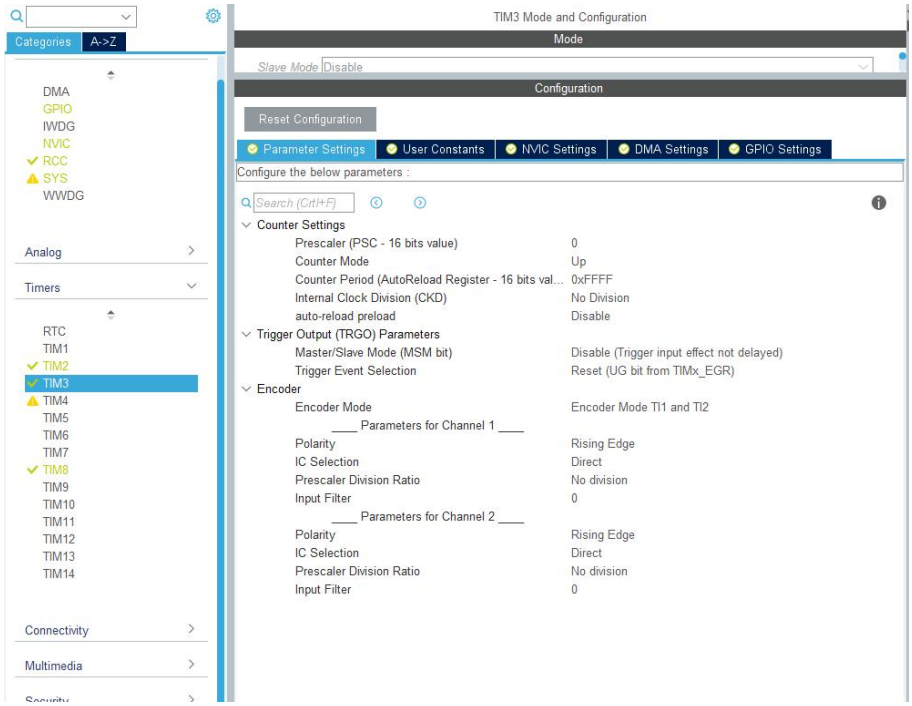


图 2.8 TIM3 配置 2

5) 使能 Timer3 的 NVIC 中断，如下图所示。

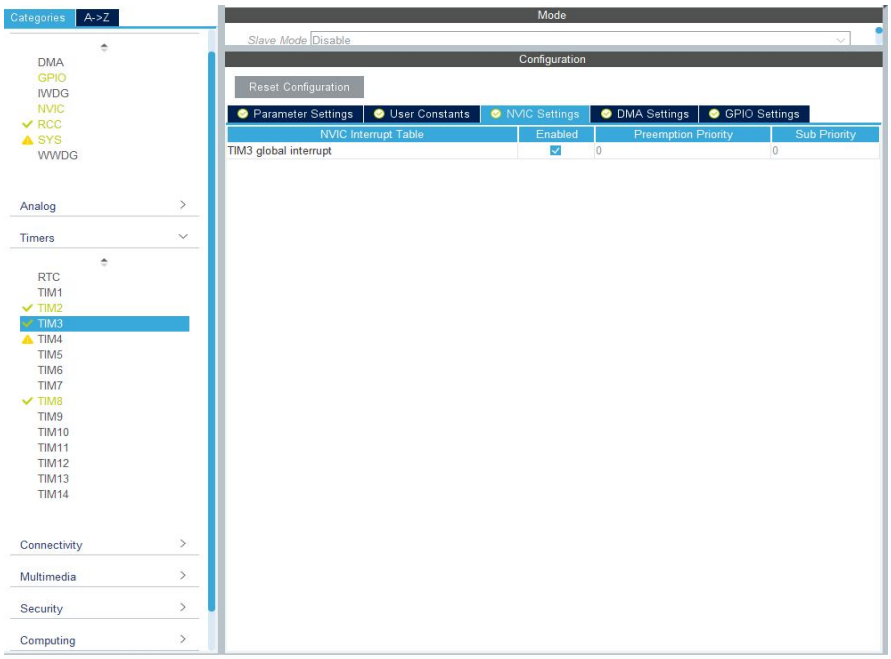


图 2.9 TIM3 配置 3

6) Timer3 的 GPIO 配置如下，

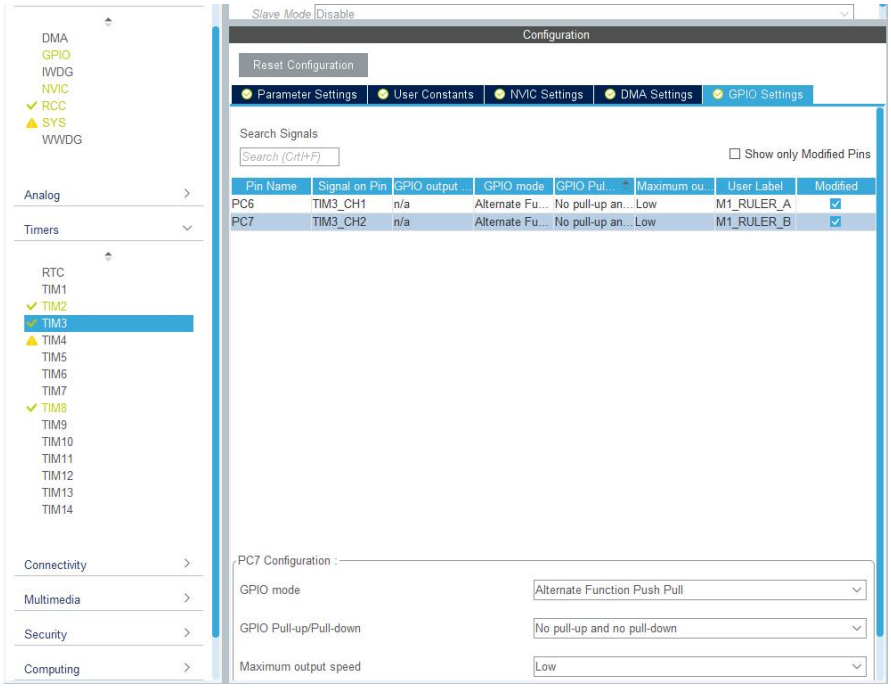


图 2.10 TIM3 配置 4

7) 下面将配置正负限位接口，PG1 为正限位开关接口，PG0 为负限位开关接口，引脚配置如下。

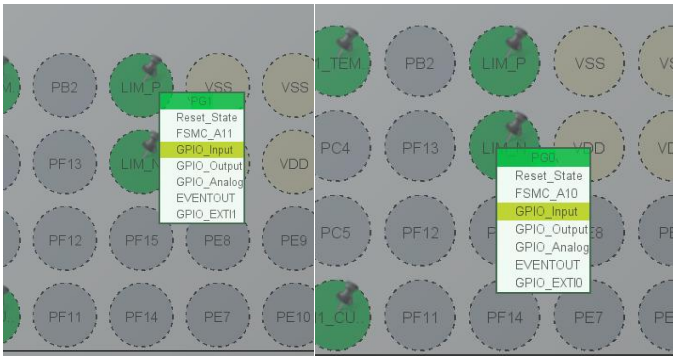


图 2.11 正负限位引脚配置

8) PG0 参数配置如下 ，

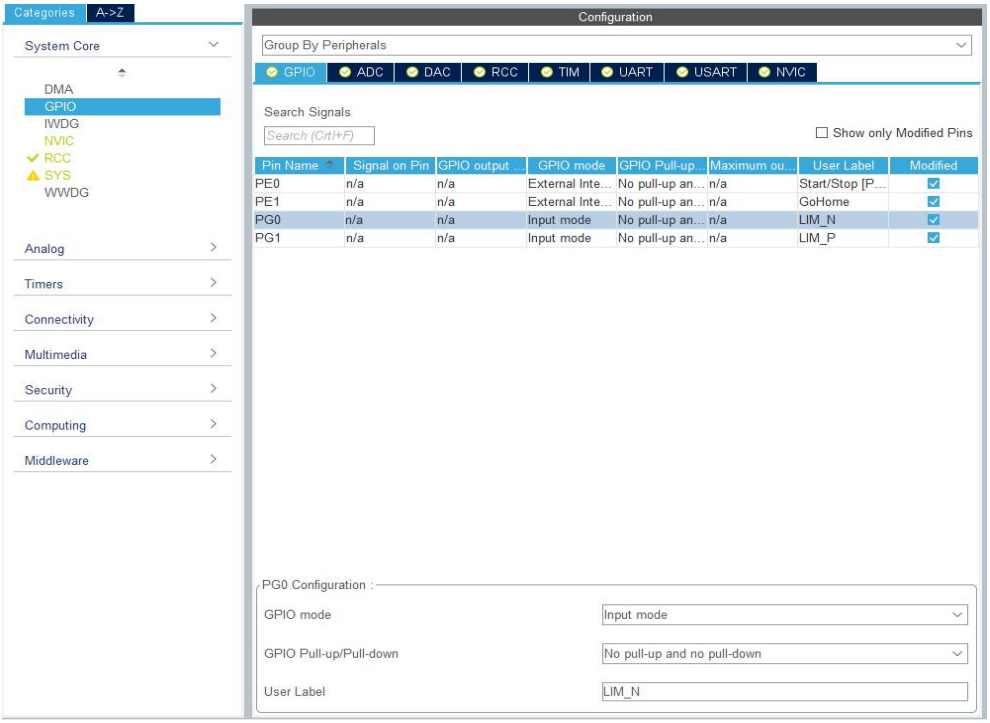


图 2.12 PG0 参数配置

9) PG1 参数配置如下，

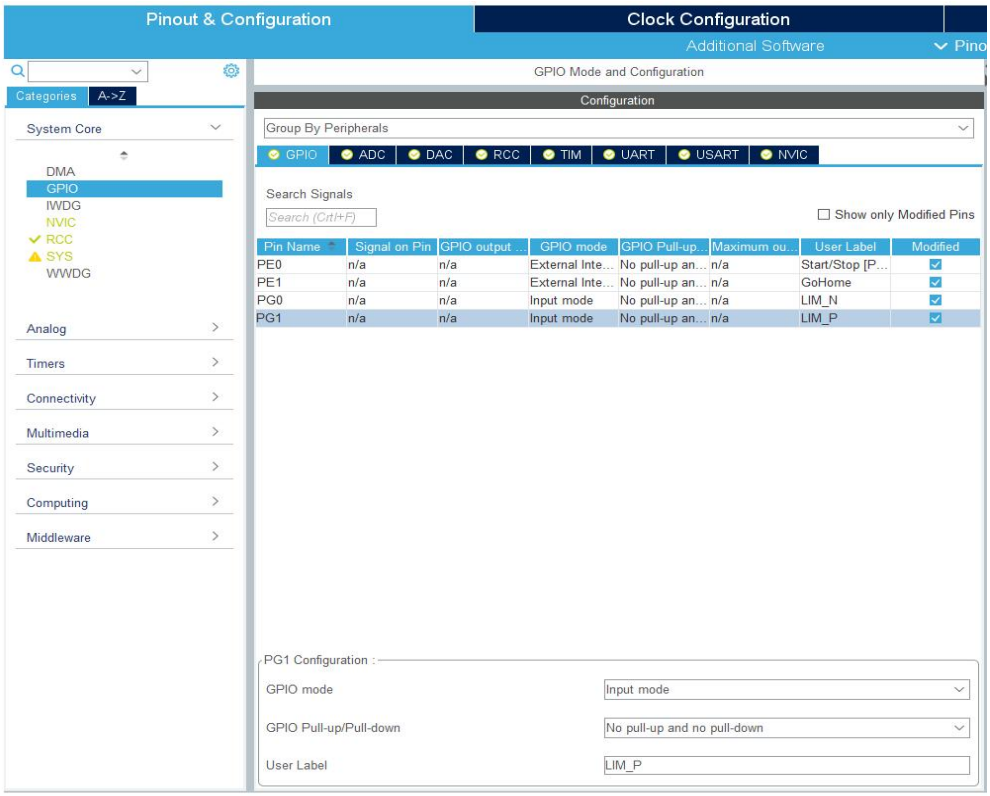


图 2.13 PG1 参数配置

10) 下面将配置控制回零启停的按键 Key2，Key2 对应的引脚为 PE1，将其设置为 GPIO_EXIT 模式。

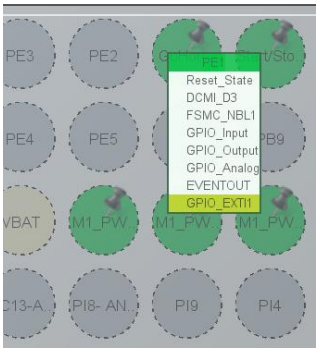


图 2.14 回零按键配置 1

11) PE1 的 GPIO 配置如下，

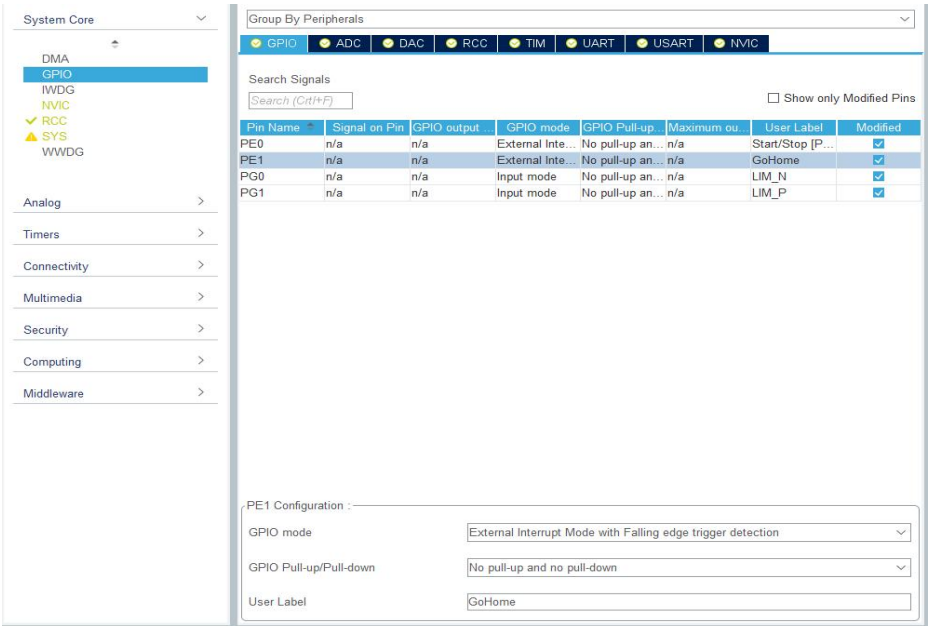


图 2.15 回零按键配置 2

12) 使能 PE1 的 NVIC 中断，如下图所示。

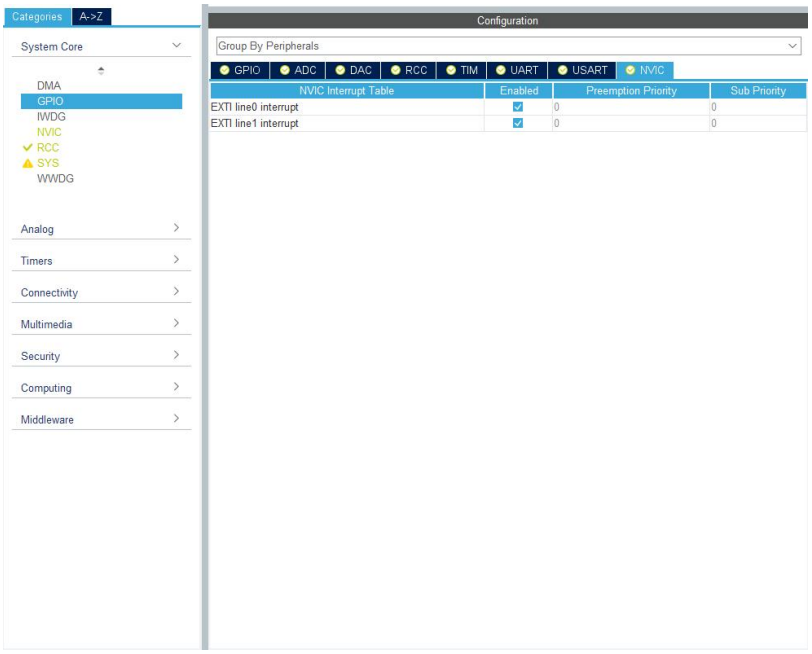


图 2.16 回零按键配置 3

13) 下面将配置 UART5 作为调试用串口，设置 UART5 为异步通信模式，如下图所示。

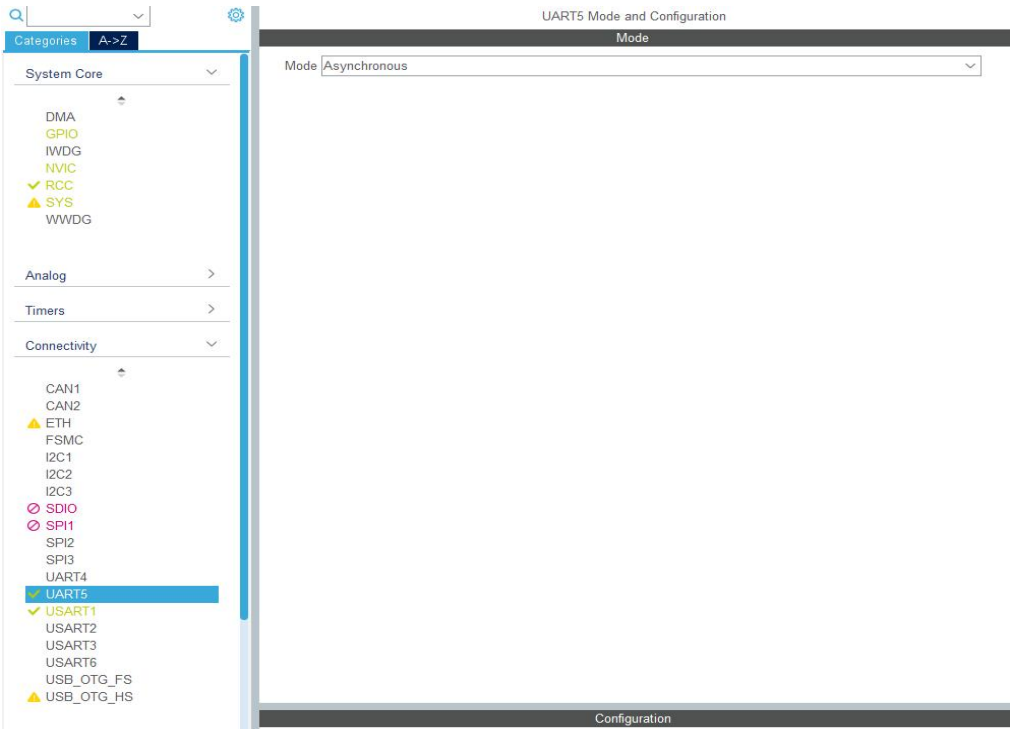


图 2.17 调试串口配置 1

14) UART5 的参数设置如下，

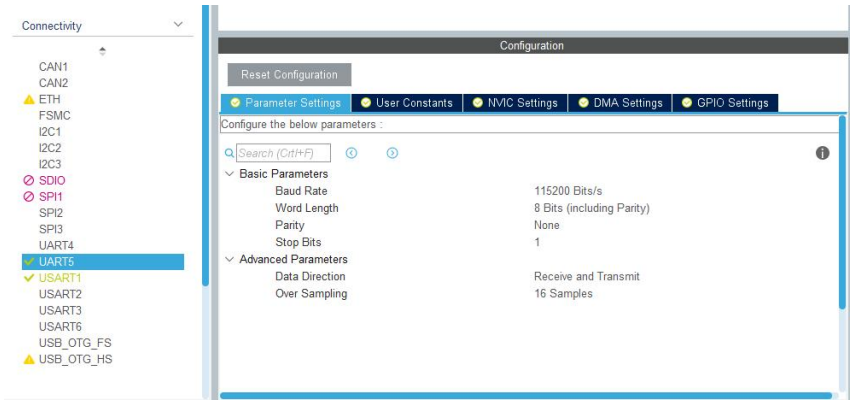


图 2.18 调试串口配置 2

15) 使能 UART5 的 NVIC 中断，配置如下。

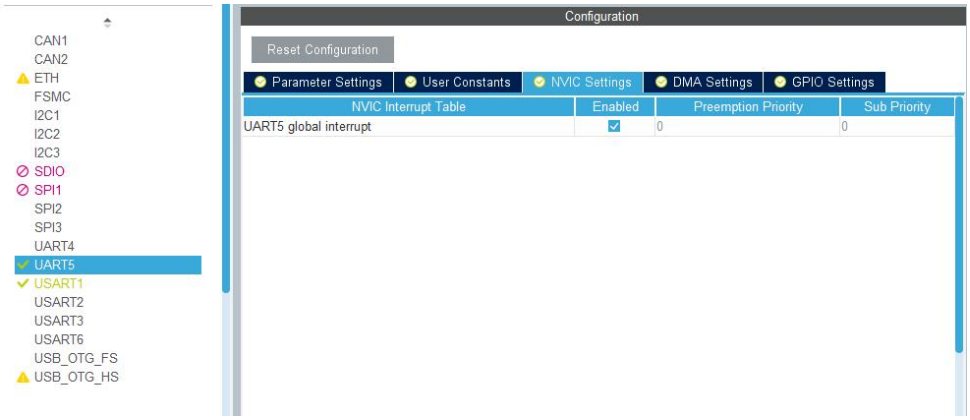


图 2.19 调试串口配置 3

16) UART5 的 GPIO 设置如下，

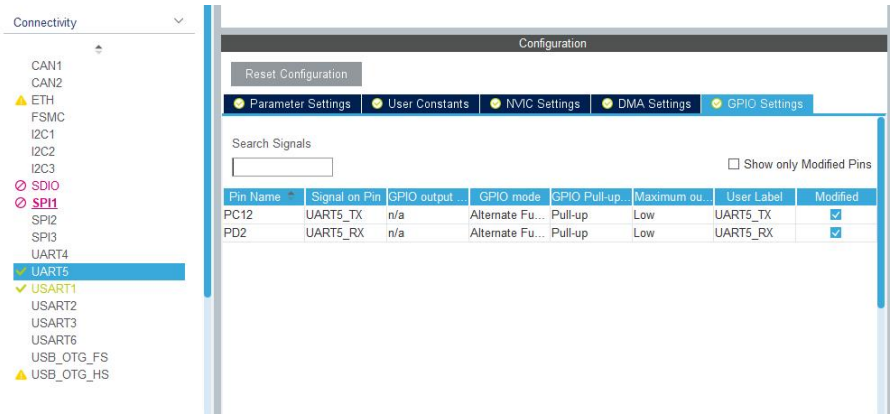


图 2.20 调试串口配置 4

17) 检查 NVIC 的配置如下，可以根据需要调整中断优先级。

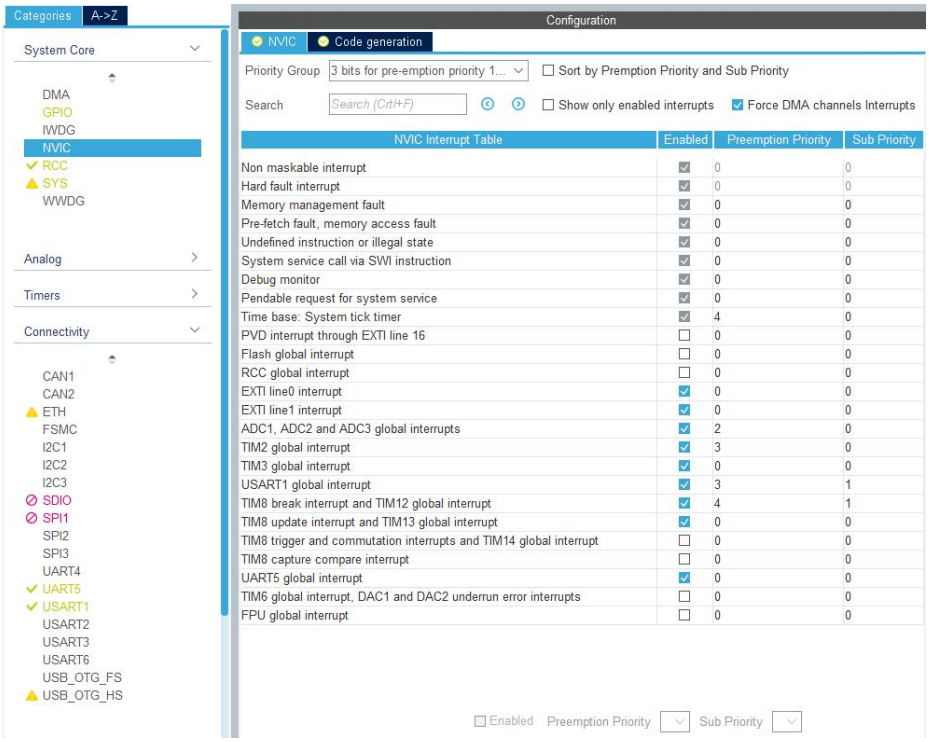


图 2.221 NVIC 配置

18) 至此，所有 Cubemx 配置完成，重新生成 keil 工程。

2.3.2 编程提示

1) 重定向 Printf，使其通过串口打印数据，以便后续通过串口对程序进行调试。重定向方法请参考 STM32 基础实验。
2) 在 main 函数中，初始化程序之后，while 循环之前，添加如下代码。其中，

HAL_TIM_Encoder_Start 用来启动 TIM3 光栅尺计数，MC_AlignEncoderMotor1 用来对 PMSM 内置编码器进行对齐，编码器对齐的作用在任务一中已说明。HAL_Delay(1000)的作用是等待编码器对齐完成。

```
/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_ADC1_Init();
MX_ADC2_Init();
MX_DAC_Init();
MX_TIM2_Init();
MX_TIM3_Init();
MX_TIM8_Init();
MX_USART1_UART_Init();
MX_MotorControl_Init();
MX_UART5_Init();

/* Initialize interrupts */
MX_NVIC_Init();
/* USER CODE BEGIN 2 */
HAL_TIM_Encoder_Start(&htim3, TIM_CHANNEL_ALL);
MC_AlignEncoderMotor1();
HAL_Delay(1000);
/* USER CODE END 2 */
```

图 2.22 初始化参考代码

3) 将 TIM3 的“counter period”设置为“0xFFFF”是为了使定时器能计更多的数，避免发生溢出。但是随着滑台运动距离的正向或负向增大，寄存器难免会有“上溢”或“下溢”的情况发生。在编程时要考虑到这两种情况，以免测量有误，可能用到的函数如下表所示。

表 2.2 TIM3 编码器函数

__HAL_TIM_IS_TIM_COUNTING_DOWN(&htim3)	检测 Counter 计数是否减少，是返回 1，否返回 0
__HAL_TIM_GET_COUNTER(&htim3)	获取当前 Counter 计数

4) 本任务使用电机控制库应用层 API 函数即可实现，无需使用位置闭环控制。可能用到 API 函数如下，下面对这三个函数作简单说明。

函数名称	函数参量	函数返回	函数功能
MC_StartMotor1	void	bool	启动电机
MC_StopMotor1	void	bool	停止电机
MC_ProgramSpeedRampMotor1	speed,time	void	设定速度以及时间
MC_ProgramTorqueRampMotor1	torque,time	void	设定力矩以及时间
MC_SetCurrentReferenceMotor1	Iqref，Idref	void	设定Iq，Id参考
MC_GetCommandStateMotor1	void	MCI_CommandState_t	返回指令执行状态
MC_StopSpeedRampMotor1	void	bool	停止速度指令执行
MC_HasRampCompletedMotor1	void	bool	指令是否执行完成
MC_GetMecSpeedReferenceMotor1	void	int16	返回机械参考速度
MC_GetMecSpeedAverageMotor1	void	int16	返回平均机械速度
MC_GetLastRampFinalSpeedMotor1	void	int16	返回上次指令速度
MC_GetControlModeMotor1	void	STC_Modality_t	返回控制模式
MC_GetImposedDirectionMotor1	void	int16	返回电机转动方向
MC_GetSpeedSensorReliabilityMotor1	void	bool	返回当前速度传感器是否可信
MC_GetPhaseCurrentAmplitudeMotor1	void	Is	返回电流值

图 2.23 任务用到的电机库 API 函数

① **MC_StartMotor1()**

启动电机转动，进入启动状态，如果电机当前状态是空闲状态，则将会立即执行启动电机，否则该指令将被丢弃。应用的时候可检查返回值以确认是否执行电机启动的命令。在调用该函数之前，需要先调用以下函数中的任意一个函数，

否则将会导致不可预测的行为：

- MC_ProgramSpeedRampMotor1()
- MC_ProgramTorqueRampMotor1()
- MC_SetCurrentReferenceMotor1()

MC_StartMotor1 仅仅是触发状态机从“IDLE”状态进入“IDLE-START”，如果成功执行命令，则该函数返回 True，否则返回 False。要想知道电机是否正常运行，应检查状态机当前是否处于“RUN”的状态。在 workbench 上可以找到与该函数相同功能的按钮：

图 2.24 Workbench 启停控制按钮

② **MC_StopMotor1()**

停止电机转动，进入停止状态。如果 Motor 1 的状态机正处于“RUN”或者“START”状态，立即执行停止电机命令，否则该命令被丢弃。应用的时候可以检查返回值以确认是否执行该命令。MC_StopMotor1()仅仅是触发状态机从运行状态进入“ANY_STOP”状态。要检查电机是否实际停止了，应该检查状态机的状态是否为“IDLE”状态，在 workbench 上可以找到与该函数相同功能的按钮。

③ **MC_ProgramSpeedRampMotor1()**

该函数用于设置线性速度目标值，该函数有两个参数，一个是最终目标速度（hFinalSpeed），一个是持续时间（hDurationms），最终目标速度值的单位按照注释是[0.1 Hz]，转子[0.1 Hz]也就是[0.1r/s]。当电机速度是 996RPM 的时候，该变量值就是 166，996[r/m] = 16.6[Hz]*60[s]。hDurationms 的单位则是 ms。运行该函数之后，电机将运行在 Speed 模式。

调用该函数的时候并不一定会立即执行，而是进入缓存队列中等待，只有在状态机正处于“START_RUN”或“RUN”的时候才会立即执行。任何时刻都只能有一个指令在缓存队列中，如果状态机当前正处于其他状态，该指令还没有被执行，在等待过程中有新的指令，则该指令将会被新的指令替换掉。要想知道当前指令是否被执行，可以调用 MC_GetCommandStateMotor1()函数获取缓存中最新的指令的执行状态。

该函数使得电机速度的目标值在运行过程中可以线性变化，逐步递增。在程序中调用该函数，给与参数值分别为 250，1000，参数设置速度是 250，那么目标速度就是 250 [0.1Hz] * 60[s] = 1500 [r/m] ，如果数值是负数，则代表电机反向转动。执行的效果如下图所示：

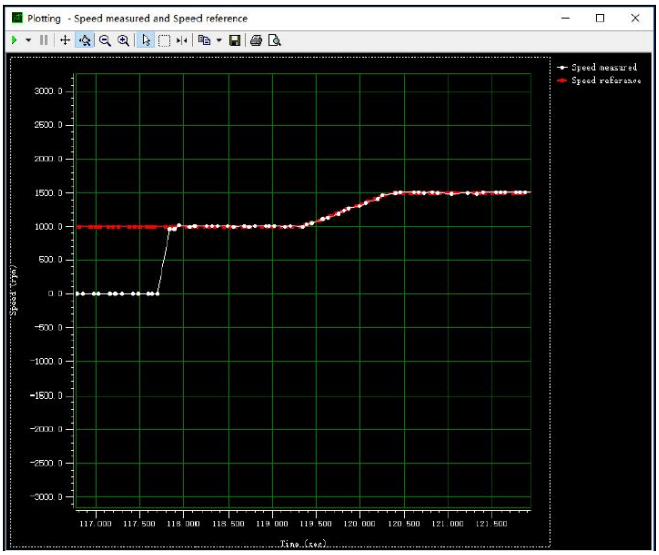


图 2.25 速度值线性变化

红色线条是目标值，白色线条是实际的速度值，一开始的时候电机是停止的，启动之后，电机速度就几乎与目标速度一致，在这个时候状态机状态是“RUN”，电机速度是 996RPM。当执行了该函数之后，则电机速度目标值呈现线性递增变化，逐步增加到所设置的最终目标速度（1500RPM），实际的速度值则会跟随目标值变化，速度爬坡所用的时间是 1000[ms]。当爬坡时间设置为 0 的时候，目标速度值将会直接变换成 1500RPM，中间不会有爬坡的效果。在 Workbench 上也有这样一个功能，见下图：

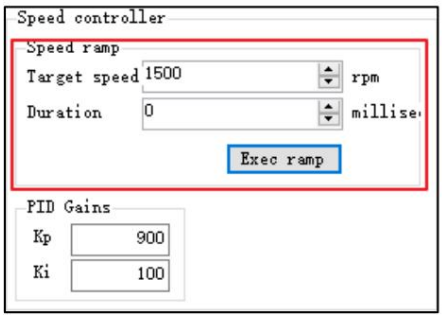


图 2.26 速度目标值线性控制

2.4 任务验收

2.4.1 当堂验收

1) 按下 key_2 按键，滑台启动回零，最终停在正、负限位中点处，可重复操作。
2) 将光栅尺位置信号通过串口打印，显示在串口调试助手上，检验为正确。

2.4.2 实验报告

1) 按照报告模板要求粘贴代码并写好注释。
2) 画出滑台回零程序流程图。

任务三 控制系统设计与实现

3.1 任务描述

实现单轴丝杆滑台的伺服控制系统设计，使得滑台的位置可以跟随期望指令运动。具体需要满足以下要求：

1) 阶跃响应要求：对于 2cm 的阶跃响应，要求 95%的上升时间不超过 0.1s；超调量不大于 10%; 稳态误差小于 1mm;

2) 跟踪带宽要求：对于幅值为 2cm 的正弦信号，以-3dB 为截止带宽标准的闭环带宽不小于 1hz;

3.2 控制系统介绍

3.2.1 机械组成

被控对象的机械组成如下图所示：

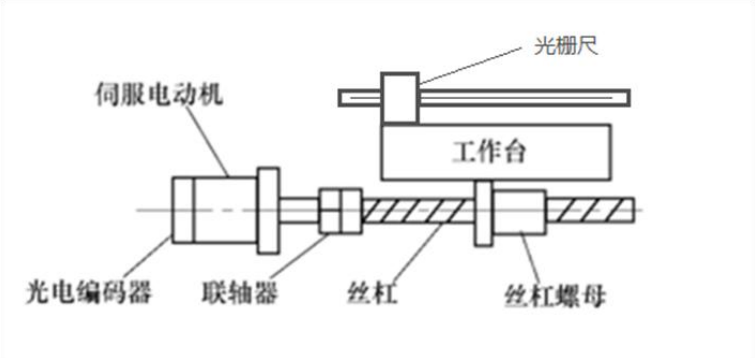


图 3.1 被控对象组成结构

被控对象为一个通过 PMSM 电机驱动的单轴的作业平台。电机转动带动丝杆转动，进而带动工作平台平动。光电编码器可以测量电机的转动过程，光栅尺可以测量工作台的平动过程。

滚珠丝杠（已基本取代梯形丝杆，俗称丝杆）是用来将旋转运动转化为直线运动或将直线运动转化为旋转运动的执行元件，并具有传动效率高，定位准确等优点。当滚珠丝杆作为主动体时，螺母就会随丝杆的转动角度按照对应规格的导程转化成直线运动，被动工件可以通过螺母座和螺母连接，从而实现对应的直线运动。

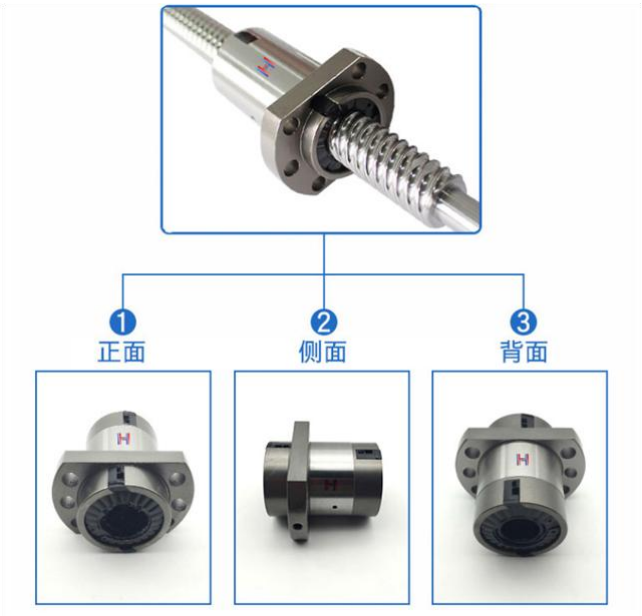


图 3.2 丝杆螺母副实物图

导程是滚珠丝杆有一个很重要的规格参数。导程描述的是螺纹上任意一点沿同一条螺旋线转一周所移动的轴向距离。

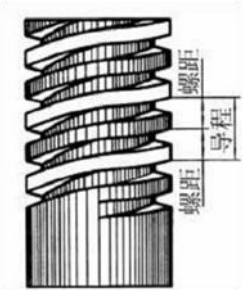
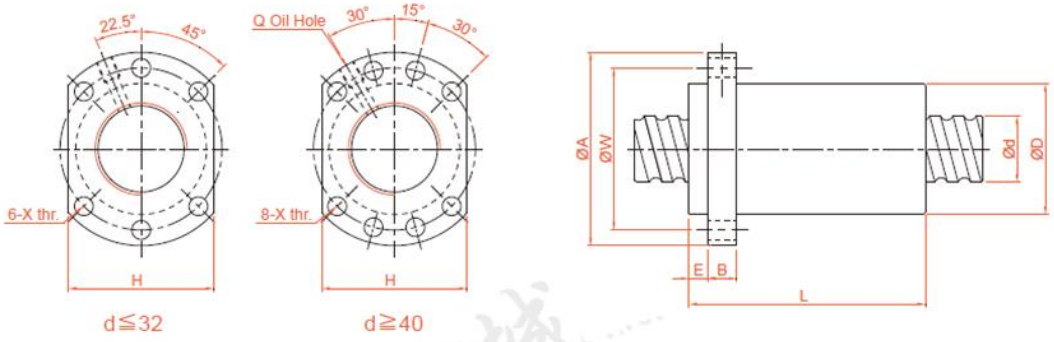


图 3.3 丝杆导程

本系统使用到的丝杆的型号为 SFA1610-2.8， 其对应的导程为 10mm，更多信息参数见下表：

SFA 轉造級系列規格尺寸表



單位：mm

型號	軸徑 d	導程 l	珠徑 Da	螺帽尺寸										滾珠螺帽額定負荷		剛性 kgf/μm
				D	A	E	B	L	W	H	X	Q	n	Ca (kgf)	Coa (kgf)	
SFA1205-2.8	12	5	2.5	24	40	5	10	30	32	30	4.5		2.8×1	661	1316	19
SFA1210-2.8		10	2.5	24	40	5	10	42	32	30	4.5		2.8×1	642	1287	19
SFA1605-3.8	15	5	2.778	28	48	5	10	31	38	40	5.5	M6	3.8×1	1112	2507	30
SFA1610-2.8		10	2.778	28	48	5	10	42	38	40	5.5	M6	2.8×1	839	1821	23
SFA1616-1.8		16	2.778	28	48	5	10	43	38	40	5.5	M6	1.8×1	552	1137	14
SFA1616-2.8		16	2.778	28	48	5	10	59	38	40	5.5	M6	2.8×1	808	1769	22
SFA1620-1.8		20	2.778	28	48	5	10	50	38	40	5.5	M6	1.8×1	554	1170	14
SFA1630-1.8		30	2.778	28	48	7	10	70	38	40	5.5	M6	1.8×1	534	1195	14
SFA2005-3.8	20	5	3.175	36	58	7	10	33	47	44	6.6	M6	3.8×1	1484	3681	37
SFA2010-3.8		10	3.175	36	58	7	10	52	47	44	6.6	M6	3.8×1	1516	3833	40
SFA2020-1.8		20	3.175	36	58	7	10	52	47	44	6.6	M6	1.8×1	764	1758	19
SFA2020-2.8		20	3.175	36	58	7	10	72	47	44	6.6	M6	2.8×1	1118	2734	29

图 3.4 丝杆螺母副参数表

有了导程h这个参数，就可以获得一些转换关系。
丝杆转动角度θ和滑台平动位移x的转换关系为：

$$x = \frac{h}{2\pi}\theta$$

由能量守恒可以得到，滑台的惯量折算为转动惯量的转换关系为：

$$J = m(\frac{h}{2\pi})^2$$

本实验使用的滑台的质量约为：540g；

本实验使用的丝杆的质量约为：600g，直径为 15mm;

思考：请问丝杆和滑台加在一起的转动惯量是多少？

3.2.2 电机

这里我们用到的执行器是一个永磁同步电机。其性能参数如下：

图 3.5 PMSM 电机实物图

42永磁同步电机(PMSM)参数	
供电电压	24 V
额定功率	63W
额定力矩	0.2 N.M
峰值力矩	0.6 N.M
额定转速	3000 RPM
额定电枢电流	3.13 A
力矩系数	0.057 N.M/A
反电势系数	4.13 V/KRPM
磁极	8(4对极)
编码器	1000线
相电阻	0.89±10%Ω
相电感	0.62±20%mH

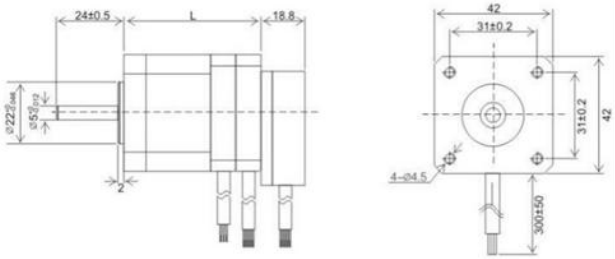


图 3.6 PMSM 电机参数与机械结构图

该电机的转动惯量为

$0.28kg*cm^2$

3.2.3 联轴器

联轴器是连接丝杆和伺服电机转动轴的机械部件，示意图如下。

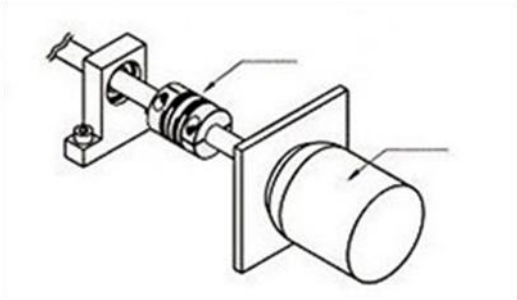


图 3.7 联轴器结构图

本实验用到的联轴器的型号为 YS3-26D，具体参数如下表：

型号	额定扭矩		最高转速 [r/min]	惯性力矩 [kg-m²]	静态扭矩刚性 [N.m/rad]	最大容许偏差			重量 [g]
	常用 [N.m]	最大 [N.m]				径向 [mm]	角向 [.]	轴向 [mm]	
YS3-19D	0.8	1.6	10000	0.58×10^{-6}	580	0.02	1	±0.10	11
YS3-26D	1.5	3	10000	2.26×10^{-6}	950	0.02	1	±0.15	25
YS3-32D	3.9	7.8	10000	7.20×10^{-6}	1800	0.02	1	±0.20	41
YS3-34D	4	8	10000	8.02×10^{-6}	2000	0.02	1	±0.20	16
YS3-39D	6	12	10000	1.80×10^{-5}	4500	0.02	1	±0.25	82
YS3-44D	10	20	10000	2.85×10^{-5}	5200	0.02	1	±0.30	100
YS3-56D	25	50	10000	8.96×10^{-5}	11000	0.02	1	±0.40	222
YS3-68D	60	120	10000	2.59×10^{-4}	19000	0.02	1	±0.45	389
YS3-82D	100	200	10000	7.10×10^{-4}	22000	0.02	1	±0.55	727

图 3.8 联轴器型号参数

其扭矩刚性为：

$$950Nm/rad$$

3.2.4 控制系统软硬件组成

被控对象的硬件和软件组成的功能模块图如下所示，其中，软件功能在 STM32 开发板和 Matlab 分两处实现。在 STM32 上，基于 HAL 库与电机控制库，实现回零、系统辨识、滑台控制算法、测试等一些列的功能。在 PC 端，基于 Matlab 实现控制系统的参数辨识、控制设计与仿真以及曲线实时显示的功能。

如图所示，绿色的部分是已经提供的代码或软件工具箱，无需我们开发；灰色的部分是需要同学开发的部分。其中，部分功能如通信功能、传感器读取、按键控制、回零功能已经在之前的任务中完成。

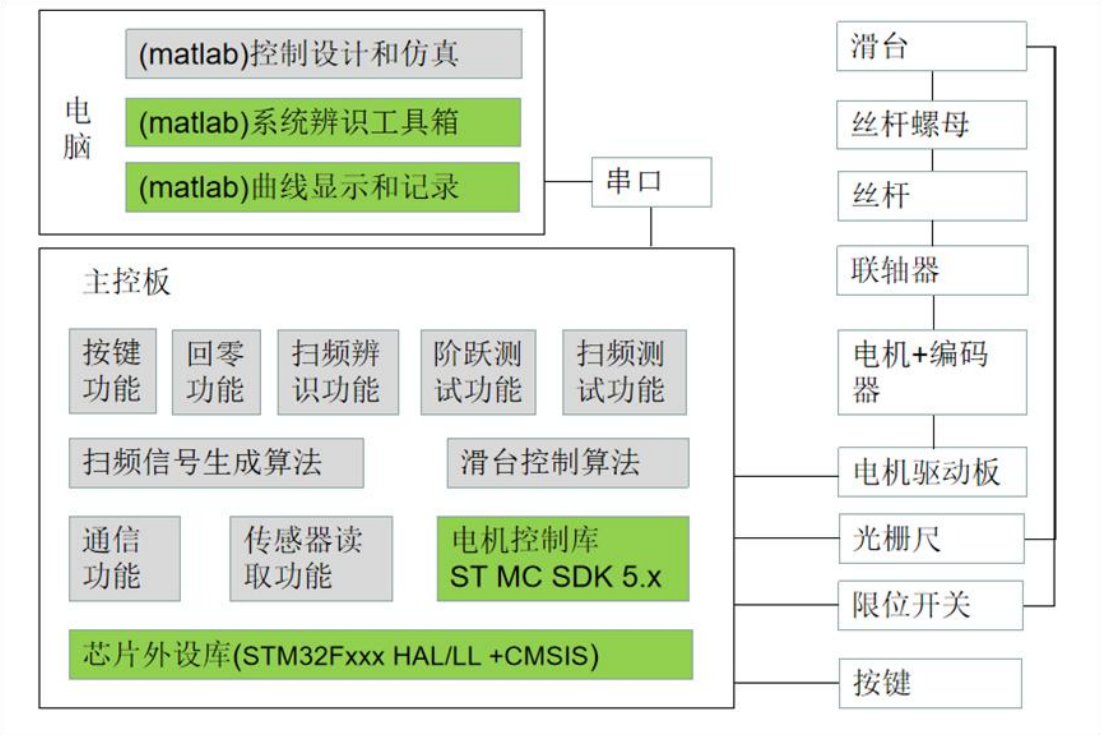


图 3.9 被控对象的硬件和软件组成的功能模块图

整个控制系统需要修改的软件部分主要在主控板上，根据上图可知，整个主控板内将会包含如下功能：

1) 芯片外设库：这是通过 Cubemx 生成的外部硬件设备的驱动代码。

2) 电机控制库：这是 Workbench 自动生成的电机控制库程序。有了这个 SDK, 在主控板中，就可以通过调用电机控制库 API 函数的方式，控制电机。

3) 传感器读取功能：这个功能，需要同学开发，读取光栅尺读取的函数，已经在任务二完成。

4) 通信功能：需要同学开发一些与上位机通信的功能，比如重定义 printf 函数，使得 debug 过程变得简单；比如开发主控板与上位机之间的通信功能，使得一些测试数据能够在 matlab 上现实出来。部分功能已经在任务二中实现。

5) 按键功能：需要同学定义不同的按键需要触发的功能。使得电机可以启动或停止某个功能，已经在任务二中完成。

6) 回零功能：需要同学开发按键后作业平台就能够回到中点的功能，这样同学就不用经常手动去推到作业平台。已经在任务二中完成。

7) 扫频测试功能：需要同学开发一些扫频函数，作为控制系统的输入信号。这个功能可以用来做系统辨识和闭环带宽测试。

8) 作业台控制程序：这个程序，是需要同学开发的实现控制作业平台的位置伺服控制功能的程序。这样输入阶跃信号或扫频信号的时候，作业平台能够快速准确地跟踪上输入信号。

3.3 任务分析

根据任务描述以及对控制对象的认知，我们可以将该任务的实现分为如下几

个步骤。

表 3.1 任务步骤列表

序号	步骤	任务描述
1	控制系统建模	构建被控对象的传递函数模型（其中未知的变量可以用符号进行表示），对传递函数进行精简；
2	控制系统辨识	在控制板中开发扫频辨识功能，获得被控对象的开环扫频测试数据。结合步骤 1 给出的精简的被控对象的传递函数模型，使用 MATLAB 工具箱，完成系统辨识工作，获得辨识的被控对象的模型；注意：辨识的被控对象的模型和实际的被控对象的模型之间会存在偏差；
3	控制器设计	基于步骤 2 得到的辨识的被控对象模型，设计控制器；
4	控制器仿真验证	集合步骤 2 得到的辨识的被控对象模型，和步骤 3 中得到控制器。在 SIMULINK 中搭建控制系统仿真模型，对控制系统的性能进行仿真验证；
5	控制程序开发	把步骤 4 中验证好的控制器，转变为离散化的控制器，然后开发 STM32 主控板上的控制程序；
6	控制系统调试	对整个控制系统进行阶跃响应调试和扫频跟随调试；对控制器进行优化，直到控制效果达到控制任务的性能指标；

3.4 任务实现

3.4.1 控制系统建模

3.4.1.1 机械谐振模态分析

请参考理论课件《第四章 控制系统的设计约束（2）》中的 4.3.2 章节中关于结构模态的内容，以及上一个小结对控制系统的介绍，计算联轴器的谐振频率；与控制系统要求的带宽进行对比，确定机械谐振模态是否可以忽略；

3.4.1.2 被控对象的系统建模

请参考理论课件《第二章 控制系统的设计流程》中关于数学建模（简化和处理）的内容，以及《第四章 控制系统的设计约束（2）》中的 4.3.4 中的内容，确定被控对象的系统模型；

注意：未知的参数可以用字符来标识，模型需要适当简化，为后续的系统辨识工作提供了模型的阶次信息。

3.4.2 控制系统辨识

参考理论课件的《第四章 控制系统的设计约束（2）》中的 4.3.4 中的关于系统辨识的内容。整个系统辨识的流程如下图所示，扫频信号发生器生成包含不同频率成分的激励信号，输入到被控系统，获得其输出。两者数据被保存到电脑上，然后使用 matlab 进行辨识。在辨识的过程中，依赖模型的阶次信息，这个在上一个步骤中通过系统建模工作已经得到了。

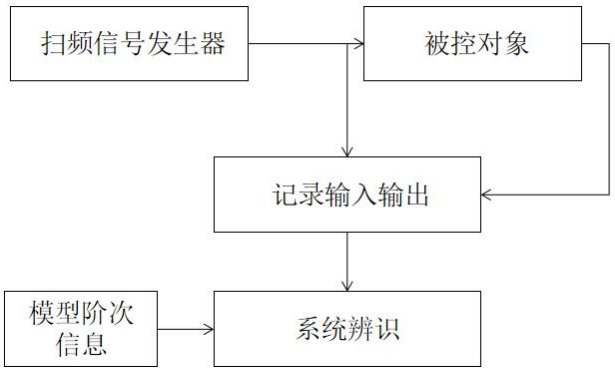


图 3.10 系统辨识流程图

3.4.2.1 在主控板上实现正弦扫频信号生成算法

编程实现扫频信号的生成，对应控制系统软硬件功能模块图如下图所示。

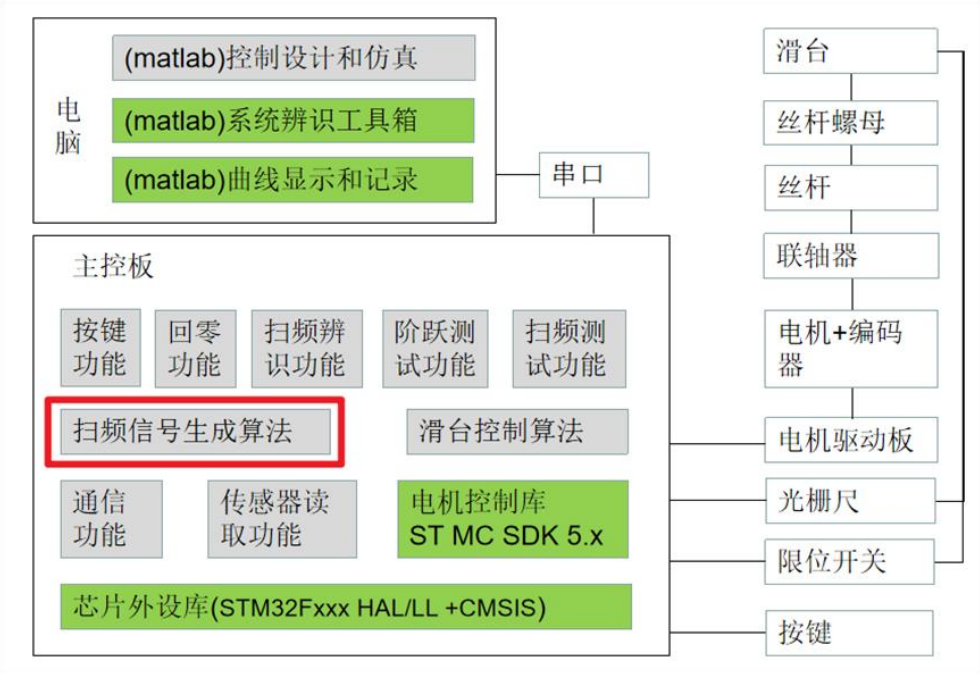


图 3.11 实现扫频信号生成算法

扫频信号是一种频率随着时间不断变化的信号，如下图所示：

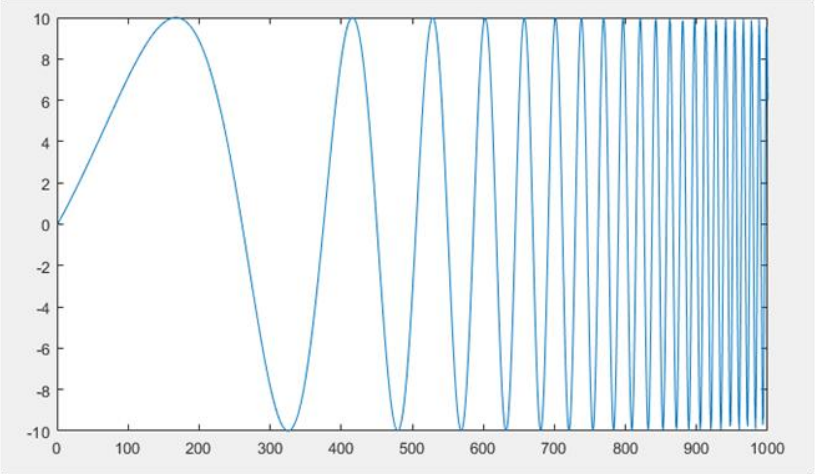


图 3.12 扫频信号波形图

假设信号的频率相对时间的函数$f(t)$是随着时间指数增长的，那么有：$f(t)=f_0k^t$，其中f_0是开始时刻t_0对应的频率；假设我们系统在t_1时刻，使得其频率达到f_1。那就可以计算出$k=e^{\frac{1}{t_1-t_0}ln(\frac{f_1}{f_0})}$；然后对$f(t)$进行积分就可以得到相位相对时间的函数：

$$\phi(t)=2\pi\int_0^t f(t)d\tau=2\pi\int_0^t f_0k^t d\tau=\frac{2\pi f_0}{\ln(k)}\left(k^t-1\right)$$

有了相位信息，就可以得到正弦型号的输出值为：

$$y(t)=A\sin\left(\phi(t)\right)=A\sin\left(\frac{2\pi f_0}{\ln(k)}\left(k^t-1\right)\right)$$

定义：

$$p=\frac{2\pi f_0}{ln(k)}$$

则有：

$$y(t)=A*sin(p(k^t-1))$$

上面就是一个频率随着时间指数增加的扫频信号的数学原理，接下来，我们需要通过编程的方式实现这个功能。

我们首先定义一个结构体来存储与该算法相关的变量：

```
typedef struct
{
    unsigned int t_0;     /* t0 信号发送器开始工作的时刻，  单位 s   */
    unsigned int t_01;    /*  从 t0 到 t1 的时间间隔，  单位 s */
```

```
    float f0;      /* 时刻 t0 对应的频率，    单位 hz */
    float f1;      /* 时刻 t1 对应的频率，    单位 hz */
    float k;       /* 指数函数的底数  */
    float p;       /* 系数  p */
    float A;       /* 扫频信号的幅值  */

}my_sweep_t;
```

然后定义一个初始化函数，用于设置扫频信号的一些配置信息

```
/*
函数：init_my_sweep
功能：初始化一个频率随着时间指数增加的正弦扫频信号的结构体
输入：unsigned int t_0     t0 信号发送器开始工作的时刻，单位 ms
输入：unsigned int t_01    从 t0 到 t1 的时间间隔，单位 ms
输入：float f0      时刻 t0 对应的频率，   单位 hz
输入：float f1      时刻 t1 对应的频率，   单位 hz
输入：float A       扫频信号的幅值
输出：int           0 = 成功，  其他表示异常
*/
int init_my_sweep(my_sweep_t *sweep, unsigned int t_0, unsigned int t_01, float f0, float f1, float A)
{
    if ((t_01 == 0)  ||  (f0 <=0.0f)  || (f1 <= 0.0f) || (f0 == f1) || (A == 0) || (!sweep))
    {
        return -1;  /*非法入参*/
    }

    sweep->t_0 = t_0;
    sweep->t_01 = t_01;
    sweep->f0 = f0;
    sweep->f1 = f1;
    sweep->A = A;

    /* start add code here */
    // sweep->k = /*计算指数函数的底数 k，注意时间的单位要转换为秒*/
    // sweep->p = /*计算系数 p，注意单位转换*/
    /* end add code here */

    return 0;

}
```

然后定义一个获取正弦扫频信号的函数

```
/*
```

```
函数：run_my_sweep
功能：根据当前时间输出频率随着时间指数增加的正弦扫频信号
输入：sweep                  信号发生器结构体指针
输入：unsigned int t_now   当前时间   单位 ms
输出：float  扫频信号

*/
float run_my_sweep(my_sweep_t *sweep, unsigned int t_now)
{
    float t = 0.0f;   //相对时间  t
    float y = 0.0f;   //扫频信号

  if (!sweep)
    {
        return 0.0f; /*非法入参*/
    }

    if (t_now < sweep->t_0)
    {
        return 0.0f; /*时间还未得到*/
    }

    t = (t_now - sweep->t_0) % sweep->t_01;   /*通过求余操作实现，周期性扫频的过程*/

    t = t * 0.001f;   /*将单位转换为  s*/


    /* start add your code here */

    //y =

    /* end add your code here */

    return   y;

}
```

3.4.2.2 实现主控板与 MATLAB 之间的通信功能

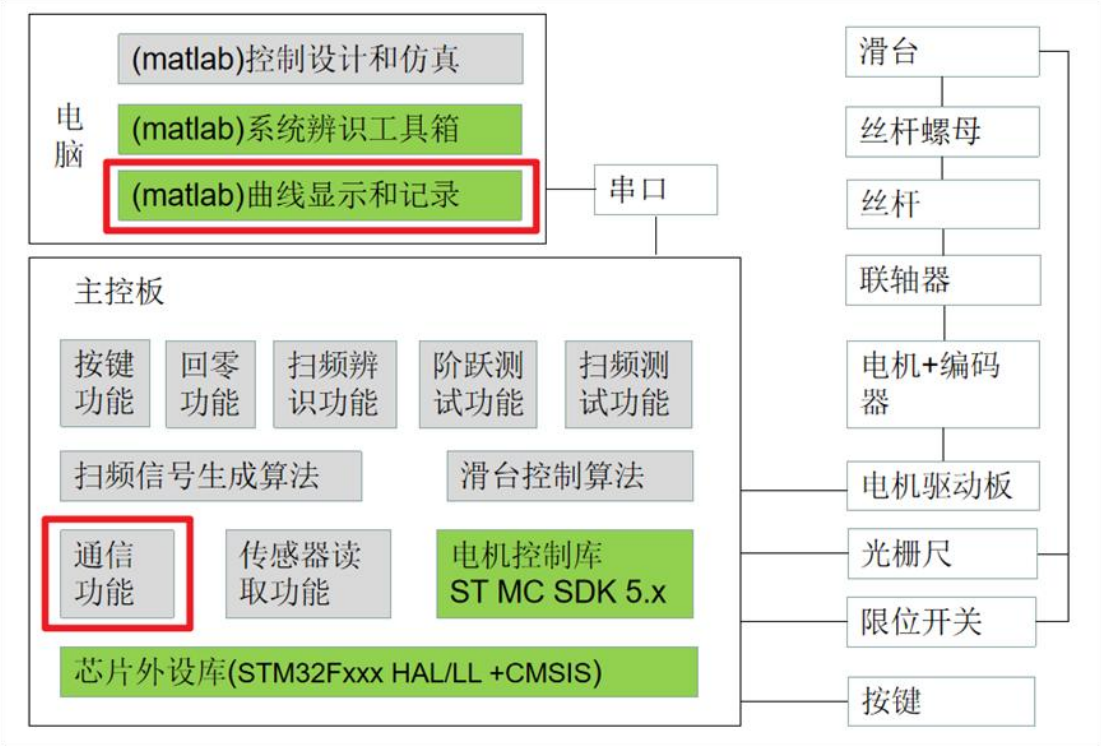


图 3.13 实现主控板与 Matlab 通信功能

为了使用 matlab 的系统辨识工具箱，进行系统辨识的工作，同学需要把系统辨识的扫频信号输入和被控对象的输出发送到 matlab 中，为此同学需要实现一个把数据发送到 matlab 的通信功能。通信功能的框架如下图所示：

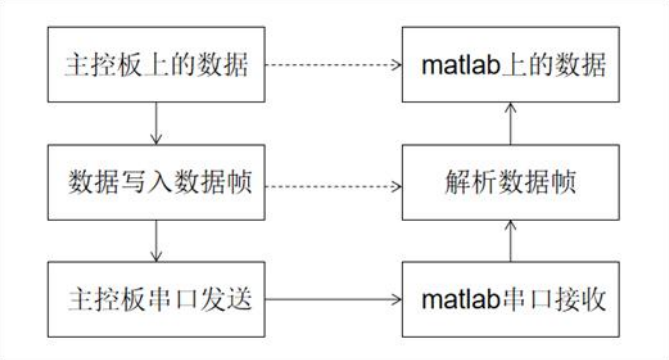


图 3.14 Matlab 与主控板通信功能框架

主控为了实现把数据发送到 matlab 中的目的。把主控板中的数据放入一个数据帧中，然后通过串口发送给电脑。电脑上的 matlab 通过串口接收到数据，然后安装数据帧的格式，把数据解析出来，就获取了主控板上的数据了。

一种主控板发送端的具体实现方式如下，同学如果有意愿也可以定义自己的实现方法：

```
typedef struct
{
    uint8_t start_flag;  /*帧的起始标志*/
```

```
    uint8_t frame_len;    /*帧的长度信息*/
    uint8_t header_check;  /*帧头的求和校验*/
    uint8_t data_buf[12];  /*数据长度，这里选择固定的数据长度*/
    uint8_t frame_check;      /*帧头+数据的求和校验,用于接收方校验数据的完好性*/

}frame_matlab_t;  /*数据帧结构体*/

/*  对  uint8_t  数值进行求和  ,获得这组数据一个 uint8_t 特征*/
uint8_t get_uint8_sum_check(uint8_t *data, int len)
{
    int i = 0;
    uint8_t sum = 0;
    for (i = 0; i < len; i++)
    {
        sum += data[i];
    }
    return sum;
}

/*
函数：send_data_2_matlab
功能：往 matlab 发送三个浮点数
输入：三个浮点数
输出：无
*/
void  send_data_2_matlab(float data1, float data2, float data3)
{
    frame_matlab_t  frame = {0};
    int i = 0;

    /*填充帧头*/
    frame.start_flag = 0xAA;
    frame.frame_len  = sizeof(frame);
    frame.header_check = get_uint8_sum_check((uint8_t *)&frame, 2);
    /*填充数据*/
    memcpy((uint8_t *)&frame.data_buf[0], (uint8_t *)&data1, 4);
    memcpy((uint8_t *)&frame.data_buf[4], (uint8_t *)&data2, 4);
    memcpy((uint8_t *)&frame.data_buf[8], (uint8_t *)&data3, 4);
    /*计算数据求和值,用于接收方校验数据的完好性*/
    frame.frame_check = get_uint8_sum_check((uint8_t *)&frame, frame.frame_len-1);

    /*通过  串口 5  发送到电脑  */
    HAL_UART_Transmit(&huart5, (uint8_t *)&frame,frame.frame_len,0xffff);
}
```

接收端的程序参考：MATLAB 上的接收程序见 matlab_ui/plot_mcsdk.m 该程序实现了打开电脑串口的，并调用 matlab_ui/callback_mcsdk.m 回调函数对接收到的串口数据进行接收，解析数据帧，存储数据，绘制数据实时曲线的功能。

plot_mcsdk.m 第 44 行的串口号，需要修改为主控板与电脑连接的串口编号。

```
43 -  try
44 -      s=serial('com4');
45 -  catch
46 -      error('cant serial');
47 -  end
48 -  set(s,'BaudRate', 115200,'DataBits',8,'StopBits',1,'Parity','none','FlowControl','none');
49 -  s.BytesAvailableFcnMode = 'byte';
50 -  s.BytesAvailableFcnCount = 160;
51 -  s.InputBufferSize = 151200;
52 -  s.BytesAvailableFcn = {@callback_mcsdk};
```

图 3.16 修改 Matlab 中串口配置

如果不确定这个串口编号，可以在设备管理器中查看：

图 3.17 查看设备管理器串口号

通过插拔串口线，可以看到对应的串口会发送变化。

到此主控板就可以通过 send_data_2_matlab 函数向 Matlab 发送数据了。

3.4.2.3 在主控板上实现扫频辨识功能

在前面两个小节中，分别实现了对扫频信号发生函数和主控板向 matlab 发送数据的功能，有了这两个基础，就可以组合扫频辨识的功能了。

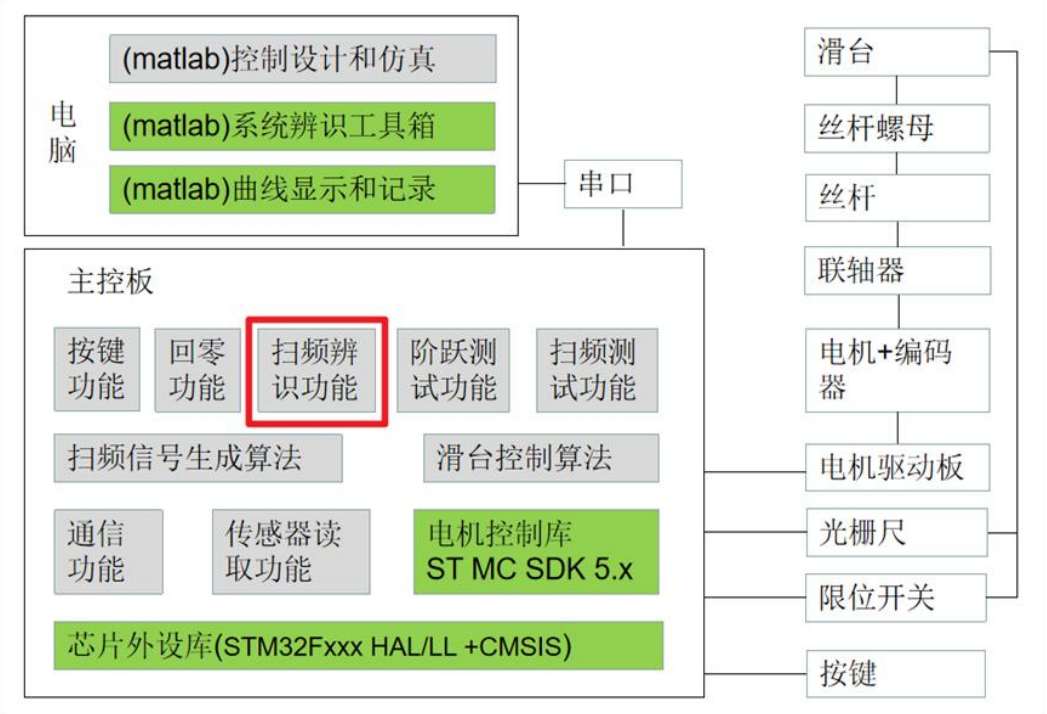


图 3.18 实现扫频辨识功能

扫频辨识功能的工作就是，将正弦扫频信号输入到电机的控制接口中，然后获取被控对象的输出，并将输入和输出发送到 matlab。在此使用的电机控制库接口函数如下图所示。在进行系统辨识时，我们的输入是电流 Iq，输出为电机的转速。输入电流可以通过 MC_ProgramSpeedRampMotor1()函数实现；测量转速可以通过 MC_GetMecSpeedAverageMotor1()函数实现。

函数名称	函数参量	函数返回	函数功能
MC_StartMotor1	void	bool	启动电机
MC_StopMotor1	void	bool	停止电机
MC_ProgramSpeedRampMotor1	speed,time	void	设定速度以及时间
MC_ProgramTorqueRampMotor1	torque,time	void	设定力矩以及时间
MC_SetCurrentReferenceMotor1	Iqref，Idref	void	设定Iq，Id参考
MC_GetCommandStateMotor1	void	MCI_CommandState_t	返回指令执行状态
MC_StopSpeedRampMotor1	void	bool	停止速度指令执行
MC_HasRampCompletedMotor1	void	bool	指令是否执行完成
MC_GetMecSpeedReferenceMotor1	void	int16	返回机械参考速度
MC_GetMecSpeedAverageMotor1	void	int16	返回平均机械速度
MC_GetLastRampFinalSpeedMotor1	void	int16	返回上次指令速度
MC_GetControlModeMotor1	void	STC_Modality_t	返回控制模式
MC_GetImposedDirectionMotor1	void	int16	返回电机转动方向
MC_GetSpeedSensorReliabilityMotor1	void	bool	返回当前速度传感器是否可信
MC_GetPhaseCurrentAmplitudeMotor1	void	Is	返回电流值

图 3.19 扫频辨识用电机库 API 函数

下面分别对这两个函数做简要说明。

1) **MC_ProgramTorqueRampMotor1()：**

该函数与 MC_ProgramSpeedRampMotor1()控制效果是一样的，不过作用对象是电机转矩（实际为电流 Iq）。当电机以 Torque 模式运行的时候，该函数就用来设置转矩（电流 Iq）的目标值。

两个参数，hFinalTorque 是运行时候的电机转矩最终的目标值，hDurationms 则是持续时间，单位是 ms。调用该函数之后，将会切换成 Torque 模式。hFinalTorque 是以 16 位有符号数表示的 Iq 值，转换成安倍（Amps）格式的时候需要使用以下公式：

$$I_{s16A} = \frac{I_{Amp} * 65536 * R_{shunt} * A_{op}}{V_{dd}}$$

根据 Workbench 中电机参数的配置可知，

$$V_{dd} = 3.3V$$
$$R_{shunt} = 0.02$$
$$A_{op} = 4.02$$

I_{AMP}是实际的电流值，其最大值不要超过额定电流值为$3.13A$，所以可以得到I_{s16A}的最大值可以计算得到是 4997，所以转矩输出的控制量 hFinalTorque 的最大值不要超过这个值。

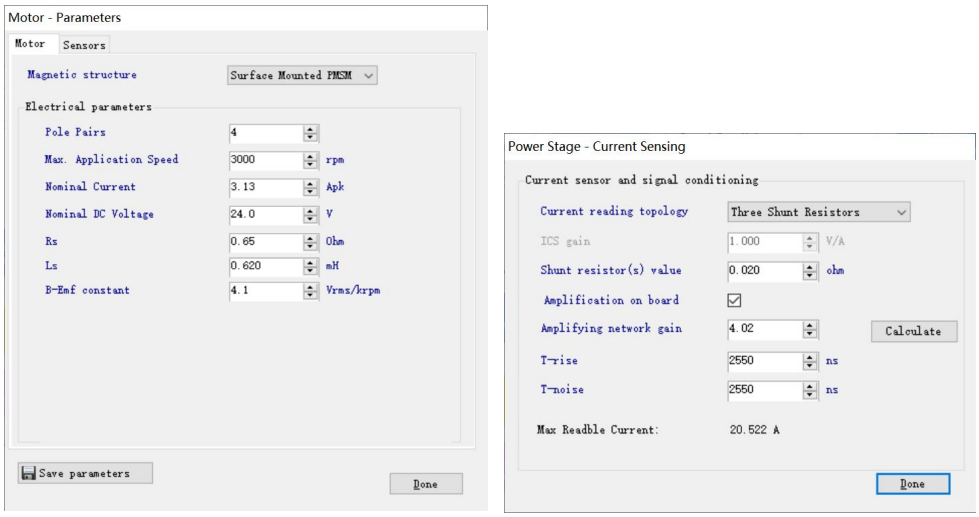


图 3.20 Workbench 中电机参数配置

2) **MC_GetMecSpeedAverageMotor1()**

该函数用于获取电机平均机械转速，返回值是电机转速的平均值,单位是[0.1Hz]。例如返回值是 25，则实际的转速就是 25*[0.1Hz] * 2*pi = 15.71 [rad/s] .

一种实现该功能的代码案例如下,鼓励同学自己开发以加深理解。

```
//*实现扫频辨识功能，该功能将被放置在 main 函数的 while(1)中运行*/
void function_sweep_identification(void)
{
  static my_sweep_t sweep = {0};

  int16_t sweep_input = 0;
```

```
int16_t sweep_output = 0;

uint32_t sys_tick = 0;
static uint32_t init_flag = 0;
static uint32_t last_sys_tick = 0;
static uint32_t start_sys_tick = 0;

// 频率在 10s 内 ，从 0.5hz 变化到 10hz，幅度为 1500 digit current
uint32_t t_period_ms = 10*1000; //10s
float f0 = 0.5;
float f1 = 10;
float Amp = 1500.0f;

float time = 0.0f;

sys_tick = HAL_GetTick();  //获取当前时刻，单位 ms
time = 0.001f * sys_tick;  //单位 s


/*进入的条件时回零成功，且按了 K1 运行键，就开始执行扫频辨识过程，
 注意 find_home_flag 是回零成功的标志位，是一个全局变量,需要在外部实现这个标志位*/
if ((find_home_flag == 1)  && (MC_GetSTMStateMotor1() == RUN))
{
    if (last_sys_tick != sys_tick)  //如果当前时刻发生了变化，这个条件每 ms 都会成立一次
    {
        last_sys_tick = sys_tick;

        if (sys_tick % 10 == 0) //通过 % 把频率从 1000hz 降低到 100hz，即每 10ms 发生一次
        {

          //初始化扫频配置
          if (init_flag == 0)
          {
              init_my_sweep(&sweep, sys_tick, t_period_ms, f0, f1, Amp);
              printf("sweep-init:k=%.5f,p=%.5f\r\n",(float)sweep.k,(float)sweep.p);
              init_flag = 1;
          }

          //获取正弦扫频信号
          sweep_input = (int16_t)run_my_sweep(&sweep, sys_tick);

          //将正弦扫频信号输入到 ST MC SDK 的力矩控制 API 中
          MC_ProgramTorqueRampMotor1(sweep_input,0);

          //获取丝杆的转速信息，单位为 0.1hz
```

```
            sweep_output = MC_GetMecSpeedAverageMotor1();


            //把时间，input, output 发送到 matlab
            send_data_2_matlab(time,(float)sweep_input,(float)sweep_output);


          }
        }
      }
  }
```

将上述程序开发后，放在 main 函数中，集成起来就能够获得扫频辨识功能。基础工作做好后，就可以正式开始系统辨识的工作了，具体操作步骤如下：

1) 将主控板串口连接电脑，将主控板上电；

2) 在 plot_mcsdk.m 中的串口号设置正确，按 F5 运行 plot_mcsdk.m

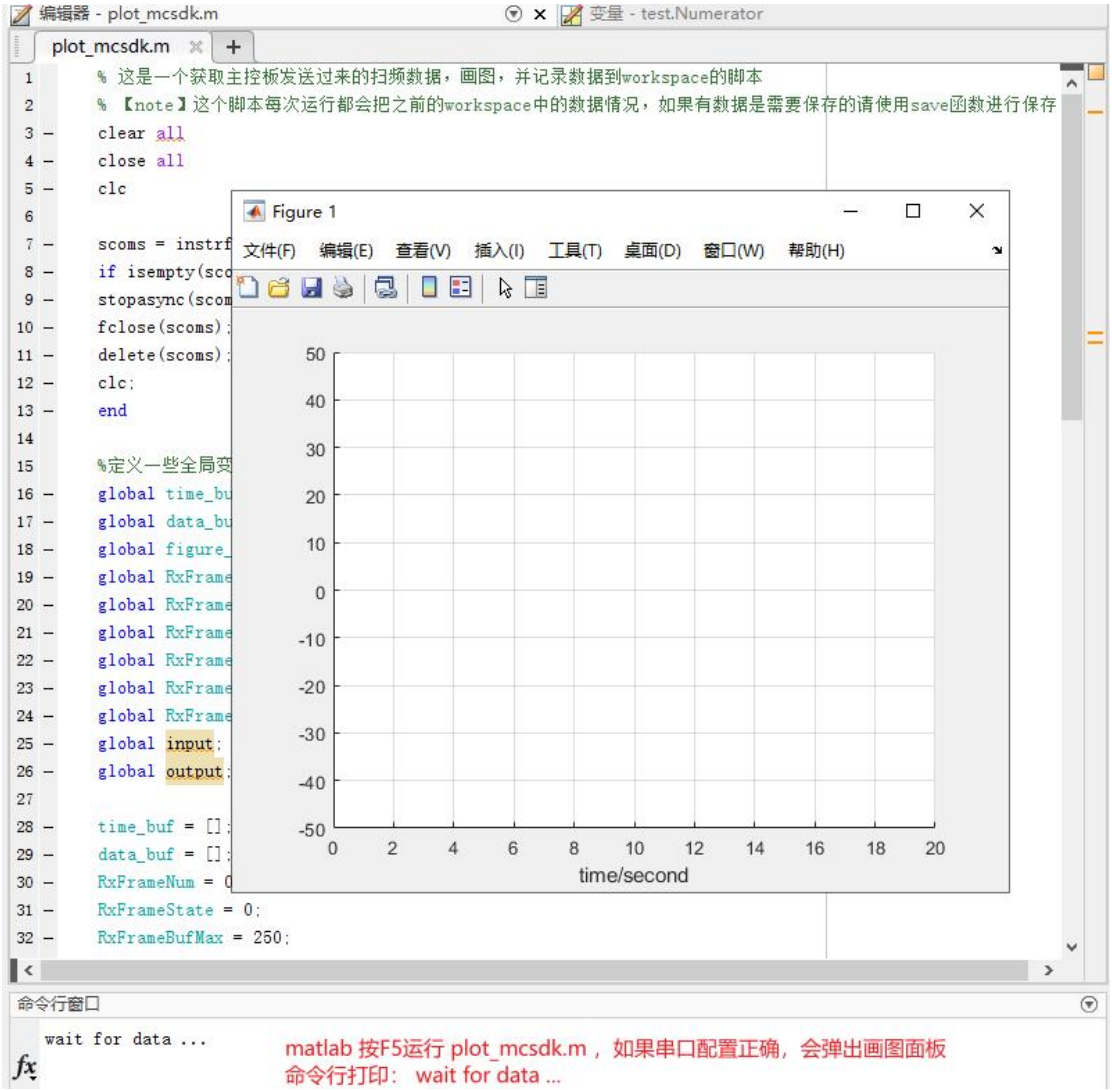


图 3.21 运行 Matlab 串口接收程序

3) 根据同学自己实现的回零功能和对应的按键，先让滑台回到中点；

4) 根据同学自己实现的扫频辨识功能，运行扫频辨识功能，使得滑台围绕中点周期性运动，运动的频率越来越快。

5) 在 matlab 中可以看到实时接收到的扫频信号曲线；

6) 因为滑台收到摩擦阻力的影响，扫频输出的 output 曲线会出现一些畸变，在完成了 3-4 个从低频到高频的扫频过程后，可以通过按键停止电机的扫频过程。

7) 因为解包的过程相对接收数据要慢一些，因此可以看到在中断接收数据后，画图工作还会进行一会儿。等待剩余未处理的字节数量变为 0 后，可以在 matlab 的命令行中通过 ctrl+c 中断数据接收的过程。

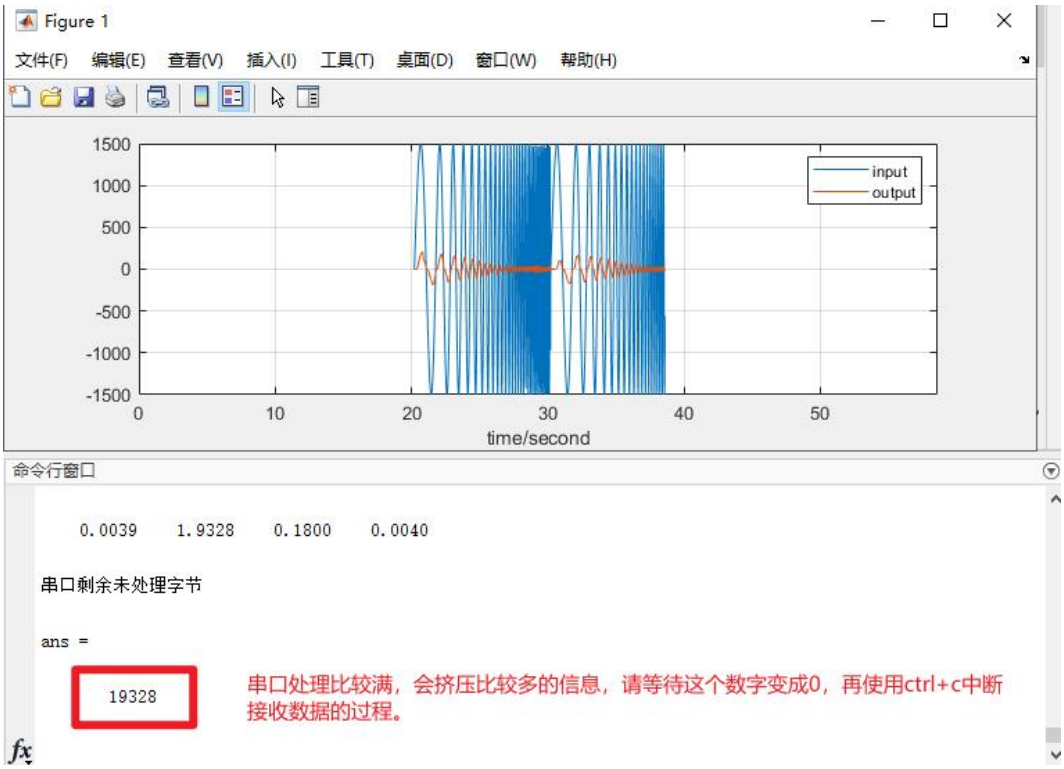


图 3.22 等待串口数据接收完成

8) 这个时候我们可以看到 workspace 中有两个变量：扫频输入信号 input 和 output, 如果我们需要多次使用这个两个数据，可以点击他们，然后另存未为.mat 文件。等到需要使用的时候，可以用 load 函数重新载入。

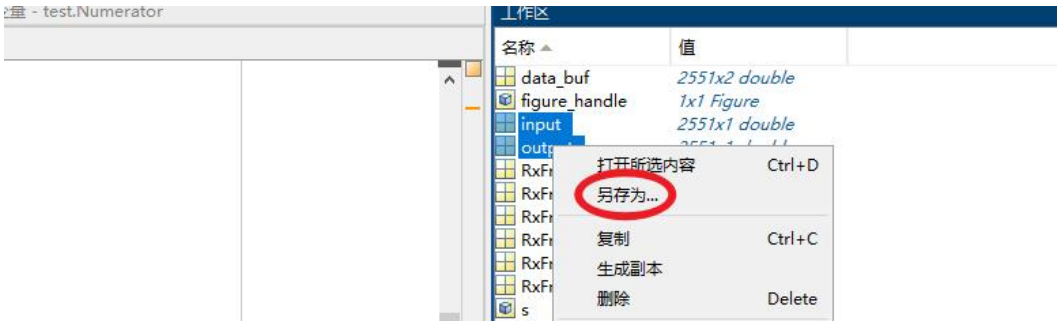


图 3.23 将数据另存为.mat 文件

3.4.2.4 使用 matlab 系统辨识工具箱，获得辨识的系统模型

Matlab 接收 STM32 数据完成后，会得到输入信号与输出信号两个变量，基于这两个变量可以对系统模型进行辨识。下面介绍一下使用 Matlab 系统辨识工具箱进行系统辨识的过程。

1) 在 matlab 的控制窗口中输入 systemIdentification，打开系统辨识工具箱，点击 Import data，选择时域数据，如下图所示。

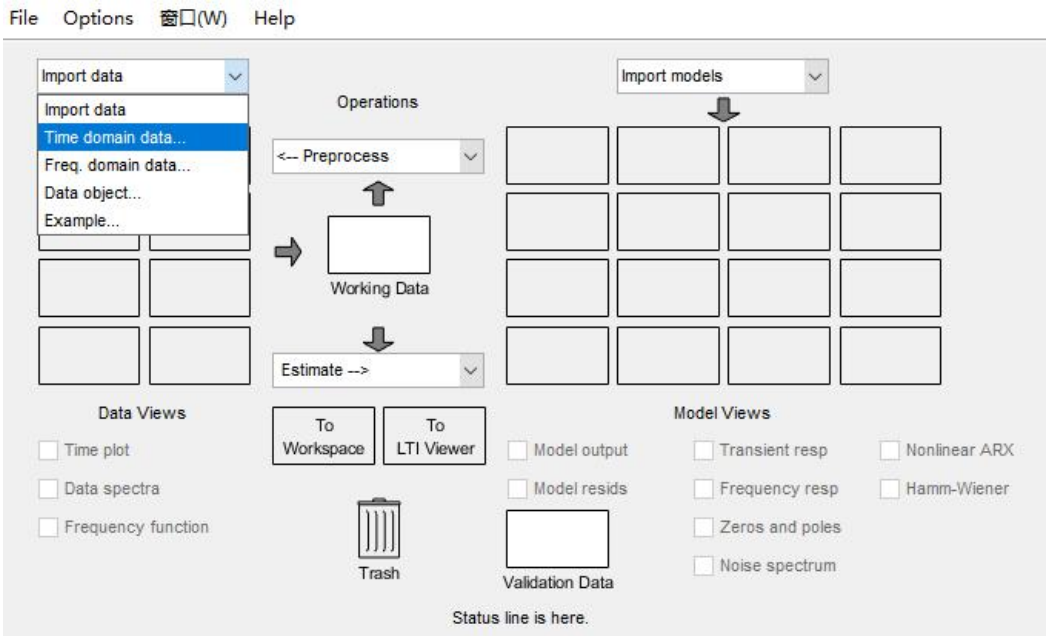


图 3.24 Matlab 的系统辨识工具箱

2) 导入输入输出数据到系统辨识工具箱中，将输入输出数据变量名称分别写在 Input 和 Output 输入框中（注意名称不要写错），Start time 设为 0，Sample time 要和数据实际采样时间保持一致。配置完成点击 Import，如下图所示。

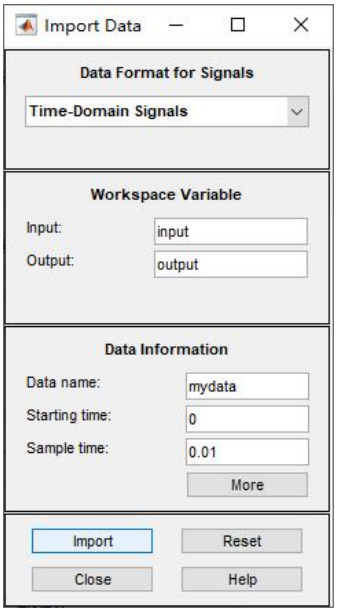


图 3.25 导入输入输出数据

3) 可以通过勾选 Time plot 来查看输入 u1 和输出 y1 曲线.

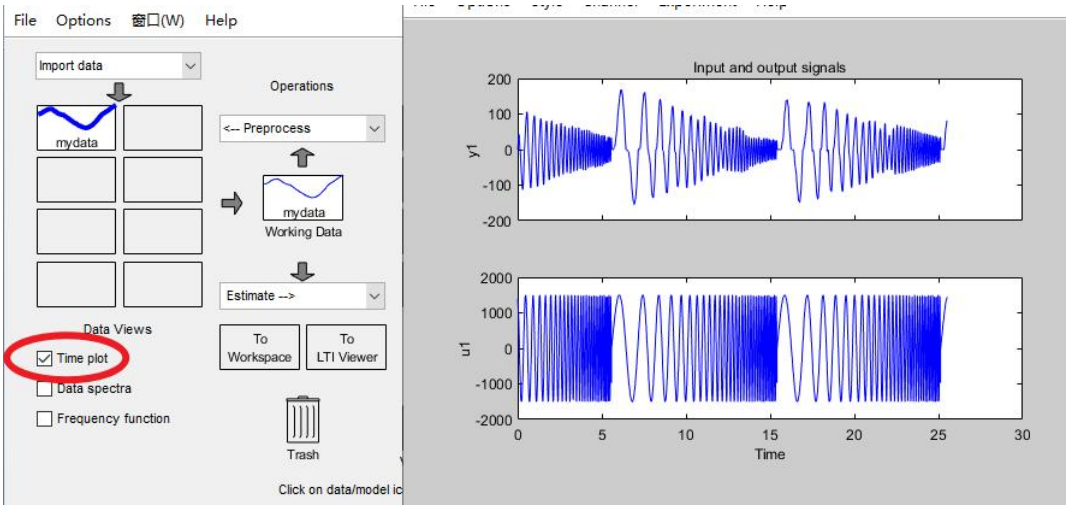


图 3.26 Time Plot 查看 u1 和 y1 曲线

4) 然后我们选择系统辨识方法 Process Models.

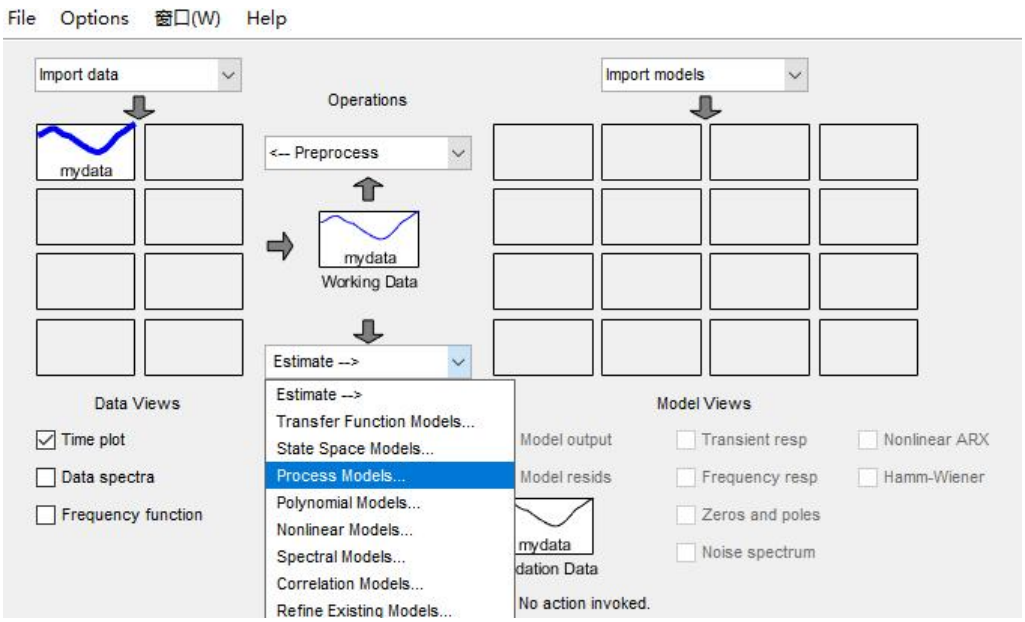


图 3.27 选择 Process Models

5) 这个会弹出一个传递函数配置面板，这个传递函数配置成什么样，就需要依据系统建模中的传递函数模型作为参考了。

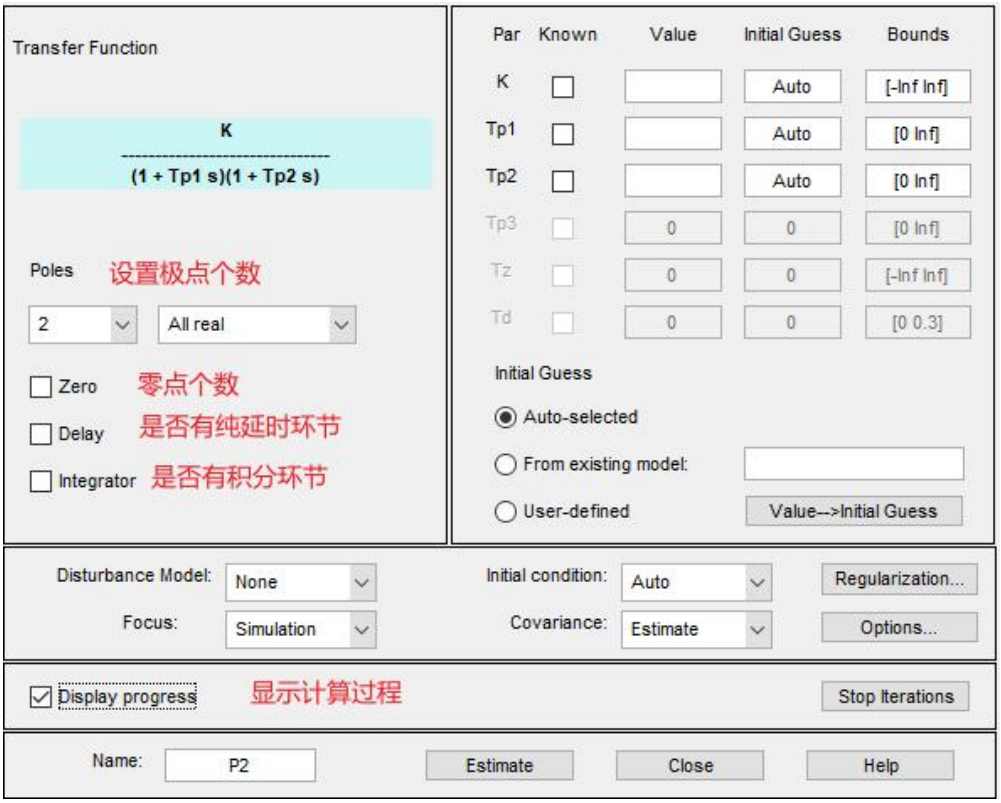


图 3.28 根据理论模型设置参数

6) 配置好传递函数的类型后，点击 Estimate 按钮，就可以看到计算过程。

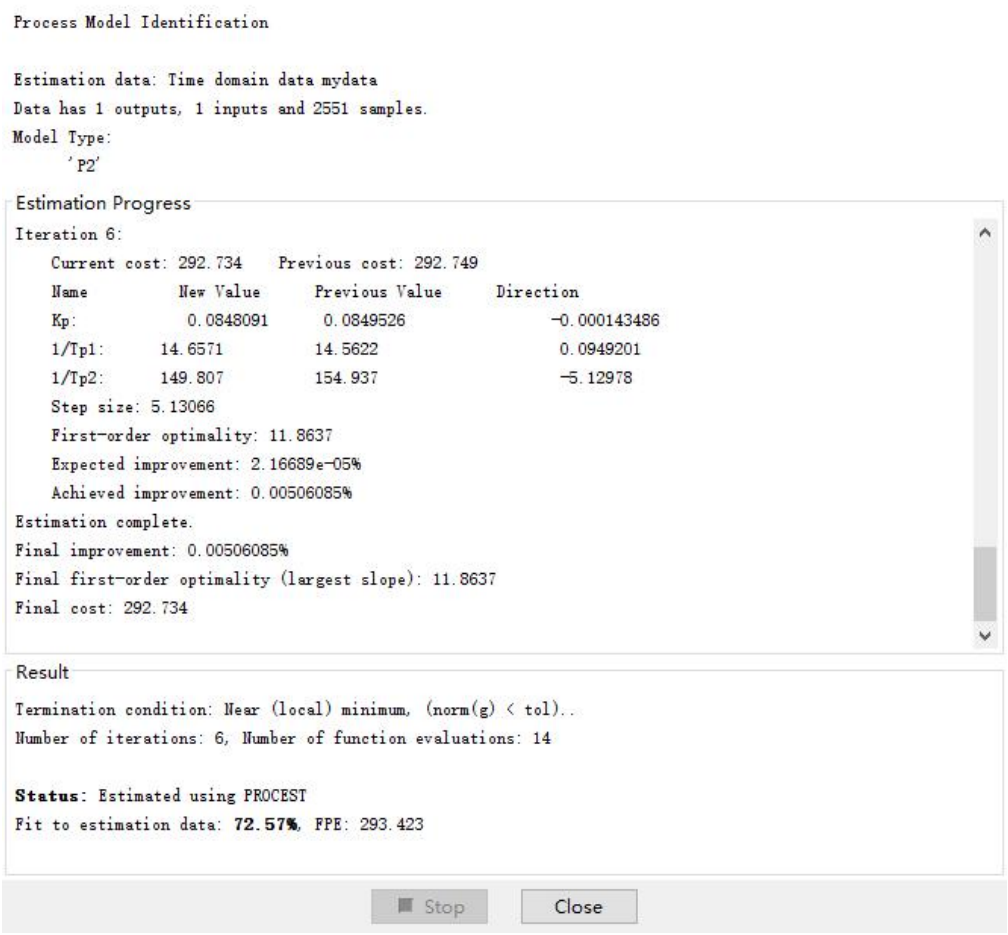


图 3.29 查看参数辨识计算过程

7) 这里最终给出了一个匹配度为 72.57%，从工程经验的角度看，因为这个滑台有比较大的摩擦，所以能够达到 70%就算是通过了，否则需要重新采集数据。经过实验检测，用这种系统辨识的方法，大部分平台能得到一个匹配度 80%左右的模型。

8) 勾选 Model output,可以看到辨识的模型输出和实际的输出 ，如下图所示：

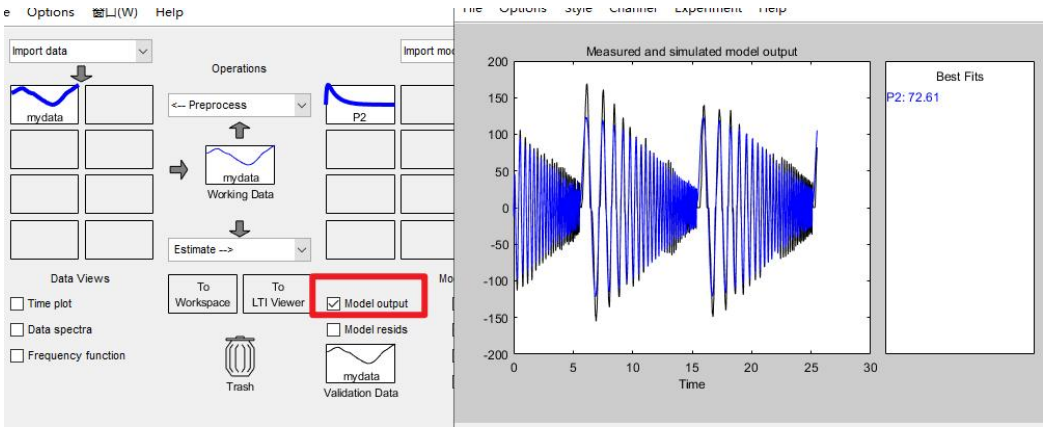


图 3.30 查看模型输出与实际输出曲线

9) 从上图可知，在低速的时候，模型存在较大的误差。

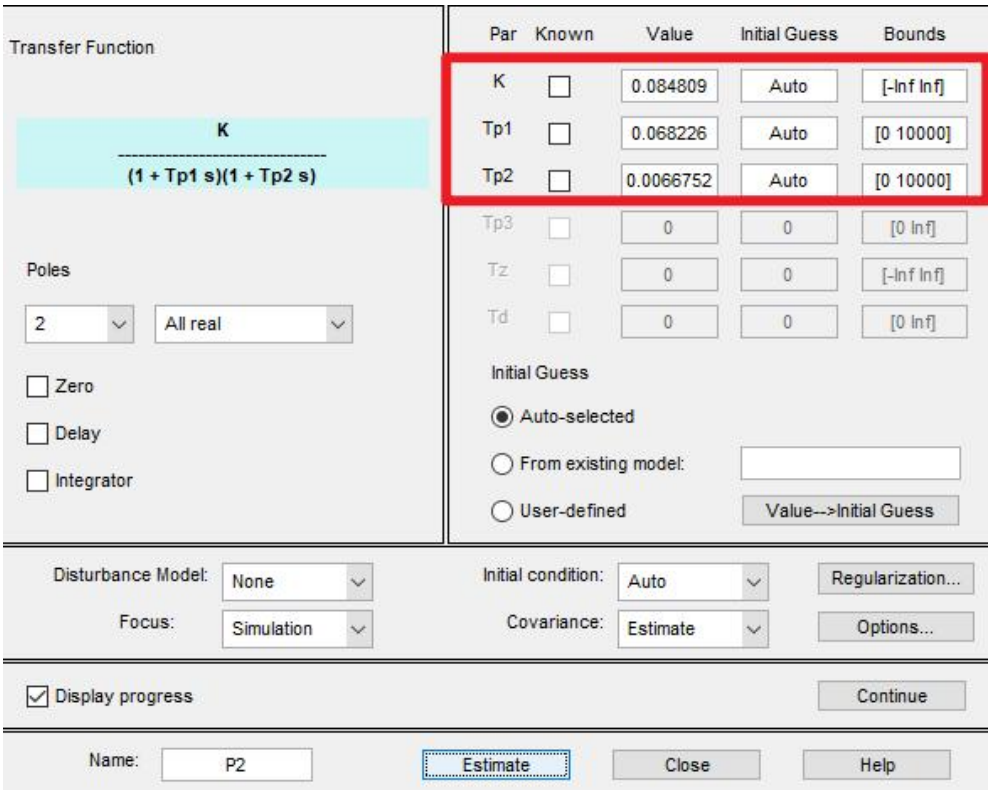


图 3.31 参数辨识结果

10) 从上面的结果看，系统有两个极点，但是时间常数相差了 10 倍。

到这里，就获得了传递函数：

$$F(s)=\frac{0.084809}{(1+0.068226s)(1+0.0066752s)}$$

$$=\frac{186.22}{(s+14.66)(s+149.8)}$$

$$=\frac{186.22}{s^2+164.46s+2196.1}$$

经过 3.4.2.1~3.4.2.4 的实现过程，我们最终完成了对该控制系统的辨识，得到了系统的模型。但是需要注意的是，不同的设备的模型是不太一样的，而且这个辨识出来的模型也是有偏差的，比如对于摩擦阻力，我们并没有对其进行建模或辨识，但是摩擦阻力在低速运行时对滑台运动的干扰还是很明显的。欢迎有能力的同学，进一步研究，把系统辨识得更加准确。

3.4.3 控制器设计

参考理论课件《第四章 控制系统的设计约束（2）》中的 4.3.4 中的内容，结合辨识的模型，设计控制器。

推荐使用：频域的方法进行控制器的设计，使得控制器的带宽，相位裕度等满足控制系统任务指标。

请同学在这里给出控制器设计的详细过程（可以是理论推导设计过程，或者 matlab 辅助设计过程）。并结合 bode 图给出相位裕度和幅值裕度的情况。

欢迎有能力的同学挑战使用理论课讲到的相对复杂一些的算法，以提高对控制算法的理解。

3.4.4 控制器仿真验证

一般为了提供算法调试的效率，在有了被控对象的模型之后，可以在 simulink 中搭建被控对象和控制算法的仿真系统。对控制算法进行快速的验证。

举例子，搭建了如下的仿真系统。

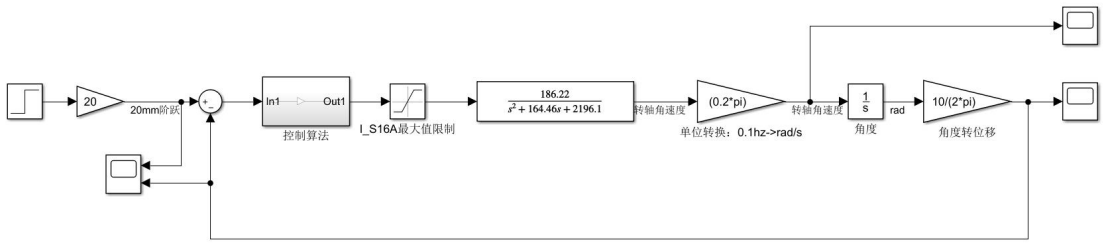


图 3.32 搭建 Simulink 仿真

其中几个局部的点需要注意：

1) 执行器有饱和限制，I_S16A 的最大值为 4997；

2) 模型输出的是 0.1hz 为单位的转轴角速度，化为标准单位需要进行单位转换。

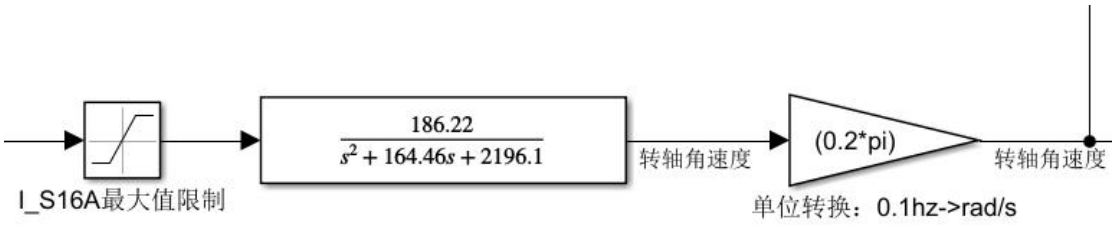


图 3.33 注意执行器饱和与角度单位转换

3) 转轴角速度积分为角度，再通过导程转换为位移，才得到滑台的位移。

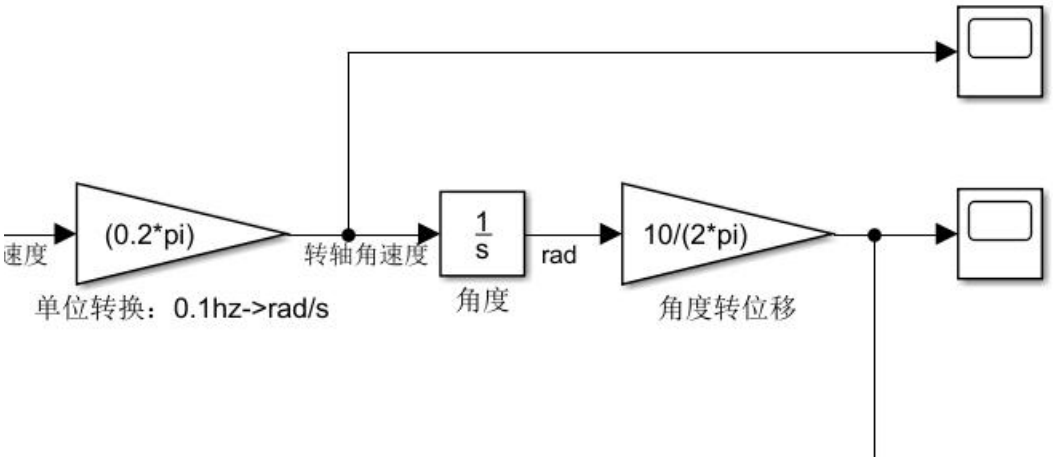


图 3.34 注意角度与位置转换

4) 控制器不一定就是这种输入输出，这里只是一个举例。比如加上多个环路，扰动观测器，前馈控制等，控制器的输入输出都会变化。这里看同学自己的构想。

5) 可以在 simulink 中选择不同的信号源头进行测试，比如可以选择 Step 信号和 Chirp Signal 号源进行阶跃响应测试和扫频跟随测试。

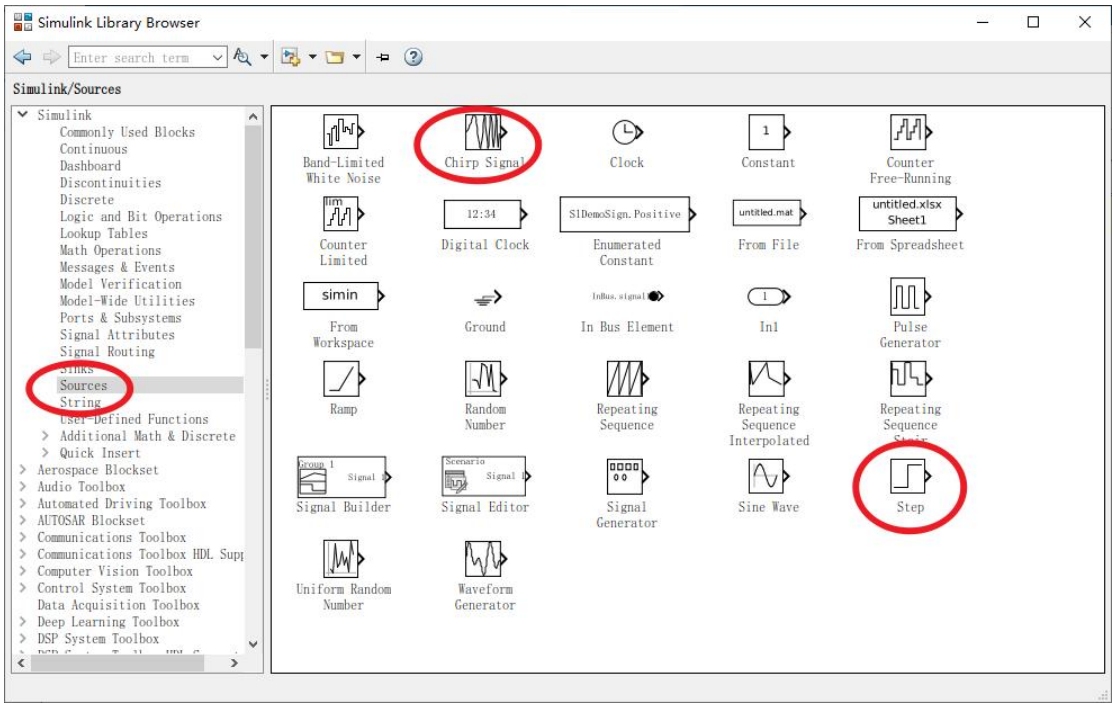


图 3.35 信号源模块位置

请同学根据自己辨识的模型，在 simulink 上完成仿真验证，使得仿真环境下，控制系统的任务指标满足要求。

3.4.5 控制程序开发

把上一节中验证好的控制器，转变为离散化的控制器，然后开发 STM32 主

控板上的控制程序；

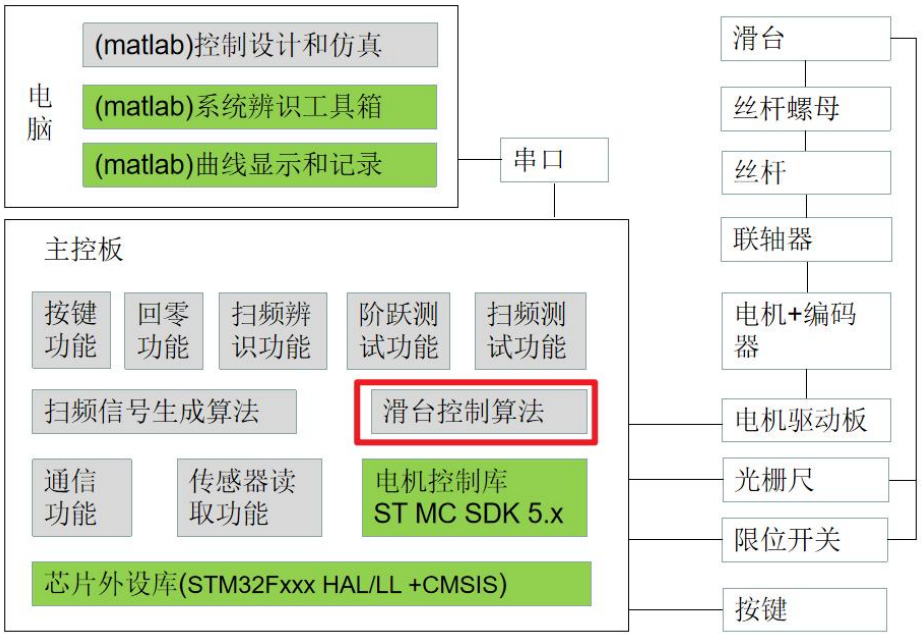


图 3.36 滑台控制算法实现

在开发程序之前，我们需要将上面开发好的连续控制器变成离控制器。这里就涉及到一个控制器离散化的过程。回忆理论课件的《第二章 控制系统的设计流程》的课件中 5.5 小节，关于数字控制设计的内容：

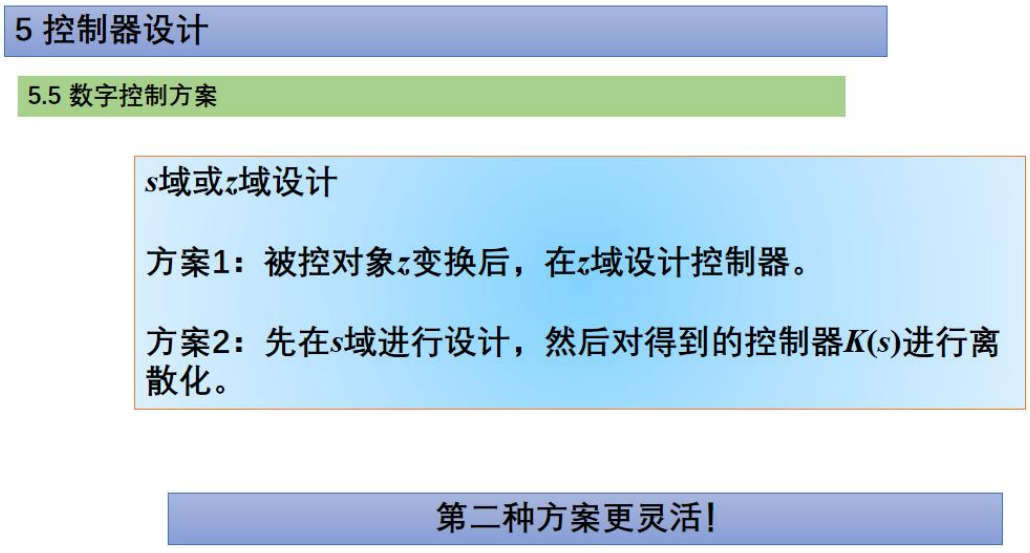


图 3.37 控制器离散化方法

这里我们举个简单的例子来指导同学，完成控制器的离散化的过程:

$$K(s)=\frac{1}{as+b}$$

如果要将其离散化为一个数字控制器，其中一种典型的方法是，采样双线性变换的方法，令：

$$s = \frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}$$

带入到 K(s)中得到：

$$K(z) = \frac{u(z)}{e(z)} = \frac{T + Tz^{-1}}{(2a+bT) + (bT-2a)z^{-1}} = \frac{\frac{T}{2a+bT} + \frac{T}{2a+bT}z^{-1}}{1 + \frac{bT-2a}{2a+bT}z^{-1}}$$

其中 T 为离散控制系统的控制周期；
通过上面的方程可以得到

$$u(k) = -\frac{bT-2a}{2a+bT}u(k-1) + \frac{T}{2a+bT}e(k) + \frac{T}{2a+bT}e(k-1)$$

写成如上的离散方程，那么 k 时刻的控制量 u(k)的表达式就是非常清晰的，非常容易通过程序来实现，代码展示就不再具体展开了。

上面介绍的只是一个简单例子，如果整个控制算法非常复杂，那么通过手动推导的方式就会非常麻烦，这里推荐大家使用 matlab 进行求解：

这里举个使用 matlab 进行求解的例子。假设一个复杂的控制器

```
ks =

     20 (s+3)
  --------------
  s (s+10) (s+2)

Continuous-time zero/pole/gain model.
```

可以使用如下的 matlab 脚本，转变为离散的控制器

```
%这个程序，介绍了如何把连续的控制器离散化的例子
clear all
close all
clc

s = zpk('s') ;
%设置你的控制器
ks = (s+3)/(s*(0.1*s+1)*(0.5*s+1))
%设置控制器周期 Ts
Ts = 0.01;
%使用 c2d 进行双线性变换；
kz = c2d(ks,Ts,'tustin')
```

```
%使用 tf 转变为离散的传递函数
kz_tf = tf(kz)
```

得到结果如下图所示：

```
kz_tf =

  0.0004785 z^3 + 0.0004927 z^2 - 0.0004503 z - 0.0004644
  ---------------------------------------------------------
            z^3 - 2.885 z^2 + 2.772 z - 0.8868
```

3.4.6 控制系统调试

到这一步骤，同学们可以开始对控制系统进行实物调试。为了完成性能的测试，我们需要开发两个测试用例：阶跃测试功能和扫频测试功能。

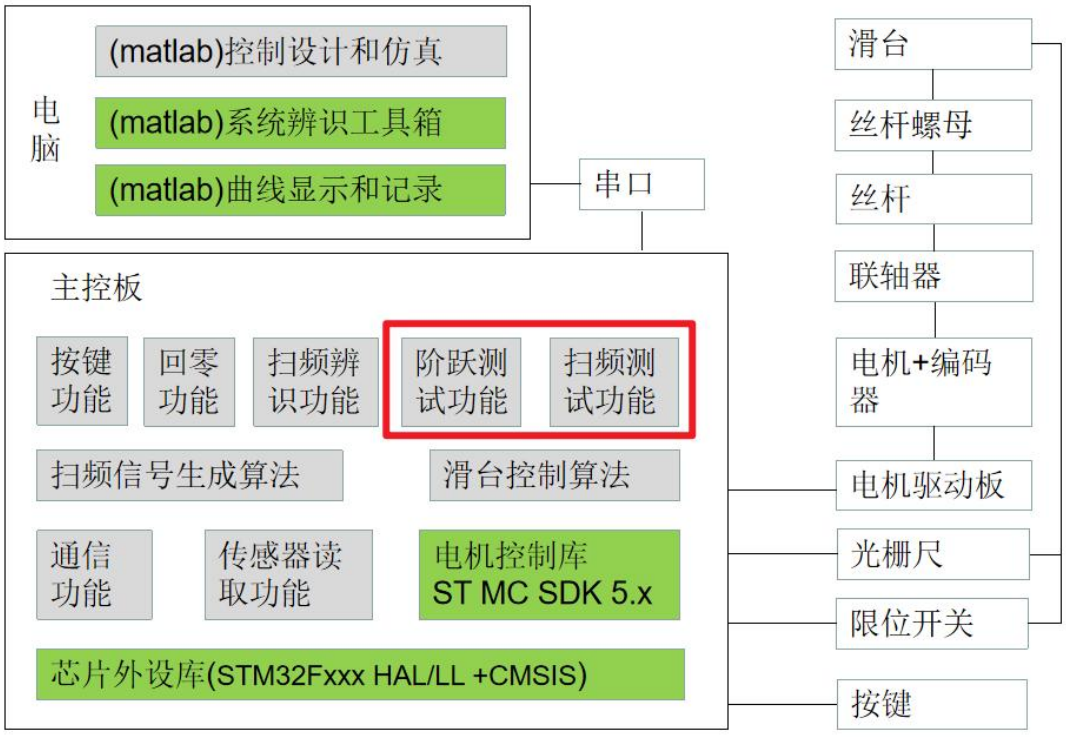


图 3.38 完成阶跃测试与扫频测试

在控制器仿真这个环节中，我们就已经对使用了 simulink 中的 step 和 chirp 信号进行控制系统性能测试，而现在要做的，就是在 stm32 主控板上实现这两个功能。

阶跃响应测试伪代码如下：

```
/*实现一个阶跃测试函数，放在 main 的 while(1)中进行阶跃测试*/
void function_step_test(void)
```

```
{

  /*1.定义局部变量和静态变量*/

  /*2.初始化你的控制器*/

  if (/*3.启动 step 测试功能条件满足*/)
  {

    if (/*4.分频器确定当前 tick 为控制器运行的 tick*/)
    {

        /*5.获取光栅尺的当前位置  fdk (mm)*/

        /*6.指定阶跃的距离 Amp 统一定义为 20mm*/

        /*7.一次性计算当前参考值  ref = fdk+Amp*/

        /*8.将 ref 作为你的位置控制器的输入命令，调用控制器*/

        /*9.使用  send_data_2_matlab,  把时间，ref, fdk 发送到 matlab 进行显示*/
    }
    else
    {
        /*10. do some thing if need*/
    }
  }
  else
  {
        /*11. do some thing if need*/
  }
}
```

扫频跟随测试伪代码如下：

```
/*实现一个扫频测试函数，放在 main 的 while(1)中进行扫频跟随测试，验证 w 位置闭环带宽*/
void function_step_test(void)
{

  /*1.定义局部变量和静态变量*/

  /*2.初始化你的控制器*/
```

```
/*3.初始化你的正弦信号发生器为 幅值 20mm, 频率在 10s 内从 1hz 变化到 2hz*/

if (/*4.启动扫频测试功能条件满足*/)
{

  if (/*5.分频器确定当前 tick 为控制器运行的 tick*/)
  {

      /*6.获取光栅尺的当前位置 fdk (mm)*/

      /*7.每个 tick 都获取最新的扫频信号输出，作为当前参考值 ref = sweep(now)*/

      /*8.将 ref 作为你的位置控制器的输入命令，调用控制器*/

      /*9.使用 send_data_2_matlab, 把时间，ref, fdk 发送到 matlab 进行显示*/
  }
  else
  {
      /*10. do some thing if need*/
  }
}
else
{
      /*11. do some thing if need*/
}
}
```

请根据上面的伪代码完成测试用例的测试，并在确保控制系统达到设计要求。有能力的同学可以挑战让性能指标在达到标准要求的基础上，更精进一些。

3.5 任务验收

3.5.1 当堂验收

1) 对于输入为 2cm 的阶跃信号，满足任务描述中的指标要求：要求 95%的上升时间不超过 0.1s；超调量不大于 10%；稳态误差要求小于 1mm。

2) 对于输入为 1Hz 的正弦信号，能实现位置跟随，且正弦波幅值衰减不超过-3dB。

3.5.2 实验报告

根据实验报告模板要求，记录整个控制系统设计与实现过程。